Aalto University
School of Science

Testing, part 2

CS-C2105, Programming studio A
CS-C2120, Programming studio 2

9.2.2024

Contents

- Software failures
- How to design tests
- Practical hints
- Various aspects of testing
- Software development process

What can we test?

- Program functionality
  - Software meets the given requirements
- Program correctness
  - Software gives correct responses to ***all kinds*** of inputs
- Performance testing
  - Performs its functionality in acceptable time
- Usability testing
  - User interaction with the software is acceptable

What can we test...

- Software works on the desired platforms
  - Different operating systems
  - Different devices
- Acceptance testing
  - Software meets the general requirements of the customer

Some more terminology

- Alpha testing
  - Testing the feasibility of the initial software (or prototype) among potential customers
- Beta testing
  - User acceptance testing for a limited audience
- Functional vs. Non-functional testing
  - Functional: what the program should do?
  - Non-functional: other aspects like performance, usability, scalability, …
- Installation testing
  - Whether the installation process works correctly

Some more terminology…

- Regression testing
  - Running a series of tests to discover if anything is broken after a major change in software
  - Typically ready-made regression test sets
- Smoke testing
  - Testing whether it is worthwhile to proceed with further testing
- Stress testing
  - Testing the limit capacity of operation, to discover when the performance breaks down.
- Internationalization and localization
  - Testing that the software works in different languages and geographical / cultural areas.

Different testing processes

- Static testing
  - Code reviews, walkthroughs in collaboration with a peer.
  - Identifying dead code
- Dynamic testing
  - Executing program with test cases

Different testing approaches

- White-box testing/glass box testing
  - Seeks to show that internal structures / algorithms within program / program unit work correctly.
  - Usually carried out in unit testing level
- Black-box testing
  - Seeks to show that the program / program unit produces correct output without considering how it does it (even with no access to it)
- Gray-box testing
  - Have access to source code but perform tests as in black-box testing.

Test quality

- How widely the test cases cover the code.
  - Function coverage
  - Statement coverage
  - Branch coverage
  - Condition coverage
  - Path coverage
- Fault injection
- Mutation testing

Debugging and user interfaces

- Debugger is a highly useful aid in many cases.
  - See A+ page ”Finding and fixing errors”
- However, debugging graphical user interfaces can be painful.
- Why?
  - Graphical user interface is based on processing *events (*mouse click, button click, key click, …) which are processed separately.
  - When you follow program execution, the program control jumps into event processing, which may be confusing.

Debugging and user interfaces…

- Jumping between uninteresting GUI methods and the actual logical code in unexpected ways is disturbing, if you try to follow progress step-by-step.
  - Setting breakpoints only in logical code is a partial solution.
  - But keeping track on which active method call you are investigating may be cumbersome.

Debugging and user interfaces…

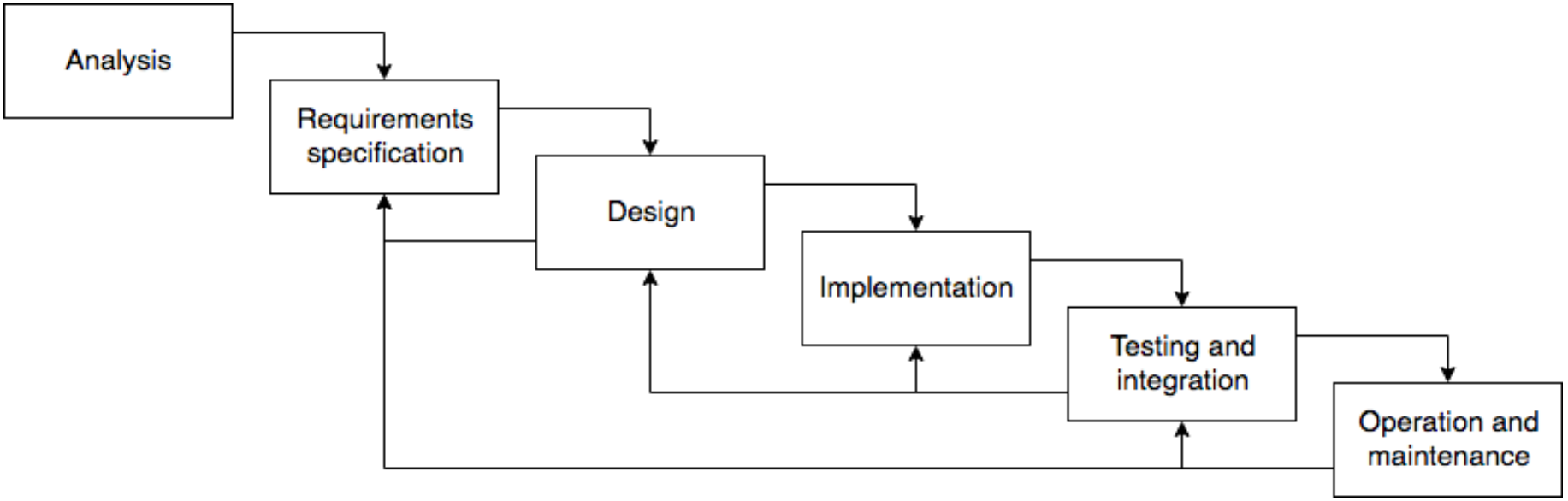
- One option is to separate the GUI code as well as possible from the logical code, and test it separately
  - Use stubs or mocks to help you to provide minimal data for testing and the user interface can deliver and show data appropriately.
- And, implement a logical part of the program using command line interaction first (or stubs / mocks) to provide necessary UI data.
  - Test that the logic works properly before you integrate the parts, followed by *integration testing*

Contents

- Software failures
- How to design tests
- Practical hints
- Other aspects of testing
- Software development process

Software development processes

- Waterfall model



Software development processes

- Agile software development
  - Development is *iterative, incremental, evolutionary*
  - Works in short cycles covering planning, analysis, design, coding, unit testing, and acceptance testing.
  - Works in close collaboration with customers
  - *Scrum* is one agile framework having 2 week sprints (and there are many others)

Software development processes

- TDD (test driven development)
  - Turns requirements into tests
    1. Add a new test
    2. Run all tests and see if the new test fails
    3. Write code that addressed the new test
    4. Run tests and revise code until all tests pass
    5. Refactor code
    6. Goto 1

Some future courses

- CS-C3150 Software Engineering
- CS-C3180 Software Design and Modelling
- CS-C2130 Software Project 1
- CS-C2140 Software Project 2