Testing

Chapter 23

(the lecture contains content not covered in the textbook. Let me know if you have questions regarding the lecture slides.)
Admin stuff

• Projects selection
• Design proposal due Oct. 31\textsuperscript{st} (next Thursday)
Testing

• Who conduct testing?
  • Developers, Q&A, clients

• What are the different stages of testing?

• Why do we need testing?

• *How much resources are spent in testing?*
Testing

• Who conduct testing?
  • Developers, Q&A, clients

• What are the different stages of testing?
  • unit testing, integration testing, system testing, acceptance testing

• Why do we need testing?
  • Catch bugs (defect testing)
  • Check if I follow all the requirements (Validation testing)
  • Providing “documentation” in XP
Ariane 5 story
Outline

• How to judge the quality of a test suite
• How to design test suite
  • Manually
  • Automatically
How to judge the quality of a test suite
When can we stop testing?
When is sufficient?
When can we stop testing?
When is sufficient?

• Exhaustive testing
• Random testing
• Coverage-guided testing
Coverage

- Testing property
- Testing space
- Coverage (%)
- Coverage criteria (coverage > K %)
Classical coverage criteria

- Statement coverage
- Branch coverage
- Path coverage
- Data-flow coverage
- Others ...

*Background: control flow graph; data flow graph*
Coverage

- **Statement coverage**
  - lines of code ( # lines executed / total # of lines)

- **Branch coverage**
  - # of branch decisions / # of branches x 2
  - Loop treated similarly as branch

- **Path coverage**
  - # of unique path covered / total # of unique paths
  - Often impractical to use
Relationship among coverage criteria

• 100% coverage may be infeasible to achieve
• 100% stmt coverage → 100% branch coverage?

• 100% branch coverage → 100% statement coverage?

• Correct under a 100% coverage testing → is bug free?
Relationship among coverage criteria

• 100% coverage may be infeasible to achieve
• 100% stmt coverage $\rightarrow$ 100% branch coverage?
  • No
• 100% branch coverage $\rightarrow$ 100% statement coverage?
  • Yes
• Correct under a 100% coverage testing $\rightarrow$ is bug free?
  • no
Program’s graph representation

- Control flow graph (call graph)
- Data dependency graph
How to compute coverage (automatically)?
Cyclomatic complexity & basis path set testing

- Cyclomatic complexity
  - Based on program flow graph
    - Calculated by $E - N + 2$
  - Represents # of (linearly) independent paths in a graph
    - If one path covers at least one edge/node not covered by existing paths, it is independent.
- Basis path set testing
  - Simplification from path-coverage testing
  - Full test space size = $E - N + 2$
Data flow testing coverage

• DU coverage
  • Exercise every pair of define-use pairs
What is a “good” test set?

- Achieve good coverage (~100%)
- Little redundancy
  - How to judge redundancy?
How to design test suites
How to design test cases?

• Black box
• White box
• Random
• ...

How to design good test set manually?

• White-box testing
How to design good test set manually?

• White-box testing
  • Obtain the list of test properties to cover
  • Cover at least one new property at a time
  • Cover all properties that can be covered
    • Some properties may be infeasible to cover
How to conduct black box testing?
How to conduct black box testing?

• Equivalence class
  • Divide the input spaces into several equivalence classes; test at least one input in each class

• Boundary cases
  • If the expected input is a range of value, ...
  • If ..... is a set of value, ...
  • If ..... is a string, ...

• Common bug patterns

• Fuzz testing
Integration testing

• Use special values as function parameters
How to automatically generate test set?

• Automated random testing
  • Non-structural inputs
  • For structural inputs
    • For even more structural inputs (how to test a compiler?)

• Coverage-oriented testing
Can testing prove bug free?

• No!

• What is the implication of 100% path coverage?
Non-functional testing

• Performance testing
• Security testing
• ...

Not covered in lecture; will not appear in quizzes or exams.
Misc.

• To cover later, if we have time
• ...
• Mutation testing
• How to save regression testing effort?
• Can we test only part of the program?
• Research topics on testing
• Code Hunt Game

Didn’t have to cover in lecture.