Introduction to Software Engineering

Introduction to Software Testing

Daniel Sinnig, PhD
d_sinnig@cs.concordia.ca

Department for Computer Science
and Software Engineering

Université Concordia University

29-July-14

What is software testing?

"Software testing consists of the dynamic verification of the behavior of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the specified expected behavior." – SWEBOK definition – ISO TR 19759

"Testing is the process of executing a program with the intention of finding errors." – Myers

What are the goals of testing?

- To find as many faults and detect as many failures with limited resources
- To demonstrate correct program execution
  - Show that the program is correct for a certain input.
    - Set of possible inputs, which might be close to infinity
    - If we test it only on one input => program is correct for that input
    - If we test it for 10000 different input: The program is correct for the 10.000 input.
- To instill confidence in the correctness of the software
  - Through each successful test case our confidence for the software is growing

How does this quote by Dijkstra relate to the goals of testing?

> "Testing can show the presence of bugs but never their absence."

Software errors, faults, failures and incidents

- An **error** (or mistake) is something people make. It is a slip-up or inappropriate decision by a software developer (or other project member) that leads to the introduction of a fault or defect.
- A **fault** (defect, bug) is the result of an error: inaccurate requirements text, erroneous design, buggy source code etc. It is a flaw in any aspect of the system that contributes, or may potentially contribute, to the occurrence of one or more failures.
- A **failure** (incident) is is the program's actual incorrect or missing behavior. It occurs when a fault executes. A fault won't yield a failure without the conditions that trigger it.
- An **incident** is a characteristic of a failure that helps you recognize that the program has failed.

Testing vs Debugging

- Testing is concerned with confirming the presence of faults
- Debugging is concerned with locating and repairing these faults

Test Cases

- Work product (artifact) of software development
- A **good test case** is one that has a high probability of finding an as-yet-undiscovered bug.
- Each test case consists of the following (ideally):
  - Identifier
  - Statement of purpose
  - Preconditions
  - Inputs and expected outputs
  - Expected post-conditions
  - Execution history (date, tester, version, etc.)

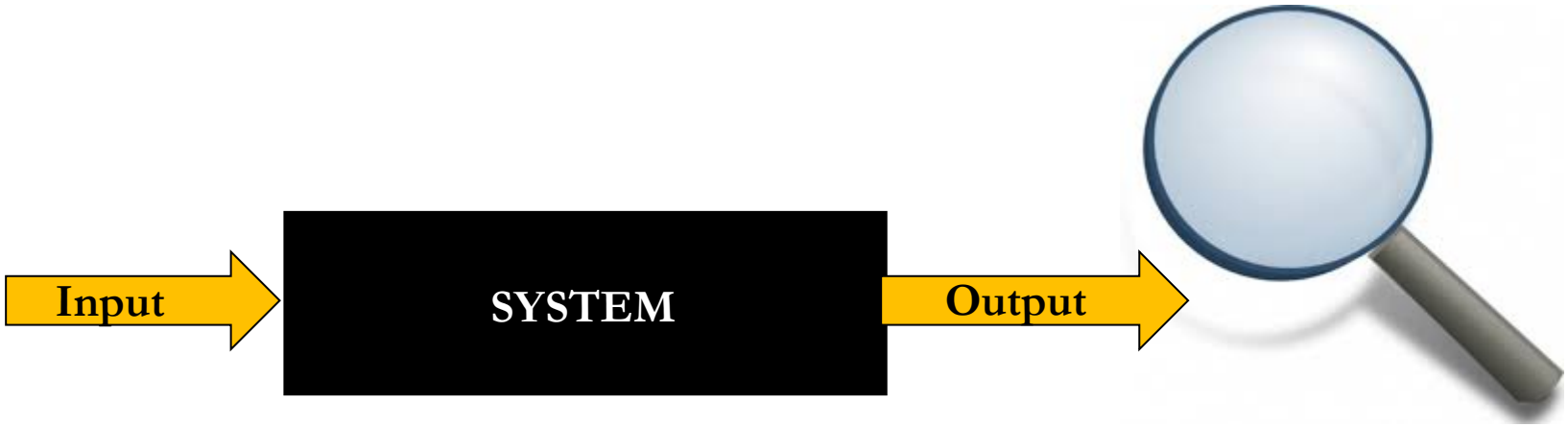
Identifying Test Cases

- Not all test cases are significant.
- Impossible to test everything in a reasonable amount of time or money
  - There are enormous numbers of possible tests. To test everything, you would have to:
    - Test every possible input to every variable.
    - Test every possible combination of inputs.
    - Test every possible sequence through the program.
    - Test every hardware / software configuration, including configurations of servers not under your control.
    - Test every way in which any user might try to use the program.

Identifying Test Cases

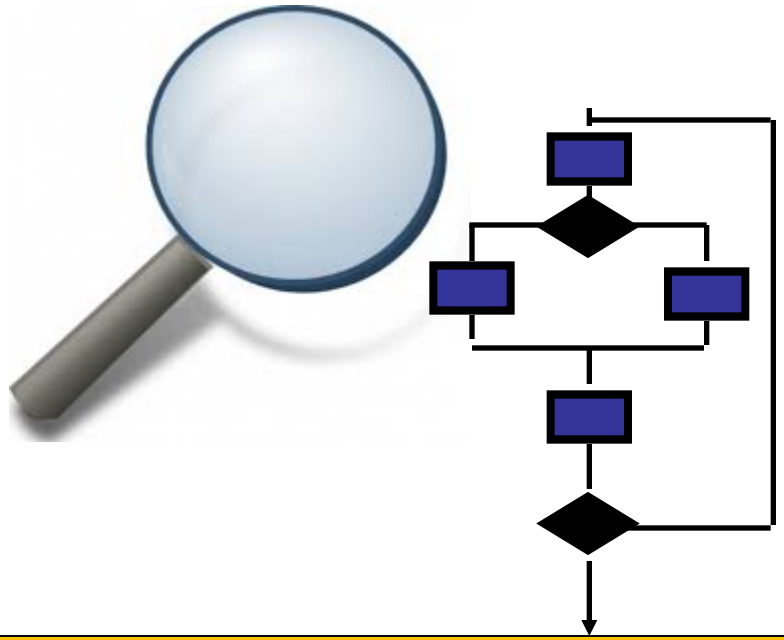
- The essence of software testing is to determine a set of test cases for the item to be tested.
  - A good test case is one that has a high probability of detecting an undiscovered defect, not one that shows that the program works correctly
- Approaches to derive test cases:
  - **Functional (black-box)**
    - Derived from specs
    - Reusable even if code changes
  - **Structural (white-box)**
    - Derived from code documentation
    - Test coverage metrics

Functional (black-box) testing



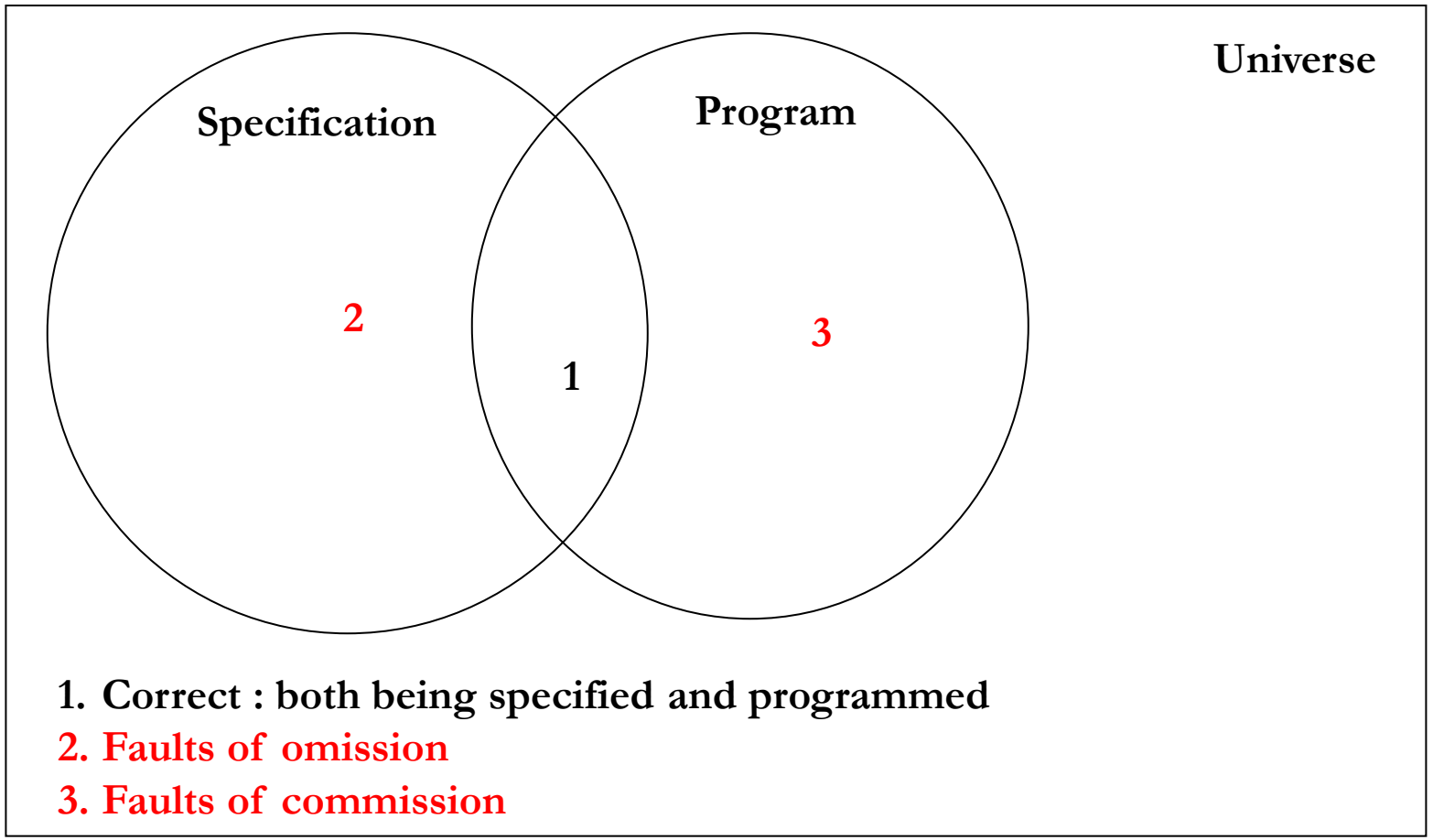
FUNCTION: System (Input) = Output

Structural (white box) testing



The structure of the code is "visible"
Targets the control flow within a Unit of code

Specification vs. Program



1. **Correct : both being specified and programmed**
2. **Faults of omission**
3. **Faults of commission**

White-box Testing: Pros and Cons

- **Pros:**
  - Easier to automate
  - Allows for more efficient debugging
  - Uses internals of program source to generate test cases
  - Able to test program source that has not been specified
  - Lends itself to the definition and use of test coverage metrics
- **Cons:**
  - Harder to use to validate requirements. White box tests typically focus on how something is implemented, not why it is implemented
  - Cannot detect faults of omission
  - Suffers from infeasible path problem
    - Control flow paths that cannot be exercised by any input data

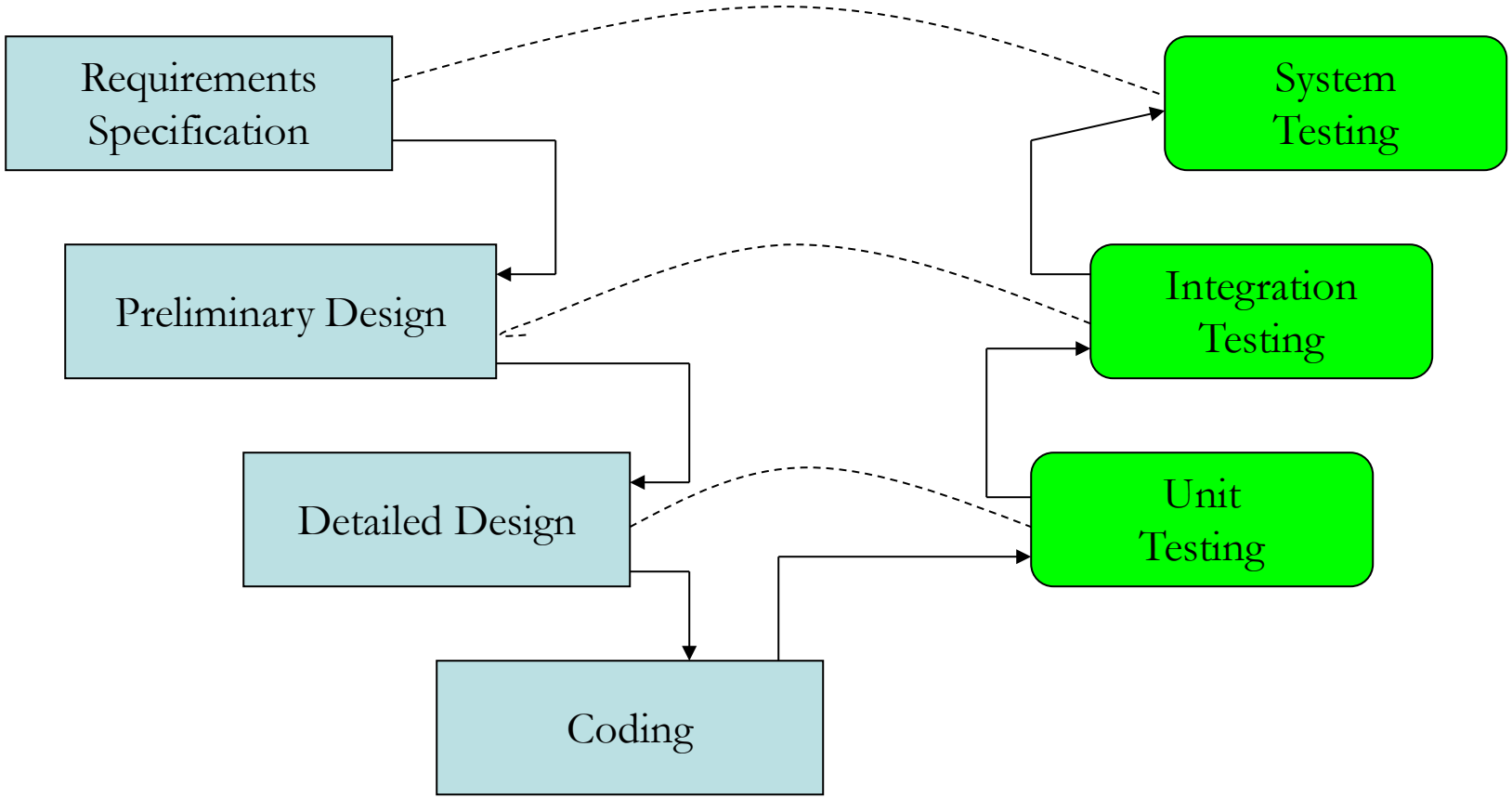
Black-box Testing: Pros and Cons

- **Pros:**
  - Functional test case independent of how the software is developed, so if the implementation changes, the test cases are still useful
  - Test case development can occur in parallel with the implementation, thereby reducing overall project development interval
- **Cons:**
  - Significant redundancies may exist among test cases. Without insight into the implementation, the same code paths can get tested repeatedly, while others are not tested at all
  - Compounded by the possibility of gaps of untested software

Black-box vs White-box Testing

- Considering program behaviors:
  - If all specified behaviors have not been implemented, **structural test cases** will *never* be able to recognize this
  - If the program implements behaviors that have not been specified, this will *never* be revealed **by functional test cases** (A virus is a good example of such unspecified behavior)
- When functional test cases are executed in combination with structural test coverage, both of two problems (redundancies and gaps) can be recognized and resolved.
  - This is also called "grey-box testing"

Waterfall Model and Levels of Testing



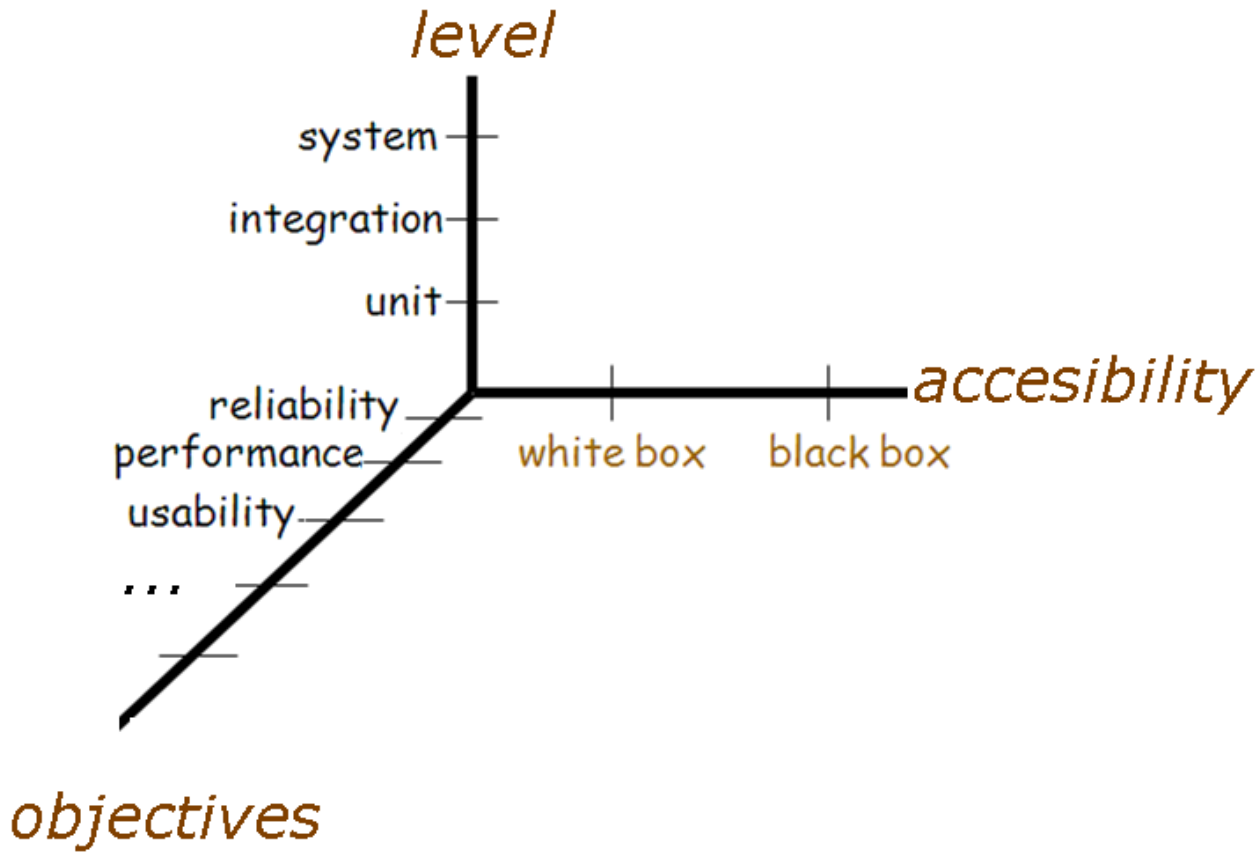
Testing levels

- **Unit testing**
  - Testing of individual components
- **Integration testing**
  - Testing to expose problems arising from the combination of components
- **System testing**
  - Testing the complete system prior to delivery
- **Acceptance testing**
  - Testing by users to check that the system satisfies requirements.
  - ***Alpha*** testing (in-house) and ***beta*** testing (at the customer site)

More types of testing

- performance testing, interoperability testing
- regression testing, **reliability testing, usability testing**,
- portability testing, **security testing**, compliance testing,
- …

Dimensions of Testing



Testing: Stakeholders

Software customer. The party or department that contracts for the software to be developed.

Software user. The individual or group that will use the software once it is placed into production. (Note: This may be the customer or it may be parties other than the customer.)

Software developer. The individual or group that receives or assists in writing requirements, designing the software, building the software, and changing and maintaining the software as needed.

Software tester. The individual or group that performs the check function on the software. (Note: These may be a subset of the developers, an independent group, or a combination of the two.)

Information technology management. The individual or group with responsibility for fulfilling the information technology mission. Testing supports fulfilling that mission.

Testing vs Software Quality Assurance (SQA)

- **Testing:** product-oriented
- **SQA**: process-oriented (improving the process leads to an improvement of the resulting product)
  - Prevent errors and faults in the first place