University of Windsor

LECTURE 6 – TESTING (PART 4)

Master of Applied Computing

COMP-8117 : Advanced Software Engineering Topics

Dr. Aznam Yacoub – aznam.yacoub@uwindsor.ca
School of Computer Science

SCHEDULE

- Introduction
- Test case design
- Conclusion

SYSTEMATIC TESTING METHODS

- Remind : Systematic testing approach needs
  - A system (rule) for creating tests
  - A measure of completeness
- Need an easy way to create test cases (what)
- Need an easy way to run tests (how)
- Need an easy way to decide when we're done (enough)

SYSTEMATIC TESTING METHODS

- Testing is a continuous process
- Testing is also an engineering activity, which requires a methodology and a lifecycle

INTRODUCTION

- Design of tests for a system is a hard engineering problem.
- Designing effective tests requires a set of procedures from an initial high level test strategy down to detailed test procedures
- Test design stages: test strategy, test planning, test case design, test procedure

INTRODUCTION

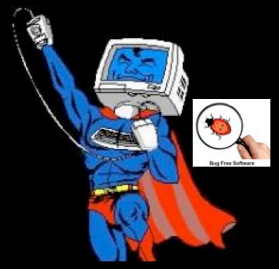
SOFTWARE TESTER

What my friends think I do

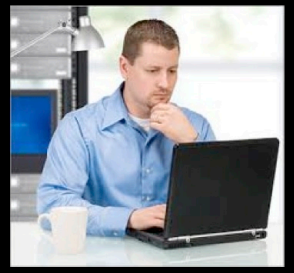
What my mom thinks I do

What society thinks I do

What programmers think I do



What I think I do



What I actually do

University of Windsor

TEST STRATEGY

- Overall approach to testing for a software development organization
- Specifies the levels of testing, methods, techniques and tools
- Part of the Quality Plan, to be followed and reported by all members of the project

TEST STRATEGY

- Big Bang testing strategy
  - Test the entire software once it is complete.
  - Exercise: Which lifecycle ?

TEST STRATEGY

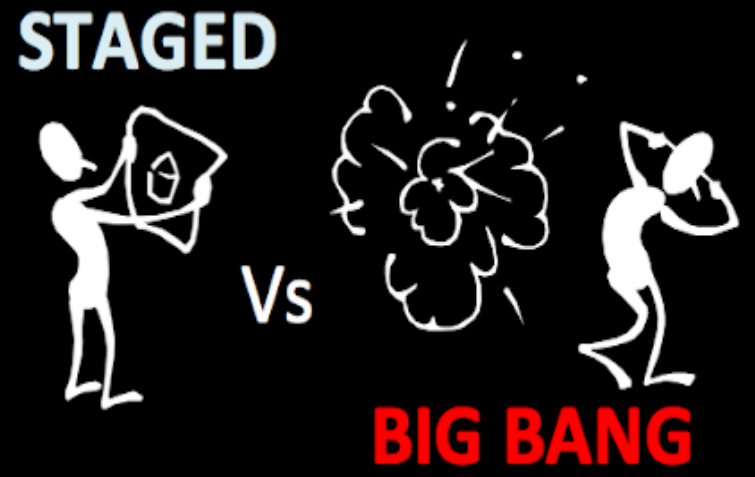
- Incremental testing strategy
  - Test the software in phases (unit, integration, system)
  - Exercise : Which lifecycle ?

TEST STRATEGY

- Incremental testing strategy
  - Test the software in phases (unit, integration, system)
  - Incremental testing can be bottom-up (using drivers) or top-down (using stubs)
  - Bottom-up is easier to perform but means the whole program behaviour is observed at a later stage of development

TEST STRATEGY

- Big-Bang or Incremental ?
  - Big bang : good for small program
  - Incremental : good for complex project – Easy to identify errors, East to fix errors



TEST PLAN

- Specifies in details how the test strategy will be carried out for the project
- Especially:
  - Items to be tested (functions, methods, modules…)
  - Level (unit, integration, system, acceptance)
  - Order (Priority)
  - Test environment
- Project-wide or separate plans for unit, integration, system, acceptance testing

TEST PLAN

- Test plan in Software Engineering follows a standard.
- IEEE-829 : https://ieeexplore.ieee.org/document/4578383
- ISO/IEC/IEEE 29119 Software and systems engineering -- Software testing (Part 1 – Part 5)

TEST CASE DESIGN

- Remember : testing cannot guarantee the absence of all errors.
- Test everything impossible => Test case design is important.
- Key issue: *what subset of all possible test cases has the highest probability of detecting the most errors ?*

TEST CASE DESIGN

- Specify a set of test cases for each item to be tested at each level
- Test case specifies:
  - How the implementation of a particular functional requirement or design unit is to be tested
  - How we will know if the test is successful

TEST CASE DESIGN

- Include test cases to test both positive (the software does what is should) testing and negative (it doesn’t do what it shouldn’t) testing
- Test cases are specified separately at each level: unit, integration, system, acceptance
- Documentation of tests = test specification for the level

TEST PROCEDURE

- Specifies the process for conducting test cases
- For each item or set of items to be tested at each level of testing, specifies the process to be followed in running and evaluating the test cases for the item
- Especially:
  - Use of test harnesses (programs written solely to execute the test cases), test scripts (automated procedures for running sets of test cases), or commercial testing tools.

TEST REPORT

- Document the test results
- Ouput of test exécution are saved in a test results file, and summarized in a report
- Test reports are concise, easy to read and clearly point out failures or unexpected results
- Use of standardized forms, for easy comparison with future test executions

FOCUS ON TEST CASE DESIGN

- As a software engineer, it's important to understand how to write efficient test cases.
- Focus on dynamic testing:
  - Black-Box : written during the specification and design phases
  - White-Box : written during design and programming phases
- Important: don't let the programmer writes the test cases !

FOCUS ON TEST CASE DESIGN

- Least effective methodology : random-input testing with random input values
- Use both black-box and white-box approach

- Black-Box:
  - Equivalence partitioning
  - Boundary-value analysis
  - Cause-Effect graphing
  - Error guessing

- White-Box:
  - Statement Coverage
  - Decision Coverage
  - Condition Coverage
  - Decision-condition coverage
  - Multiple-condition coverage

FOCUS ON TEST CASE DESIGN

- Recommended Procedure:
  - Develop test-cases using black-box methods then develop supplementary test cases with white-box methods.
- White-Box Methodologies => Part 5
- Black-Box Methodologies => Part 6

REFERENCES

This lecture is based on:

- Introduction to Software Testing – Ziad Kobti – University of Windsor
- Test – Amine Hamri – Aix-Marseille University
- CSCI3060U – Jeremy Bradbury - Ontario Tech University
- CSCI 5828 – Kenneth Anderson
- The Art of Software Testing – Glenford Myers

