Software Quality Assurance and Tests; Software Reviews

Jochen Ludewig, Universität Stuttgart

This is a (draft) English summary of my slides written in German. Please note that they are far from complete.

This lecture is based on our book "Software Engineering" (by Ludewig and Lichter, dpunkt, Heidelberg, 2007).

What is Quality Assurance?

Industrial Quality Assurance was created some one hundred years ago, when the car industry in the US experienced poor quality. Products are compared to an ideal product, which is either real or represented by a precise description, e.g. a drawing.

For software, however, this approach cannot be used, because we do not have the ideal product. Creating this product is exactly what software engineering is all about. Therefore, the experience from industrial quality assurance cannot be used for software quality assurance.

13.1 Software Quality Assurance (SQA)

software quality assurance — See: quality assurance.

quality assurance — (1) A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.

(2) A set of activities designed to evaluate the process by which products are developed or manufactured.

IEEE Std 610.12 (1990)

The second meaning is no longer used, because process assessment became a separate area.

We can improve our confidence in a piece of software

- by improving the software or
- by proving its high quality.

Both aspects are contained in what is called SQA.

SQA can be divided in three areas:

(a) Organizational SAA (project planning and project management, risk management, organisation for SQA)

(b) Constructive SQA (applying all our knowledge, our techniques and tools, for achieving good quality)

(c) Analytical SQA (testing any results, at any level)
Note that testing does no improve the quality of the object being tested. Therefore, (a) and (b) are more important than (c).

13.2 Analytical SQA

Note: While in German, "Test" usually means executing a program on a

computer, I use the English word "test" for any technique of checking a software unit for its properties, in particular for its correctness.

Tests show general and particular deficiencies of the software under test (SUT).

Tests do not improve the SUT directly; but the developers, expecting that their results will be subject to test, deliver better quality.

13.3 Deficiencies, faults, etc. (skipped)

13.4 Software Engineering and Tests

Testing (like any other software engineering activity) is not a religious command, like "Though shall test!". We apply testing in order to minimize the total cost.

13.5 Test: an Overview

In any test, there are two tracks starting from the task description (usually a requirements specification):

- On one track, the software is implemented, and run on some data.
- On the other track, the task description is analyzed, and the expected result with the same data is concluded.

Finally, the results from both tracks are compared.

Note that:

1. The test cannot indicate any problems in those parts that are common to both tracks (e.g. in the task description).
2. If there is a conflict between the real result and the expected result, there is a problem. This problem is not necessarily in the implementation. It may as well be somewhere else.
3. A positive result in a test means there is a problem. If the problem turns out to be not in the SUT, it is false positive. False positive results cause some extra work, but are otherwise harmless. False negative are far worse: There is in fact a problem, but the test does not show it.
4. Testing should be separated from debugging. There are several strong reasons for this separation.
5. The person who developed the SUT is unable to test it. The tester ought to show that there is something wrong; the author wants to show that everything is right.

Dynamic Testing versus Inspection

Only code can be executed. All readable documents can be reviewed, but only code can be executed. Therefore, dynamic testing (by execution on a computer is a very limited technique, compared to reviews.

There are many ways for inspecting software; in this lecture, I will talk about a technique called **technical review**. It is not necessarily better than any other technique, but has been applied very often, both in industry and in university.

13.6 Reviews

There are many ways to organize a review. We describe a type of review called **technical review**: Several experts inspect a software unit. In a meeting, the problems are collected and documented.

The goal is to find errors, not to improve the SUT, which can be any readable part of software, e.g. a piece of specification, a code module, or a data module.

For the inspection, reference documents are required.

Roles

Moderator: chairs the review

Secretary: takes notes

Experts: colleagues, able to discuss the SUT

Author: the person (or one of the people) who has created the SUT.

Note that the **manager(s)** should not participate in the review!

Each expert is asked to check some special points (from a generic check list).

The review meeting takes two hours, possibly a third hour for informal discussions. The experts are informed some two weeks in advance, so they have time to check the SUT. After the review, the author has another fortnight for improving the SUT.

Rules for reviews

1. The **moderator** takes care of invitations, documents, room, etc. organisational task are
2. The meeting does not exceed **two hours**.
3. The moderator will **cancel the meeting** if the conditions are insufficient (e.g. experts do not show up or did not inspect the SUT).
4. Don't discuss the author; **only discuss the SUT**.
5. Do not try to save personnel by **mixing roles** (like: one of the experts takes the notes).
6. Do not discuss **matters of style** not covered in any of the reference documents.
7. Do not **solve the problems**, just find, and document them.
8. Each expert is given the **opportunity to contribute**.
9. The secretary has to wait until the experts have reached a **consensus**, which is then documented.
10. The findings are classified as
    - *critical errors*
    - *major errors*
    - *minor errors*
    - *O.K.* (no problems found).
11. The review team recommends to accept the SUT as it is, or after improvements according to the notes, or not at all.
12. Everybody signs the notes.