

Testing, Debugging, and Verification

Testing, Part I

Wolfgang Ahrendt, Vladimir Klebanov, Moa Johansson

30 October 2012

Let's review the course contents

- ▶ Testing (lab)
- ▶ Debugging
- ▶ Formal Specification (lab)
- ▶ Test Case Generation
- ▶ Formal Program Verification (lab)

Some practical things

- ▶ Please sign up for the google group (follow News link)
- ▶ Testing block consists of
 - ▶ 3 lectures (Today + Monday 5/11)
 - ▶ 1 exercise (Wednesday 7/11)
 - ▶ 1 lab (deadline 16/11)
- ▶ About labs in general
 - ▶ Use Fire system
 - ▶ Working in pairs is **mandatory**

Motivation for Course Unit on Testing

Ideas and techniques of testing have become essential knowledge for all software developers.

Expect to use the concepts presented here many times in your career.

Testing is not the only, but the primary method that industry uses to evaluate software under development.

Motivation for Course Unit on Testing (cont'd)

- ▶ The field of testing is **large**
- ▶ This course (unit) is rather small
- ▶ Does it make sense to get started even?

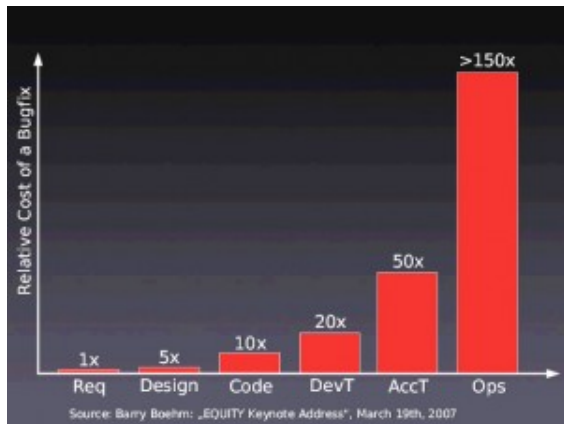
Cost of SW faults gets higher and higher. If the course will help to detect a few more faults in real applications, it was a good investment.

A few basic software testing concepts can be used to design tests for a large variety of software applications.

The testing techniques present in the literature have much more in common than is obvious at first glance. We highlight commonalities.

Motivation for Course Unit on Testing (cont'd)

Adequate software testing is **necessary** and **achievable**.



A Quiz

A simple program

Input

Read three integer values from the command line.
The three values represent the lengths of the sides of a triangle.

Output

Tells whether the triangle is

Scalene: no two sides are equal

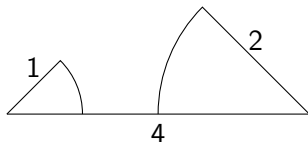
Isosceles: exactly two sides are equal

Equilateral: all sides are equal

Create a Set of at least 15 **Test Cases** for this Program

Solution — 1 Point for each Correct Answer

Q 1: An **invalid** triangle? e.g. (4,1,2)



Why not a valid triangle? (a,b,c) with $a > b + c$

Define valid triangles: $a \leq b + c$

Solution — 1 Point for each Correct Answer

Q 2: Some permutations of previous? e.g., (1,2,4), (2,1,4)

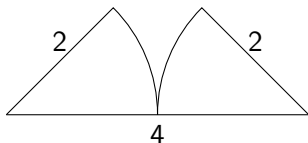
Fulfill above definition, but are still invalid.

Patch definition of valid triangles:

$$a \leq b + c \text{ and } b \leq a + c \text{ and } c \leq a + b$$

Solution — 1 Point for each Correct Answer

Q 3: An **invalid** triangle with **equal** sum? e.g., (4,2,2)



Fulfills above definition, but is invalid (depending on what we want!).

Patch definition of valid triangles:

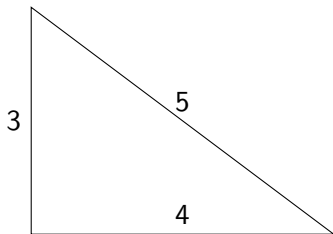
$$a < b + c \text{ and } b < a + c \text{ and } c < a + b$$

Solution — 1 Point for each Correct Answer

Q 4: Some permutations of previous? e.g., $(2,2,4)$, $(2,4,2)$

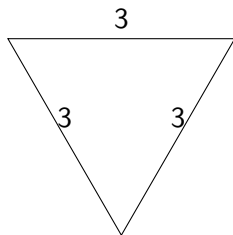
Solution — 1 Point for each Correct Answer

Q 5: A **valid scalene** triangle? e.g., (3,4,5)



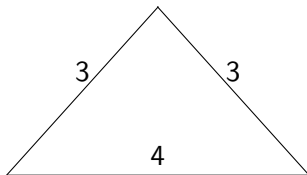
Solution — 1 Point for each Correct Answer

Q 6: An **equilateral** triangle? e.g., (3,3,3)



Solution — 1 Point for each Correct Answer

Q 7: A **valid isosceles** triangle? e.g., (3,4,3)



Solution — 1 Point for each Correct Answer

Q 8: All permutations of valid isosceles triangle?

$(3,4,3)$, $(3,3,4)$, $(4,3,3)$

Solution — 1 Point for each Correct Answer

Q 9: One side with **zero** value? e.g., (0,4,3)

Solution — 1 Point for each Correct Answer

Q 10: One side with **negative** value? e.g., $(-1,4,3)$

Solution — 1 Point for each Correct Answer

Q 11: All sides zero? e.g., (0,0,0)

Solution — 1 Point for each Correct Answer

Q 12: At least one value is non-integer? e.g., (1,3,2.5)

Solution — 1 Point for each Correct Answer

Q 13: wrong number of arguments, e.g., (2,4) or (1,2,3,3)

Solution — 1 Point for each Correct Answer

Q 14 (the most important one):

Did you specify the expected output in each case?

About the Quiz

- ▶ Q 1–13 correspond to failures that have actually occurred in implementations of the program
- ▶ How many questions did you answer?
< 5? 5 – 7? 8 – 10? > 10? All?
- ▶ Highly qualified, experienced programmers score **7.8** on average

First Conclusions

- ▶ Finding good and sufficiently many test cases is difficult
- ▶ Even a good set of test cases cannot exclude **all** failures
- ▶ Without a specification, it is not clear even what a failure **is**

The discipline of Testing is all about Test Cases

well, almost ...

Remark: At Ericsson, ca. 35% of code is test cases!

Brainstorming

brainstorm about:

- ▶ What is the purpose of testing?

...

Test Process Maturity Level in an Organisation

(adapted from [Beizer] and [AmmannOffutt])

- Level 0** There is no difference between testing and debugging.
- Level 1** Purpose of testing: show correctness.
- Level 2** Purpose of testing: show that the software does not work.
- Level 3** Purpose of testing: reduce the risk of using the software.
- Level 4** Testing is a mental discipline helping IT professionals to develop higher quality software.

Level 0 Thinking

Testing is the **same as debugging**

- ▶ Does *not* distinguish between incorrect **behaviour** and defects in the program
- ▶ Does not help develop software that is **reliable** or **safe**

Level 1 Thinking

Purpose: showing **correctness**

- ▶ Correctness is (almost) **impossible** to achieve
- ▶ Danger: you are subconsciously steered towards tests likely to **not** fail the program.
- ▶ What do we know if **no failures**?
good software? or bad tests?
- ▶ **Test engineers** have:
 - ▶ no strict goal
 - ▶ no real stopping rule
 - ▶ no formal test technique

Level 2 Thinking

Purpose: showing failures

- ▶ Looking for failures is a **negative** activity
- ▶ Puts testers and developers into an **adversarial** relationship
- ▶ What if there are **no failures**?

This describes most software companies.

Level 3 Thinking

Purpose: **reduce risk**

- ▶ Whenever we use software, we incur some **risk**
- ▶ Risk may be **small** and consequences **unimportant**
- ▶ Risk may be **great** and the consequences **catastrophic**
- ▶ Testers and developers work **together** to reduce risk

This describes a few “enlightened” software companies.

Level 4 Thinking

A mental discipline that **increases quality**

- ▶ Testing only **one** way to increase quality
- ▶ Test engineers can become **technical leaders** of the project
- ▶ Primary responsibility to **measure** and **improve** software quality
- ▶ Their expertise should **help developers**
- ▶ Purpose of testing: **improve ability of developers** to produce high quality software (compare w. spell checker)

This is the way “traditional” engineering works

Don't Pretend

Neither you nor me should pretend to have arrived at Level 4.

Test engineer: IT professional in charge of **test activities**, including:

- ▶ designing test inputs
- ▶ running tests
- ▶ analysing results
- ▶ reporting results to developers and managers
- ▶ **automating any of the above**

Developer = or $\neq$ Test Engineer?

Should testing be done by the developers (of the same software)?

different takes on this:

Contra: Test Principles in [Myers]

Principle 2: Programmer should avoid testing his/her own program.
(misunderstanding of specs carry over to testing)

Principle 3: A programming organisation should not test its own programs.

Pro: Extreme Testing (ET) [BeckGamma]

Principle: Developers **create tests first**

Principle: Developers **re-run test** on all **incremental** changes

Tool: **JUnit** designed for ET