Software Testing – General Concepts

Andrea Polini

Advanced Topics on Software Engineering – Software Testing
MSc in Computer Science
University of Camerino

ToC

ToC

WARNING

Slides are distributed to help students in their preparation to the exam. In **no way** they intend to substitute text books. Instead a **thorough study of the text books** constitutes the **most wise strategy** to maximize the chances to pass the final exam.

ToC

Course and Teachers

- Advanced Topics on Software Engineering – Software Testing
  - Lessons:
    - Tuesday from 2pm to 4pm
    - Thursday from 11am to 1pm
  - web: http://didattica.cs.unicam.it/...
- Andrea Polini, Barbara Re
  - e-mail: andrea.polini@unicam.it, barbara.re@unicam.it
  - weekly office hours: Thursday 2pm-3pm
- Exam dates:
  - June 18th, 2020 - 11am-1pm, room AB1
  - July 16th, 2020 - 11am-1pm, room AB1
  - September 8th and 29th, 2020 - 11am-1pm, room AB1
  - February 2nd and 23th, 2021 - 11am-1pm, room AB1

Course Objectives

- The course permits to the student to acquire the knowledge needed to understand software testing issues and solutions. The course then aims at permitting the development of competences needed to operate in real scenarios in order to test complex software systems.

Study material

- **Reference book**:

  Aditya P. Mathur
  Foundations of Software Testing, 2nd Ed.
  Pearson, 2014.

- **Further references provided by the teachers**

ToC

Testing intuition

Software testing concerns the execution of some “experiments” in a controlled environment in order to acquire enough confidence on the behaviour of a software system when deployed in the real environment. Software Testing can equally aim at assessing functional properties and extra-functional properties (some of them at least)

Two different objectives and “moods”:

- Try to demonstrate that the system correctly satisfy the specifications, and the needs of users and customers
- Try to discover bugs in the code

Testing can never guarantee the absence of fault but just their existence

E.W. Dijkstra

Testing intuition

Software testing concerns the execution of some "experiments" in a controlled environment in order to acquire enough confidence on the behaviour of a software system when deployed in the real environment. Software Testing can equally aim at assessing functional properties and extra-functional properties (some of them at least)

Two different objectives and "moods":

- Try to demonstrate that the system correctly satisfy the specifications, and the needs of users and customers
- Try to discover bugs in the code

Testing can never guarantee the absence of fault but just their existence

E.W. Dijkstra

Testing vs. Debugging

Clearly strictly related but different objectives!

- Testing: finds bugs and shows possible divergences between what is observed, and what it is expected
- Debugging: removes bugs, and alineates the characteristics of the system to what is is expected

Testing vs. Debugging

Clearly strictly related but different objectives!

- Testing: finds bugs and shows possible divergences between what is observed, and what it is expected
- Debugging: removes bugs, and alineates the characteristics of the system to what is is expected

Testing and Verification

Verification is the general activity in SE the aims at assessing the relation among different artefacts in the development process.

- Static strategies: static analysis, model checking, code inspection, ...
- Dynamic strategies: software testing, software monitoring, ...

Testing vs. Formal Verification

Formal verification aims at proving the correctness of artefacts by showing that they satisfy specific properties. Relevant properties generally to be checked:

- something bad will never occur (safety)
- something good will eventually occur (liveness)

Testing and Verification

Verification is the general activity in SE the aims at assessing the relation among different artefacts in the development process.

- Static strategies: static analysis, model checking, code inspection, ...
- Dynamic strategies: software testing, software monitoring, ...

Testing vs. Formal Verification

Formal verification aims at proving the correctness of artefacts by showing that they satisfy specific properties. Relevant properties generally to be checked:

- something bad will never occur (safety)
- something good will eventually occur (liveness)

Genesis of failures

Error: *an error occurs in the process of writing a program (or document)*

Fault – aka bug, defect: *a fault is the manifestation of one or more errors, and is constituted by a piece of code that do not correspond to what is actually needed*

Failure: *a failure is the observation of a behaviour that does not correspond to the desired one*

Genesis of failures

Error: *an error occurs in the process of writing a program (or document)*

Fault – aka bug, defect: *a fault is the manifestation of one or more errors, and is constituted by a piece of code that do not correspond to what is actually needed*

Failure: *a failure is the observation of a behaviour that does not correspond to the desired one*

Genesis of failures

Error: *an error occurs in the process of writing a program (or document)*

Fault – aka bug, defect: *a fault is the manifestation of one or more errors, and is constituted by a piece of code that do not correspond to what is actually needed*

Failure: *a failure is the observation of a behaviour that does not correspond to the desired one*

ToC

SE and Software Qualities

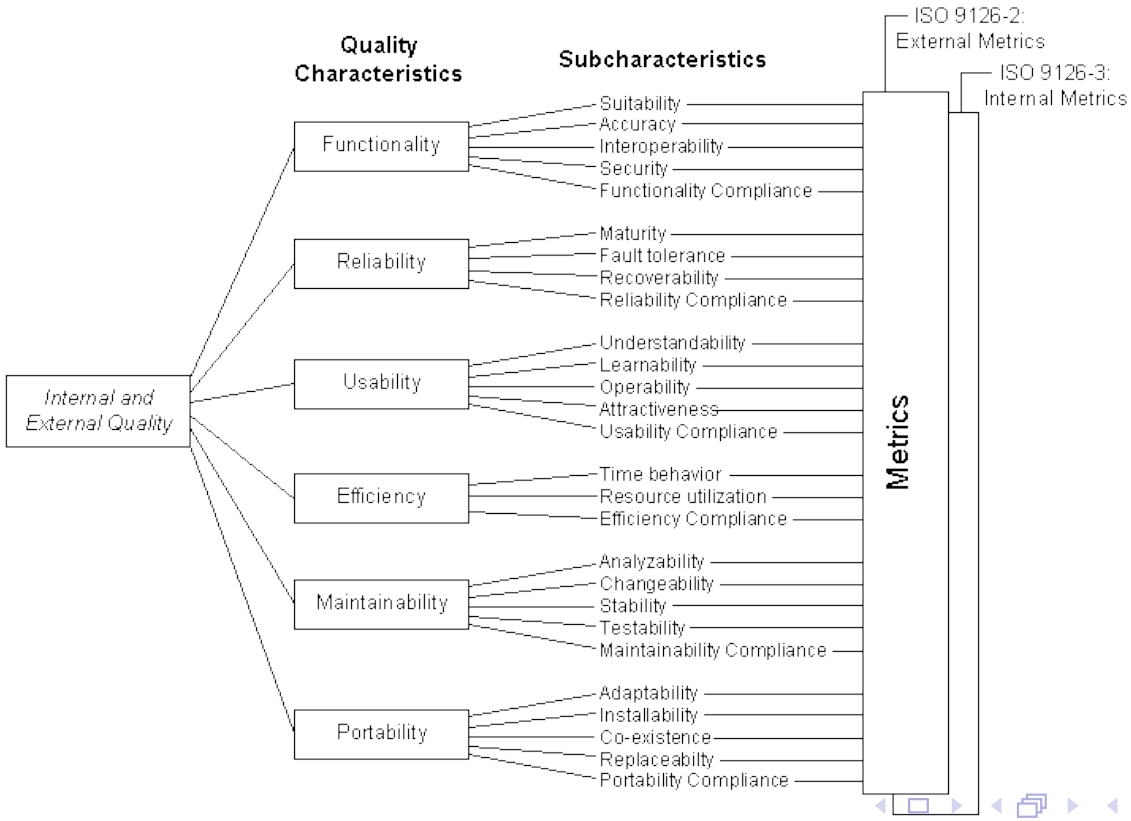
Software Engineering provides you with methodologies, techniques, approaches, and tools to build GOOD software

What does good stand for?

SE and Software Qualities

Software Engineering provides you with methodologies, techniques, approaches, and tools to build GOOD software

What does good stand for?



Software Quality

Software Quality

Degree to which a software . . .

1. conforms to specified requirements (functional and non-functional)
2. meets the needs and expectations of customers, users and stakeholders in general
3. is designed and developed according to sound engineering practices and standards

Metrics and Measures

To assess the degree to which a software conforms to a quality we need to define metrics and measurements procedures. A metric define a correspondence between entity and attributes of the real world with mathematical models and sets in order to better understand the real world itself. In order to make comparisons we need to consider sets with ordering relations.

Measure what is measurable, and make measurable what is not so

(Galileo Galilei)

Quality Dimensions

Quality attributes can be classified according to other several dimensions (internal/external):

- Static
  - understandability
  - maintainability
  - structuredness
- Dynamic
  - reliability
  - correctness
  - completeness
  - consistency
  - usability
  - performance

Once metrics have been defined for a given quality requirements will have to declare measures to satisfy

Correctness

Correctness

A program is considered correct if it behaves as expected on each element of its input domain

Correctness is just and ideal property, it asks for exhaustive testing, therefore it is more important to have a perception of how likely is that a software system will fail

Ex: The system should provide aritmetic operations for Natural numbers

```
public class math1{
    public double sum(double x, double y)
    { return x+y; }

    public double subtract(double x, double y)
    { return x-y;}

    public double abs(double x) {
        if (x>0) {return x;}
        else {return x;}
    }
```

```
public class math2{
    public int sum(int x, int y)
    { return x+y; }

    public int subtract(int x, int y)
    { return x-y; }

    public int abs(int x) {
        if (x>0) {return x;}
        else {return -x;}
    }
```

Reliability

ANSI/IEEE STD 729-1983: Reliability

Software reliability is the probability of failure free operation of software over a given time interval and under given conditions

- considers an operational profile

Reliability

Software reliability is the probability of failure free operation of software in its intended environment

Reliability

ANSI/IEEE STD 729-1983: Reliability

Software reliability is the probability of failure free operation of software over a given time interval and under given conditions

- considers an operational profile

Reliability

Software reliability is the probability of failure free operation of software in its intended environment

Operational profile

sort

Consider a `sort` program able to order input sequences both of strings and numbers (obviously not mixed).

Operational profile

An OP is a numerical description of how a program is used

- Different operational profile can be defined for `sort`

Requirements and testers

REQ 1

It is required to write a program that takes in input two integers and provides in output the maximum of the two

REQ 2

It is required to write a program that takes in input a sequence of integers and provide in output the sorted version of this sequence

Definition of tests can help in clarifying requirements. Incompletness of requirements can lead to ineffective testing activites.

Robustness

Input domains should be covered to include valid and invalid inputs

Requirements and testers

REQ 1

It is required to write a program that takes in input two integers and provides in output the maximum of the two

REQ 2

It is required to write a program that takes in input a sequence of integers and provide in output the sorted version of this sequence

Definition of tests can help in clarifying requirements. Incompletness of requirements can lead to ineffective testing activites.

Robustness

Input domains should be covered to include valid and invalid inputs

Testability

Testability

Degree to which a system or component facilitates the establishment of test criteria, and the performance of tests, to determine whether those criteria have been met
Related aspects are controllability and observability

Depending on how you measure testability it can be classified as static or dynamic

ToC

Some testing taxonomy

Test case

A test case is a pair consisting of test data to be provided in input and exepected as output

Test set (aka test suite)

A test set is a collection of zero or more test cases, generally homogeneous in terms of the functionality they stress

Test plan

A test plan is the definition of test requirements to be satisfied by the selection of test sets

Test Plan

The following checks must be carried on to test the `sortAD` program (where the A/D stays for Ascending/Descending and the program takes in input A or D to define which behaviour to perform)

- Execute the program on at least two input sequences, one with "A" and the other with "D"
- Execute the program on an empty input sequence
- Test the program for robustness against erroneous inputs such as "R" typed in as the request character
- All failures of the test program should be recordered in a suitable file using an appropriate form

Minimal test set? Which test set is the best?
Is a given test set adequate?

Test Plan

The following checks must be carried on to test the `sortAD` program (where the A/D stays for Ascending/Descending and the program takes in input A or D to define which behaviour to perform)

- Execute the program on at least two input sequences, one with "A" and the other with "D"
- Execute the program on an empty input sequence
- Test the program for robustness against erroneous inputs such as "R" typed in as the request character
- All failures of the test program should be recordered in a suitable file using an appropriate form

Minimal test set? Which test set is the best?
Is a given test set adequate?

Test execution

Test Execution

Test execution is the activity of performing the selected test case

At a first glance can seem an easy activity . . .

- load tests
- bring the system in the right status for test execution
- record results
- check results

Test Harness

A test harness is a tool that helps the tester in performing one or more testing execution activities

- setup
- reset
- test execution
- test reporting

Test execution

Test Execution

Test execution is the activity of performing the selected test case

At a first glance can seem an easy activity . . .

- load tests
- bring the system in the right status for test execution
- record results
- check results

Test Harness

A test harness is a tool that helps the tester in performing one or more testing execution activities

- setup
- reset
- test execution
- test reporting

Test execution

Test Execution

Test execution is the activity of performing the selected test case

At a first glance can seem an easy activity . . .

- load tests
- bring the system in the right status for test execution
- record results
- check results

Test Harness

A test harness is a tool that helps the tester in performing one or more testing execution activities

- setup
- reset
- test execution
- test reporting

The oracle problem

How can we assess the results provided by the system under test (SUT)?

This is the famous oracle problem

The oracle problem

How can we assess the results provided by the system under test (SUT)?

This is the famous oracle problem

The Oracle

Manually defined oracles:

- \- Costly
- \- Error Prone
- \+ More precise conditions

Automatically derived oracles:

- \- Difficult to implement
- \- Necessary conditions are generally checked (more false negative)
- \+ More reliable
- \+ Cheap

Logs of previous system usage can help in deriving oracles

The Oracle

Manually defined oracles:

- \- Costly
- \- Error Prone
- \+ More precise conditions

Automatically derived oracles:

- \- Difficult to implement
- \- Necessary conditions are generally checked (more false negative)
- \+ More reliable
- \+ Cheap

Logs of previous system usage can help in deriving oracles

Soundness vs. Completness

- A test set is sound if all discovered faults are actually faults in the system (no false positive)
- A test set is complete if it can discover all faults but can generate false positive

Which kind of test set would you like to define?

Soundness vs. Completness

- A test set is sound if all discovered faults are actually faults in the system (no false positive)
- A test set is complete if it can discover all faults but can generate false positive

Which kind of test set would you like to define?

ToC

Test generation

Test generation

Test generation deals with the definition of strategies for the selection of appropriate data input and invocation sequences in order to form test sets satisfying given properties

Strategies can be defined for:

- Requirements
- FSM
- Statecharts
- PN
- Timed I/O Automata
- Algebraic and logic specifications
- Code (generally using monitored run-time data)

Type of testing

What types of testing do you know and apply in your organization?

Testing can be classified in many different dimensions in some case orthogonal with respect to each other. A classification framework helps in clarifying concepts:

- Source of test generation
- Lyfe cycle phase in which testing takes place
- Goal of a specific testing activity
- Characteristics of the artefact under test
- Test process

Source of test generation

- Requirements > Black-box testing
  - Ad-hoc testing, Boundary value analysis, Partition testing, Predicate testing, Random testing, Equivalence testing, ...
- Code > White box testing
  - Mutation testing, Coverage testing, Data flow testing, Symbolic/Concolic testing, ...
- Formal model > Model based testing (BB special case)
  - FSM testing, Pairwise testing, Syntax testing, Conformance testing, ...
- Component interface > Interface testing (BB special case)
  - Interface mutation, Pairwise testing, ...

Lyfe cycle phase in which testing takes place

- Verification: are we building the product right
- Validation: are we building the right product

In the software production life-cycle different test are carried on with different objectives:

- Coding > Unit Testing
- Integration > Integration Testing
- System integration > System Testing
- Maintenance > Regression testing
- Pre-release > Beta testing, Acceptance testing

Goal of a specific testing activity

Goal oriented testing aims at showing specific properties for the system and then intends to show specific failures of the system

- Advertised features > Functional
- Invalid inputs > Robustness
- Vulnerabilities > Vulnerability
- Security > Security
- Errors in GUI > GUI
- System performance > performance testing, Stress testing, Load testing
- Customer Acceptability > Acceptance
- Peripherals compatibility > Compatibility

Characteristics of the artefact under test

The focus here is on the characteristics of the artefact that is under test. The specific characteristics of the technology/paradigm represent an important aspect of the testing strategy:

- OO testing
- Real-time testing
- Software testing
- Web service testing
- ...

Test process

In this case the focus is on the development process model and its relation to testing activities:

- Testing in the waterfall model
- Testing in the V-Model
- Spiral testing
- Agile testing
  - testing through the whole process, customer involvement, collaborative tester/developers, test often
- Test driven development
- ...

How can we assess a testing campaign?

Testing any artefact would require to specify:

- Test generation methods – number of test generated, number of test run, number of tests failed and number passed
- Test adequacy criteria used – results of test assessement states in quantitative terms
- Test enhancement – number of additional tests generated based on the outcome of the applied adequacy assessment, number of additional test run, number of additional failures discovered

How do we compare different strategies?

The saturation effect

Confidence vs. Reliability

- Confidence is a subjective assessment of the quality of the software with respect to its “correctness”
- Reliability should be an objective assessment of the quality of the software with respect to its “correctness”

The saturation effect warns testers on the efficacy of test generation strategies and suggests to apply more than one strategy

Testing principles

1. Testing is the activity of assessing how well a program behaves in relation to its expected behaviour
2. Testing may increase one's confidence in the corectness of a program through this may not match with program's reliability
3. A test case that test untested portions of a program enhances or diminishes one's confidence on whether the test passes or fails
4. Statistical measurements of the reliability of an application and code coverage are orthogonal
5. Code coverage is a reliable metric fo the quality of a test suite
6. Test derived manually from requirements alone are rarely complete
7. Random testing may or may not outperform non random testing
8. Saturation effect is real and can be used as an effective tool for improvement
9. Test automation aids in reducing the cost of testing and making it more reliable, there could be exceptions
10. Integrating metrics into the entire test process aids in process improvement
11. Every developer is also a tester