1. Introduction to Software Testing

Hans Schaefer
hans.schaefer@ieee.org
http://home.c2i.net/schaefer/testing.html

Basic facts

Contents

Why to test?

- Murphy's law.
  - Humans err. -> There are always errors.
- Errors may lead to negative consequences.
  - From irritation to death. They can cost money, time and image.
- Software is everywhere.
- Important to know the quality level.
- Testing shall assure that the risk for problems is known. (a concern for project owners)
- After testing, it is possible to correct defects -> quality improvement, risk reduction.
- Contractual or legal requirements, industrial standards.

Here is a link to an interesting article detailing some of the most famous software errors.

http://wired.com/news/technology/bugs/0,2924,69355,00.html?tw=wn_tophead_1

Examples

- Any examples from your software practice?
- Cost of such problems?
- Why was the problem not-found before?
- How could it have been found before?

A bug is something that bugs somebody.

James Bach

Cost of downtime

Per hour of order processing system

$275K (Intel)

$167K (Cisco)

$83K (Dell)

$28K (Amazon)

$3K (EBay)

Barry W Boehm et al., "The ROI of Software Dependability: The iDAVE Model", IEEE Software May/June 2004

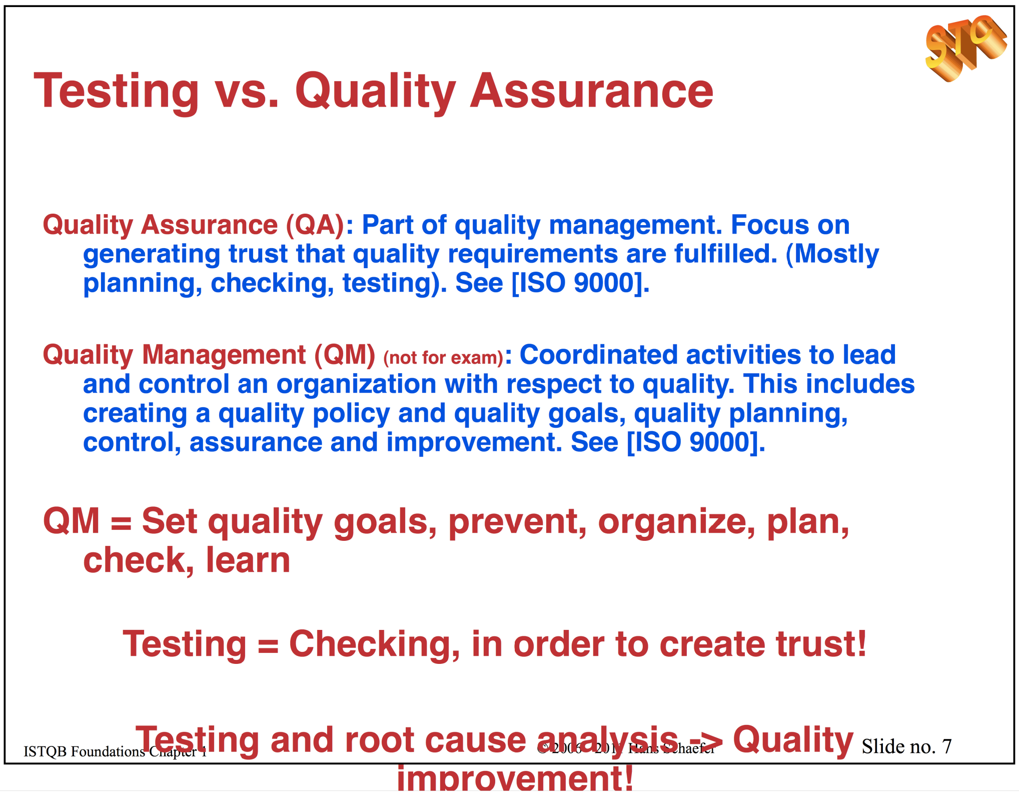
What is an error?

- **An error or mistake is a human oversight or a wrong decision (Root cause).**
- **Errors lead to defects (or faults or BUGS) in software.**
  (Fault = Manifestation of an error in software).
- **Defects, when executed, may lead to failures during run time.**
  (Failure = Deviation of software from expected results or service).

Testing can be concentrated on finding defects, or failures, or both.

Normally no follow-up of a failure until the <u>error</u> is known.

Be careful when doing error statistics!



Testing vs. Quality Assurance

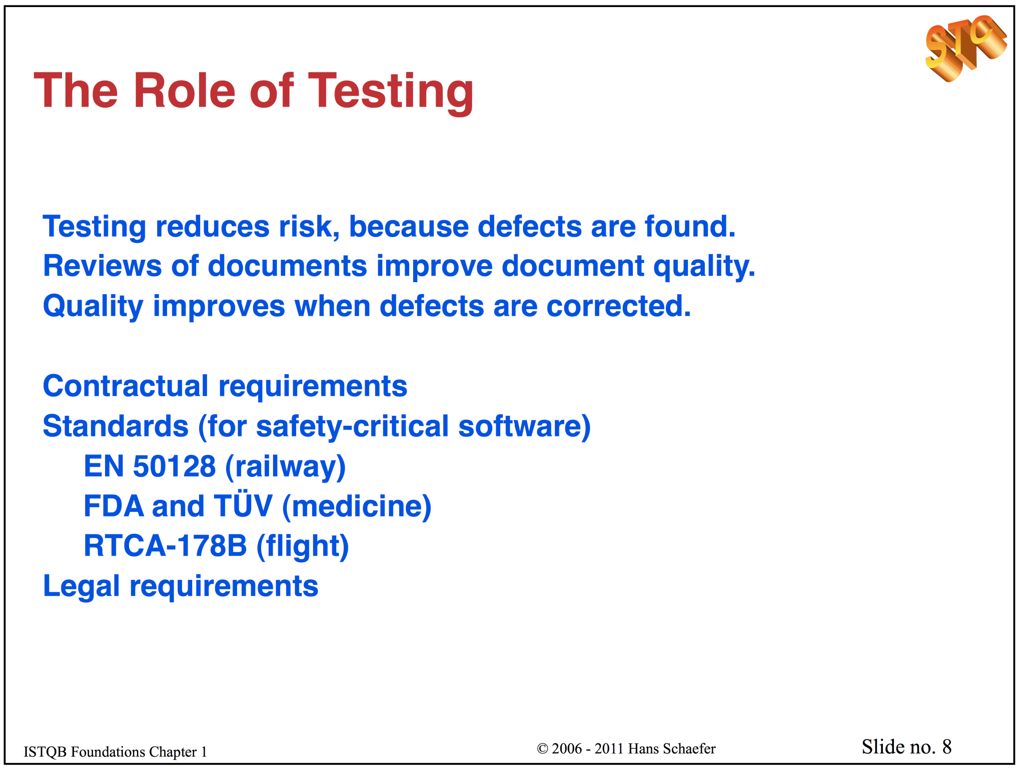
Quality Assurance (QA): Part of quality management. Focus on generating trust that quality requirements are fulfilled. (Mostly planning, checking, testing). See [ISO 9000].

Quality Management (QM) (not for exam): Coordinated activities to lead and control an organization with respect to quality. This includes creating a quality policy and quality goals, quality planning, control, assurance and improvement. See [ISO 9000].

QM = Set quality goals, prevent, organize, plan, check, learn

Testing = Checking, in order to create trust!

Testing and root cause analysis -> Quality improvement!



The Role of Testing

Testing reduces risk, because defects are found.
Reviews of documents improve document quality.
Quality improves when defects are corrected.

Contractual requirements
Standards (for safety-critical software)
- EN 50128 (railway)
- FDA and TÜV (medicine)
- RTCA-178B (flight)

Legal requirements

What Is Testing - Test Objectives?

Show that it works

Show that it does <u>not</u> work

Measure quality

Measure risk

Safeguard the receiver

Generate trust - confidence

Prevent faults

Learn to be a better tester

Kaner, Bach, Pettichord, in "Lessons Learned in Software Testing":
"Testers don't like to break things; they like to dispel the illusion that things work."

Tester motivation: www.testmanagement.com -> article "The Test Team Paradox"

Test and Quality

Test can increase quality ...... If defects are really corrected.
Test -> learn -> better work
Test can be required (certification for safety-critical systems).
Systematic test minimizes risk.
Test can measure quality (no. of defects found).
Test must be coordinated with other quality assurance.
Independent test may lead to people losing responsibility! („they test it anyway")
Testing is only part of the solution:
More = prevention, training, learning from errors

Learn from your mistakes!

Does Testing Contribute to Better Quality?

Yes, but...

- Errors in requirements can often not be corrected by testing.
- Some errors and misunderstandings are found, some not.

What Is Quality

The degree to which a component, system or process meets specified requirements and/or user/customer needs and expectations. [After IEEE 610]

The degree to which a set of inherent characteristics fulfills requirements [ISO 9000]

Product ability to fulfill specified and implied needs and requirements (older ISO 8402)

Value to some people who matter (Weinberg)

ISO 9126-1 quality characteristics:

– Functionality	Reliability
– Efficiency	Usability
– Maintainability	Portability

How Much Testing Is Enough?

Depends on risk
- Technical: How many defects, how new is the application?
- Business / Use: How ”dangerous”are the defects?
- Project: New solutions, methods, complexity, distributed?

Depends on restrictions
- Calendar time
- Budget, resources
- Project restrictions

Test is enough when the product owner has enough information to make a serious decision about releasing the product.

Risk-based Testing

You cannot test everything!

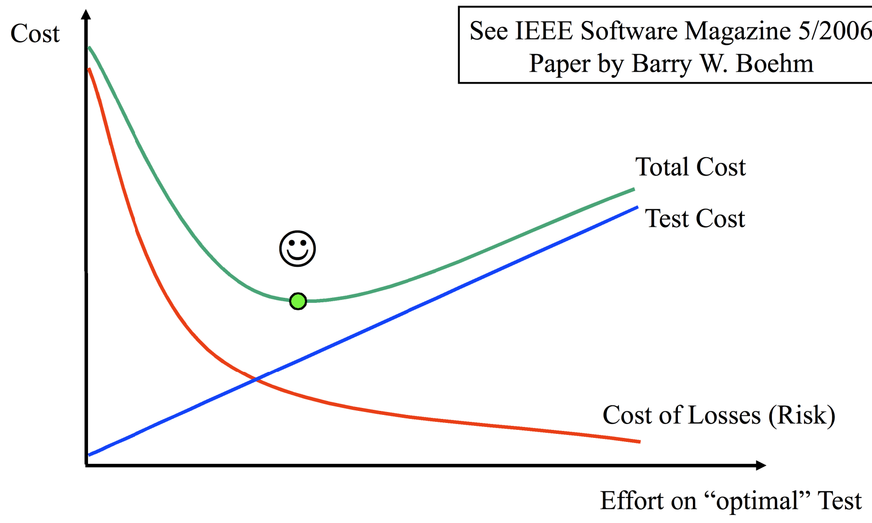
We test where the risk is:
- Most important (largest cost for the user / customer if problems)
- Probably worst (full of defects), most used

Every test plan must be prioritized by risk!
Standard test methods address risk of typical faults!

Details, see: http://home.c2i.net/schaefer/testing/risktest.doc
More in chapter 5

Balancing Risk and Test Investment

See IEEE Software Magazine 5/2006
Paper by Barry W. Boehm



The Final Goal

Save money

By finding defects when they are (still) cheap
By getting defects fixed
By preventing defects

Test effort depends on defect cost!
Test is financed by saving correction cost of defects later!

Testing is a <u>service</u>.

Questions?

Definition: What Is Testing?

Official definition (ISTQB):

Testing: The process consisting of all life cycle activities,
both static and dynamic,
concerned with planning, preparation and evaluation of software products and related work products
to determine that they satisfy specified requirements,
to demonstrate that they are fit for purpose and
to detect defects.

Practically: What Does Testing Mean?

To execute analyses in order to check software and documentation against requirements and expectations.

To execute programs and check if the result is as specified or as expected.

Both!

First we analyze what we develop, then we plan, prepare and execute test!

The Test Process

Plan

Prepare (Analyze, design, implement)

Execute

Analyze and make conclusions, report

Archive and learn from experience

Not only execute!

Differing Viewpoints About Testing

- Development testing (find bugs to fix)
- Acceptance testing (verification + validation against requirements, contract, check quality)
- Maintenance testing (verification that no new faults are built in)
- Operational testing (reliability and operational properties)

Testing And Debugging

Testing shall find problems or defects.

Debugging comes after a problem is found:

- Isolate the cause
- Fix the bug
- Install the fix

Testing

- Verify that the bug is fixed (re-test)
- Verify that there are no side-effects (regression test)

Who does / pays for what?

Normally, testers test and developers debug.

Fundamental Test Principles(1): Growing Defect Costs -> Early Test

Finding and fixing defects immediately is cheap, later is more expensive.

From phase to phase in a project, defect costs increase by a factor (typically between 2 and 10).

Exponential increase.

The same is true during test execution: Problems in higher level test cost more than in lower level test.

For customers, the cost can be enormous. (product recalls, train crash).

Testing shall contribute to finding bugs as early as possible!

Fundamental Test Principles(2): Complete / exhaustive Test Is Impossible

Software is very complex.

A program can fail at any combination of values or execution path.

Astronomical number of test cases.

Testing can only check a small part of a program!

Every ”thing” by itself.

risk-based testing!

Bill Hetzel, 1988, The Complete Guide to Software Testing

"The only **exhaustive testing** there is is so much **testing** that the tester is **exhausted**! "

Fundamental Test Principles(3): Test shows the presence of defects

Test cannot prove correctness.

Test shows that there are bugs.

Test reduces the probability for not-found bugs.

Testing without finding bugs increases the trust to the product, but is no proof for correctness!

Fundamental Test Principles(4): Defects are social / clustering

80% of the faults are in about 20% of the components (Pareto-distribution).

Or: Some faults are responsible for most failures in operation.

Defects are symptoms of deeper causes. Such causes may generate more defects.

Testing must be tailored to the defects found!

Fundamental Test Principles(5): Test is context dependent

A test cannot be copied blindly from system to system.

How much do the bugs cost?

When do we have to deliver?

Integration with other products?

safety-critical?

And more...

Don’t switch your brain off!

Testing must be planned individually for every product and project!

Fundamental Test Principles(6): Pesticide paradox

Re-executing existing test cases on later program versions finds fewer and fewer problems.

New problems found nearly only with new combinations or values.

Every test must be analyzed and optimized on a regular basis for effectiveness.

Cem Kaner, Jan 2008: What is the benefit of running the same tests over and over for the other areas? Repetition of exactly the same tests every time ensures that any bug missed the first time stays missed. If your notion of regression testing is retesting the same area of the product to check whether something was broken as a result of change, is it better to run exactly the same test or is it better to run variations that check the same issues but (a) with different data or (b) in different combinations with tests of other areas?

Do not trust a test blindly!

Fundamental Test Principles(7): No failures in test -> useful system? (”absence-of-errors-fallacy”)

One thing are functional defects.

Another thing is a product that is not useful.

Many types of ”stakeholders” - consider everyone.

Requirements and needs change over time.

Test does not find all requirements problems.

Test must cover verification and validation!

Verification: Is the product right?

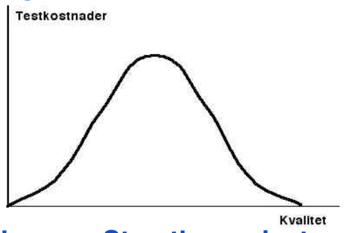
Validation: Is it the right product? (value)

Test Principle 8: Test cost depends on product quality before testing

(principle not from syllabus)

Test cost impossible to estimate reliably.

Depending on the ”restaurant curve”



Very low quality -> Test finds many fatal problems -> Stop the project.

Medium quality -> Test, isolate problems, fix problems, new test. Find new problems, isolate, fix, new test, ...

Very good quality -> Test, that’s it.

Questions?

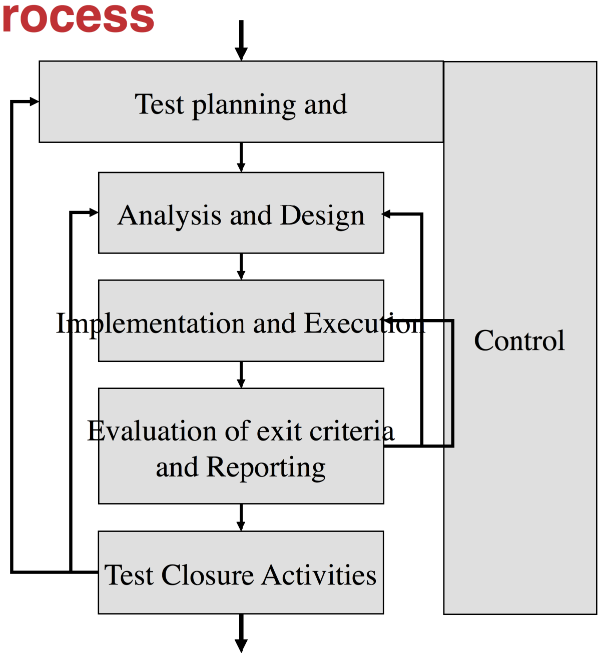
The Test Process

Execution most central

Before and after?

Testing requires a systematic process!

The Test Process


Test planning and
Analysis and Design
Implementation and Execution
Evaluation of exit criteria and Reporting
Test Closure Activities
Control

Test Processes According to ISO 29119 Part 2 (not for exam)

Organizational Test Process
(Test policy and strategy)

Test Management Processes
(Plan, monitor, control, complete)

Fundamental Test Processes
(Design, implementation, environment set-up, execute, incident reporting)

The test process

The next pages are a rough checklist.

More details in

IEEE Standard 1008 (test process)

IEEE Standard 1012 (verification and validation)

IEEE Standard 829 (test documentation)

1 - Test Planning and Control

Project plan for testing

- What do we want to achieve?
- How?
- Test plan

Control

- Throughout the project
- Monitoring, coordination, adaptation
- Management with respect to the plan
- Decisions based on information collected
- Reporting status and deviations from the plan

Planning tasks

Determine scope and risk
Determine the goal with testing and what to test more and less
General approach to testing (techniques, coverage, coordination, infrastructure, ...)
Resources (personnel, platforms, tools)
Tailoring of the test strategy or policy (master test plan)
Schedule
Start- and Exit criteria
Risk determination

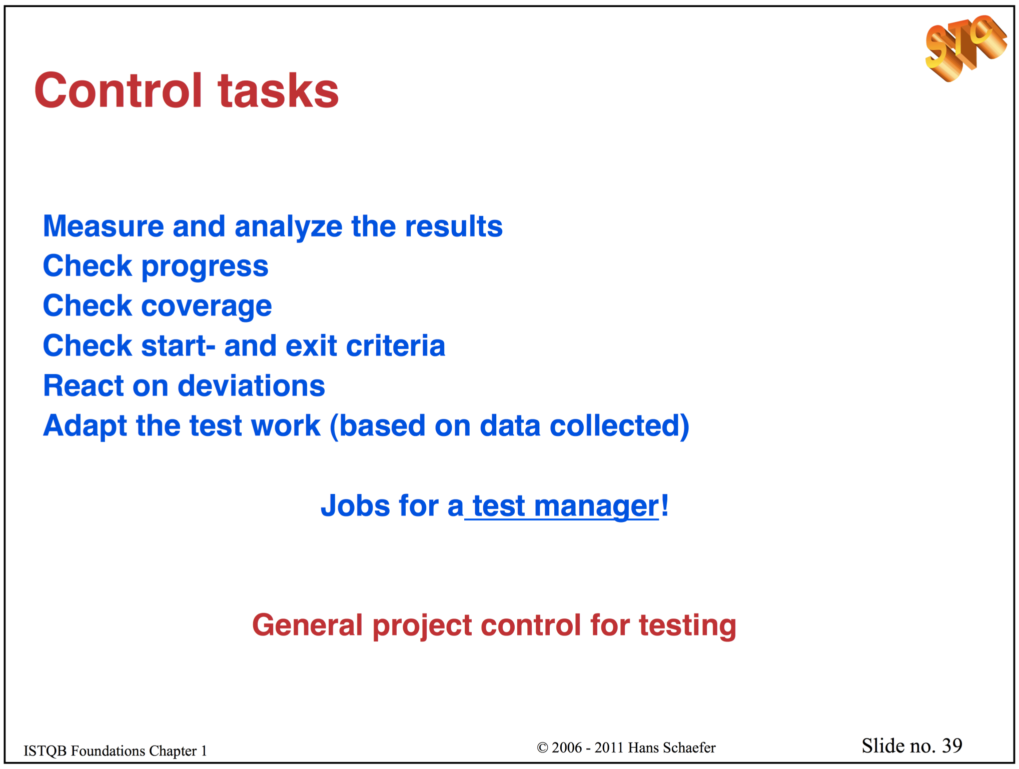
Jobs for a test manager!

IEEE Std 829 - test plan

Terminology
Organizational documents

Test policy: A high level document describing the principles, approach and major objectives of the organization regarding testing. SHORT!

Test strategy / handbook: A high-level description of the test levels to be performed and the testing within those levels for an organization or program (one or more projects).

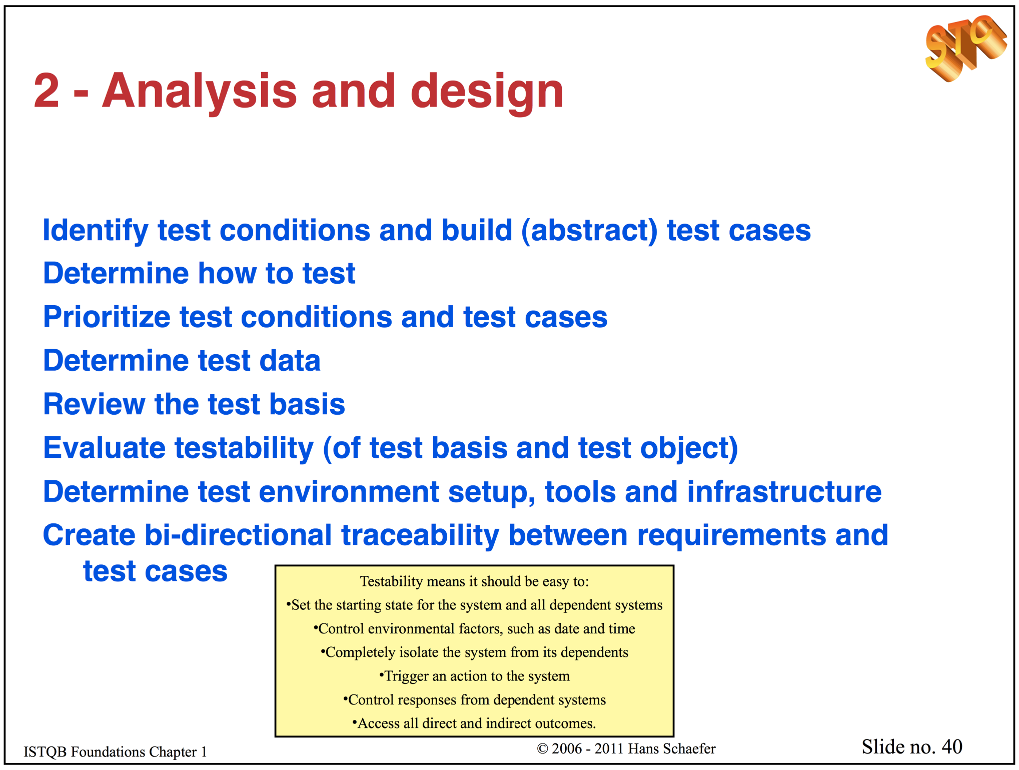


Control tasks

Measure and analyze the results
Check progress
Check coverage
Check start- and exit criteria
React on deviations
Adapt the test work (based on data collected)

Jobs for a test manager!

General project control for testing



2 - Analysis and design

Identify test conditions and build (abstract) test cases
Determine how to test
Prioritize test conditions and test cases
Determine test data
Review the test basis
Evaluate testability (of test basis and test object)
Determine test environment setup, tools and infrastructure
Create bi-directional traceability between requirements and test cases

Testability means it should be easy to:
•Set the starting state for the system and all dependent systems
•Control environmental factors, such as date and time
•Completely isolate the system from its dependents
•Trigger an action to the system
•Control responses from dependent systems
•Access all direct and indirect outcomes.

Analysis and design: Advice

Testability = How easy is something to test.

If you analyze early enough, you may require better testability. -> better design, ”test hooks”!

Require design- and programming standards!

Do NOT just accept what developers give you. Testability must be considered during design!

3 - Implementation and execution

Test data, environment, test lab, test tools

Build test data (concrete test cases)

Describe and prioritize test procedures / scripts (order of test cases and other info)

Develop test harnesses, programs, utilities etc. (”testware”)

Arrange test procedures into test suites

Verification of test material and environment

3 - Execution

Logging of versions (test object, test environment, test cases)
Executing the test object with test input (execute the test procedures)
Manual and/or automatic
Logging of results
Comparing actual and expected result
Reporting and analysis of problems and deviations
    Isolating, find the cause, extra search
Test after fault repair (re-test, regression test)

The execution phases

Pre-test / Smoke test:	
	Run a few test cases in order to see if the product is ready for testing
Main test:	Run all planned test cases.
After test:	Run extra test cases.
	Repeat the test after repair (regression test).

4 - Evaluating exit criteria and Reporting

Compare the run test with the plan.

Check completeness of the software and any open defects

Check exit criteria (for example test coverage)

- More test needed?
- Should the criteria be changed?

Summary - test report for the stakeholders

Test (summary) report = Recommendation to management about what to do with the product

5 - Test Closure Activities

Collect, archive, learn

Archive the test material, equipment and environment for later reuse (next release)

Test material to the maintenance crew

Document acceptance of the test object

Retrospective / Lessons learned:

What could we do better?

Can we improve the test process?

Can we improve development?

Questions?

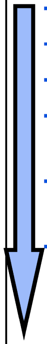
The Psychology of Testing

Goal Conflicts About Testing

You (as developer) are blind for your own mistakes.
You are constructive, testing is destructive.
You want to get finished. Testing delays you.
-> Someone else should test!
Learning takes time for someone else
-> Test yourself!
A good test cannot save a bad program.
-> A management problem!

It is very difficult to get people to understand facts, if their salary is dependent on their misunderstanding the facts.
Al Gore in his film, “An inconvenient truth”

Levels of Independence (can be at any test level)



Test done by developer.
Test done by developer after some time.
Test done by colleague (in the development group).
Test done by independent group (”Test group”) or by test specialists (performance test for example).
Test group can be in the project, in the development department or independent.
Test done by a person in another organization or company (outsourcing).

Special possibilities:
- Random test
- Use of standard test suites
- Use of tools to generate the test

Who Is Usually Responsible For Test Work? (not for exam)

Component test (Module test, Unit test): Developer self (maybe a colleague should help prepare).

Component Integration Test: Developer self (maybe a colleague to help).

Higher level-Integration Test: Testers and test specialists (from a special test group).

System test: Test group and support people, maybe with customer-like people (marketing). Typically an independent test group.

Acceptance test: Customer and test specialists. Third parties.

Why these?

A compromise between knowledge about details in the system, an independent view and special knowledge about testing

Diplomatic Skills

Defect reporting is a problem

Info about failures is not very welcome!

Author may take it as personal criticism.

Tester needs diplomatic ability.

Tester has a sales job.

Testers should make it easy to find the cause (isolating, repetition of the failure).

Diplomatic ability also important in reviews!

Details: http://home.c2i.net/schaefer/testing/testermidnighteng.pdf

Properties of a Good Tester

- Wants to know
- ”Professional pessimist”
- Critical
- Focused on details - and on the whole
- Understands software development
- Experience about where to expect problems (error guessing)
- Domain knowledge
- Good ability to communicate facts (not always good news) in a constructive way - diplomacy
- Can formulate objective and fact-based reports

Communication

Communication problems may occur, particularly if testers are seen only as messengers of unwanted news about defects. How to improve:

Collaboration rather than battles.

Remind of common goal of better quality systems.

Communicate findings on the product in a neutral, fact-focused way .

Don’t criticize the person who created it.

Empathy: Try to understand how the other person feels and why they react as they do.

Confirm that the other person has understood what you have said and vice versa.

Questions?

Information and literature

1 - Tilo Linz, Andreas Spillner, Hans Schaefer, Software Testing Foundations, dpunkt Verlag 2006, (German and Dutch editions available).

2 - C. Kaner, J. Falk, H. Q. Nguyen, "Testing Computer Software" (2nd ed.), John Wiley & Sons, 1999.

A good book for beginners, especially in projects under time and market pressure and consumer software production. Contains some good advice for system testing. Good list of possible defects. Very practical.

3 - Lee Copeland, "A Practitioner's Guide to Software Test Design", Artech House, 2004.
A good introductory test on test design methods. Easy to understand and many examples.

4 - Glenford Myers: "The Art of Software Testing", New York, John Wiley, 1979.
The classic book about software testing. Still, more than half of it is important knowledge. Some good advice for every test level, and the basic ideology.

5 - Bart Broekman, Edvin Notenbom, "Testing embedded software", Addison-Wesley 2003.

Test design and management for technical software.

6 - Martin Pol, Ruud Teunissen, Erik van Veenendal, Software Testing, A Guide to the Tmap Approach, Addison Wesley, 2001.

This is the standard way to organize testing in the Netherlands. A good guide for managing development of application software.

7 - IEEE Software Magazine, Something about testing, quality management or reviews in nearly every edition.

8 - Better Software Magazine, www.bettersoftware.com. www.stickyminds.com Very practical!

9 - Erik van Veenendal (ed.), The Testing Practitioner, UTN 2003, ISBN 90-72194-65-9

10 - Boris Beizer, "Black Box Testing", John Wiley and Sons, New York, 1995.

A book about basic methods for test data selection. Useful especially for unit and integration test, and the functional part of system testing.

11 - Rex Black, Critical Testing Processes, Addison-Wesley, 2003, Rex Black, Managing the Testing Process.