Software Testing

Advanced Level Syllabus

ISTQB™-Certified-Tester, Advanced Level

Version 1.2(E)

Date: 09/2003

English translation of the syllabus of the German-Testing-Board.
Only for information.
Meant for delivery to the ISTQB™.

List of Amendments

Version	Date	Remarks
1.0	05/2003	First work group draft for review by the GTB
1.1	06/2003	Revision based on reviews, to be reviewed by GTB
1.2	07/2003	Marginal changes, additions to the literature directory
1.2(E)	09/2003	English Version

Keynote

Informatics today permeates practically all facets of society such as Business, Social, Cultural, Leisure etc., and frequently plays a central role in the development of all kinds of processes. The result is an increasing dependency on software with the result that software and IT supported systems have to function error free and reliably under all operating conditions in spite of increasing complexity.

Therefore it is important that software and IT systems be systematically tested in order to prove and guarantee their reliability before use.

Test objects are IT systems in commercial and technical environments (e.g. Accounting, Production Planning, Machine or Vehicle Control). As well as the complete end product, the individual components of the IT system (subsystems, components, modules etc.) and the development documentation (requirements specification, design, user documentation etc.) must also be subjected to testing in the course of development.

Consequences for training

The associated inspection and test tasks are complex and require not only a good understanding of the test object in question, but also a detailed knowledge and command of the necessary inspection and testing techniques. This calls for well-trained, qualified specialists.

Aim of training as a Certified Tester

In the course of training as a certified tester, you are taught the appropriate knowledge and techniques that in practice lead to a systematic, structured process for inspection and testing and thereby contribute to the improvement of the quality of the software.

Training as a certified tester is aimed at all people involved in software testing, who wish to base their knowledge on a sound foundation or extend it. The training is suitable for quality and test professionals, programmers, developers, specialists and (project-) managers who are responsible for the execution, planning or control of testing.

- Certified testers are capable of designing and planning the project-specific inspections and tests. In order to do this they have to analyse the current status in particular, evaluate the risks to the business process and identify the objects to be tested.
- Certified testers are aware that within the framework of an IT project the test plan is influenced by the chosen architecture, the risk to the business process, the maturity of the technology and the capabilities of the development staff. They are capable of defining adequate test targets, of choosing test techniques (review and inspection techniques), of identifying the necessary test tasks and defining the timescales as well as selecting and preparing the resources.
- Certified testers put into practice the inspections and tests that accompany the software and IT system development processes. They execute the tests, create the corresponding test reports and forward fault reports to the developer.
- Certified testers are able to plan, organise, direct and take responsibility for document reviews (e.g. data model, requirements specifications or program code).
- Certified testers can evaluate how tests can be supported by tools and which test infrastructure should be used.
- Certified testers are aware that 100% testing is not possible. Test cases must be selected depending on the probability of faults and the associated risks. Which parts of the software are to be tested and how intensively must be evaluated during the test planning.

Notes on Curriculum and Examination

The curriculum defines:

- the examination-relevant material in its contents (keywords whose definition must be known are emphasised in **bold),**
- the minimum time in which the material must be taught in accredited training courses. The material is to be illustrated and taught using suitable examples and exercises.

The curriculum does not define:

- the (chapter-)sequence in which the material has to be taught,
- the timescales for examples and exercises in accredited training courses.

The examination is a multiple-choice test. Practical competence to use the material and the processes will be examined.

Subject	Description	Time (min.)
Introduction		**45**

Introduce the concept of the certified tester (see above), certified tester training modules, curriculum coherence, training, examination and certificate, associated institutes and international connections (examination/certification bodies, training providers and accreditation, national boards/ISTQB™).

1 Basics of Software Testing		**90**
1.1 Overview	Overview of the basics of software testing which are handled in „ISTQB™-Certified-Tester, Foundation Level"	

Introduce the central themes, philosophy and problem definitions of software testing as presented in the foundation course curriculum.

Introduce the "ISTQB™-Certified-Tester, Advanced Level" course.

1.2 Testing in the software lifecycle	Explain how testing is embedded in the various software development models: • sequential (waterfall model, V-model) • iterative (incremental and evolutionary development) • Rapid Application Development (RAD)

Explain that the testing process is no isolated process but is interconnected with other processes:

- Project management
- Configuration- and change management
- Software development
- Technical support

- Producing technical documentation

Point out how early **test planning** and the later **test execution** are related in the V-Model.

Explain the difference between **verification** and **validation.** Explain that verification and validation should take place in the early phases of the software life cycle and can be accomplished in the form of reviews.
Point out the role of change and **configuration management**.

Define the following test steps:

- **Component testing**
- **Integration testing**
- **System testing** (functional and non functional)
- **Acceptance testing**

Explain that in the project context, other test steps can also be defined.

Each test step has the following characteristics:

- **test goals**
- **test size**
- input and output criteria
- **test results** to be delivered
- **test technology** to be used
- measurements and metrics to be used
- **test tools** to be used
- **test standards** to be complied with

Explain that the development process influences the test process.

Explain the terms **retesting** and **regression testing**.
Retesting is the repeat execution of test cases which lead to a fault, with the aim of proving that the fault has been cured.
Regression testing is the renewed testing of an already tested program or part after modification, with the aim of proving that the **modifications** incorporated have not introduced new faults or exposed any masked faults.

2 Test process 150

2.1 Fundamental test process

Explain the position of the fundamental test process in the software development process.

2.2 Test planning

This topic is handled in detail in Chapter 3

2.3 Test specification

Explain that the choice of procedure for test specification depends on the following factors:

- the risks to be covered
- the degree of knowledge about the software to be tested
- the documentation available

List some procedures for test specification, e.g. as described in: Myers [Myers82] , Balzert ([Balzert 98], [Balzert 00]) oder BS 7925-2:1998 [BS 7925-2]

Explain the phases of a test specification:

- test case investigation
- test data creation
- test script creation

In test planning, the test objects (e.g. the functions to be tested) are identified. In test case investigation, the test cases for these test objects are created and described in fine detail. The test data substantiates the test cases. The test scripts make the test cases executable.

Criteria for prioritisation and risk evaluation, as set down in the risk analysis and the test outline, can be used for test object demarcation as well as for test case drafting. A recognised standard for the layout and content of test documentation is the norm IEEE 829 – 1998 [IEEE 829]
Explain that the test specification techniques can be used for all test steps.
In acceptance testing test cases can be extracted, e.g. from the analysis of the business process or use cases.

Explain which requirements of the **test case description** are posed in the **test specification.** Test case description is structured as follows:
Structure:

- **Test inputs**
- **Preconditions** (system condition, database condition)
- functions to be tested (aim of the test case)
- **final conditions** (expected results)
- **Outputs** (e.g. error messages)

Understanding of the specification of the software to be tested is required in order to be able to deduce the **expected result** of the function to be tested
This knowledge is often not formally specified, e.g. in the form of an expert draft. Because of this it is sometimes necessary to have access to alternative sources such as the expert know-how of specialists or developers.

An example of a development process where formally specified **requirements** are not needed is RAD (Rapid Application Development)

If the description of the expected results is missing from the test specification, the test can still be executed but cannot be correctly evaluated.

As well as the output, the final condition of the software under test and the test environment are part of the test results

- **outputs**
- **final conditions**

Explain the non-functional quality attributes according to ISO 9126/DIN 66272;

- **Reliability**
- **Efficiency** (consumption and time
- **Usability**
- **Changeability**
- **Transferability** (portability)

Plus attributes of lesser importance e.g.:

- **Instalability**
- **Interoperability**
- **Conformance**
- **Safety**
- **Recoverability**

These **non-functional quality attributes** should, however be documented and prioritised in the software specification.

Explain that **reviews**, **inspections** and **static analyses** are equallv suitable for discovering faults and thus for enhancing tests. Instead of the software, the objects to be tested here are documents, e.g.:

- Requirements specifications
- Draft specifications
- program source code

2.4 Test execution

2.4.1 Preparation for test execution

Explain the preconditions for test execution:

- **Test specification**, **test scenario** or **test script**
- Provisioning of the test environment (including rooms, equipment, personnel, hardware, software, tools, peripherals, communications equipment, user authorisations etc.)
- Naming those responsible for the creation and maintenance of the test environment and ensuring they are available.
- Setting up **configuration management** and **deviation management** and other support areas.
- Verification of the test environment (in working order, complete, correctly reports good and error conditions)

2.4.2 Execution of tests

Explain that in order to guarantee the verifiability of tests and the repeatability of tests during **regression testing** and/or **re-testing**, it is necessary to adhere to the **test script.**

There should be a formal way that allows the tester to suggest and execute further tests, which are not documented in the **test scripts.**

It is sometimes necessary to familiarise the representative of the requesting department or client in the test procedures in order to raise their confidence in the testing.

2.5 Test logging

2.5.1 Test scrutiny

Explain that every deviation, independent of the cause, between expected and actual result must be logged and investigated. Possible causes could be e.g.:

- Faults in the software
- Faults in the test data
- Faults in the test environment
- Faults in the requirements specification
- etc.

It is helpful when analysing the cause of the fault, to log the actual inputs, so that activities carried out by the tester are traceable. The log does not necessarily have to be automated, it can be documented by hand written notes or a hardcopy.

2.5.2 Test outcome documentation

Explain that each and every execution of a test must be recorded for the purposes of examination (e.g. through audits) and traceability. The log entries serve as a record of:

- the degree of **test coverage** achieved
- the verification of **completeness** and **interruption criteria**.

Explain what demands are made on **test logging** during **component testing** and how these demands are transferable to other test steps.
Requirements on **test logging** in component testing:

- explicit **identification** of the tested **components** and their respective **versions**
- explicit **identification** of the corresponding **test cases**

Minimum requirements for test logging in other test steps are:

- explicit **identification** of the **software version** tested
- explicit **identification** of the corresponding **test cases**

2.6 Verification of test completion criteria	Explain that the test is only completed when the test completion criteria are fulfilled Illustrate the test completion criteria for the various test steps

If the **test completion criteria** are not fulfilled then normally more tests must be executed and where necessary further specified.

As an alternative, the criteria in the **test plan** could be adjusted in either direction (stronger or weaker). A strengthening of the criteria is to be recommended when unexpected weaknesses are detected during the test.

The risks associated with changing the **test completion criteria** should be weighed up before they are changed in the **test plan**. Furthermore, these changes must be agreed with the responsible parties concerned.

3 Test management 390

3.1 Test management documentation

Describe the following types of document:
- test policy
- test hand book
- test concept
- test step plan

The test policy is the document in which the Company philosophy towards the testing (or quality assurance) of software is described.

The test handbook is a framework document in which the test steps to be excuted and the test activities to be carried out are described.

The test concept describes the test steps and test activities to be executed for a particular project.
The test step plan details the procedure for a test step and describes implementation of the test concept for a particular test step.

In some organisations, the four documents described above are often compressed into one document or subdivided into other types of documents. Altogether though they should contain the same information.

3.1.1 Test policy

Explanation that the organisational beginning of testing commences with the test policy. This is normally developed by the IT department or a similar department but reflects the company philosophy

The test policy is complementary to, or a component of, the quality policy. The quality policy describes the basic views and aims of a company, regarding quality, as pursued by management.

Explain that test policy consists of an outline document with the following content:
- a definition of testing (e.g. checking that the software solves a business problem)
- an outline of the test process (e.g. conception and execution of a test according to departmental procedures and user requirements)
- the evaluation of the test (e.g. measuring the cost of faults which are discovered after release)

- the quality level to be achieved (e.g. not more than one fault of the highest severity per 1000 lines of code in the first 6 months after introduction)
- the procedure for test process improvement (e.g. post project reviews to be carried out after each completion of a project, targeted to reach CMM Level 3)

The test policy concerns test activities for new developments as well as for maintenance.

3.1.2 Test handbook — The test handbook is based on the test policy

The sphere of use of the test handbook covers the generic test requirements for an organisation.

Explain that the test handbook deals with the risks and describes a process whereby these risks can be weakened, in conjunction with the test policy. The connection between risks and testing must be explicitly explained.

A test handbook consists normally of two main parts:
- An explanation of the risks which should be covered by software testing and
- The specific test activities that must be carried out to cover the identified risks.

Explain that a typical test handbook contains a description of the test steps to be used.

Outline roughly the following aspects for each test step:
- Input and output criteria
- The test procedure (top-down, bottom-up, priority driven)
- The test specification technique to be used
- The test completion criteria
- The degree of independence of the testing
- Standards to be maintained
- The environment in which the software tests are executed
- test automation procedure
- The degree of reusability of software
- When to use re-test or regression testing
- The test process to be used, including the test results, e.g. test reports
- Measurements and metrics to be documented
- When to use deviation management

The test handbook must not necessarily consist of one document, but can be divided into a set of documents, e.g. company test handbook, branch test handbook, department test handbook, project test handbook.

Explain why different test handbooks are suitable for different areas of use, and use an example to explain e.g. where security is critical and non-critical.

Explain that measures for improving the test process can be included in later versions of the test handbooks.

3.1.3 Test concept	Explain how the test concept represents the precise use of the test handbook for a particular project

Deviations from the test handbook are documented in the **test concept**.

The **test concept** is referenced from the project plan.

Explain that the **test concept** contains additonal information, which allows the tester:

- to set project costs and testing timescales in order to adhere to the approved budget and resources
- to identify test cycles based on the software release plan
- to convince management and the user of the necessity of testing
- to define and communicate auxiliary services which should be provided by other people or departments
- to identify the project results to be tested.

3.1.4 Test step plan	Explain that the test step plan describes the detailed procedure for a test step. It introduces the refinement of the test concept for one test step

The **test step plan** describes the implementation of the test concept for a particular test step.

Normally it contains the sequence of the test activities in each test step as well as a timetable of activities and the associated milestones.

3.2 Test concept documentation	Explain how a test concept or test step plans can be developed in accordance with the IEEE 829-1998 Standard.

3.3 Test *cost* estimation

Cost estimation is an approximate evaluation of the cost of the test, based on experience

It is necessary to consider the following aspects in **test cost estimation**:
- the number of tests
- the time required for each test step
- the number of repeat tests for each test step.

All test process activities are taken into account in the estimate, e.g. the time for test preparation and execution as well as the time for test palnning.

Explain why, with unknown or poor software quality, the cost to achieve the required test completion criteria is difficult to predict.

The cost estimate can be based on:
- intuition, advice
- experience
- company standards
- a detailed breakdown of all test activities
- based on the formula:
  - function point method
  - test points
  - relative to the development cost (e.g. 40% in new development of software)
- metrics
- comparable new test projects for estimating re-test cycles
- calculation of the average cost per **test object** or **test case** from a previous test run and multiplication by the estimated number of test objects or test cases in the current test run.

3.4 Scheduling test planning

Explain why **test planning** should be done as soon as possible, e.g. in order to notify people whose services are required and to give them enough time for preparation.

Explain how the **test planning** can play a supporting role in the preparation of the development plan for software components. For example, the high priority of a **test object** can be used to decide which components to develop and deliver first.

Explain the advantages of releasing test plans step-by-step, when all the information is not available and delays are to be avoided.

3.5 Test progress monitoring and checking

Explain the different procedures for controlling test progress including monitoring of deviations and test cases as well as the use of statistical methods.

Test progress reports are used to communicate test progress. But the reports must be tailored for the respective addressees (tester, management, etc.).

Explain how the test progress can be presented graphically and in tabular form.
Identify the opportunities of influencing the course of the test process in order to minimise deviations from the test plan, e.g. by:

- correction mechanisms including revision of test priorities
- calling in additional resources
- delaying release in agreement with the project manangement
- altering the test completion criteria.

4 Risk oriented testing 240

4.1 Introduction to risk oriented testing

Explain that the term risk means the probability that a problem may occur in the future

The intention of **risk oriented testing**: under pre-defined boundary conditions (cost, time, availability) prioritise the tests so that the risks are minimised as far as possible. Explain that **risk** is a combination of the possibility of the occurrence of a problem and the resulting effect of that problem.

Risks can be subdivided into **product** and **project risks.** A **product risk** is e.g. a software error which could lead to a system breakdown, a **project risk** is the failure to meet delivery dates.

Illustrate the typical product and project risks, together with risks related to:

- operational safety
- data security
- business related risks
- technical factors
- political factors

Explain that risks are both qualitative and quantitative.

Explain that **risk analysis** can be used for:

- goal oriented testing: different risks will be covered by different **test procedures** or **test depths**
- prioritised testing: areas with a higher risk receive a higher priority
- illustrating the risks to software delivery: curtailing the tests or forgoing them altogether means that the risks, not covered by the testing, still remain.

Testing can be used additionally, for:

- minimising **risks** – a fundamental way towards the reduction of product risks is the finding and removal of errors in a product. Project risks can be reduced by the implementation of a suitable **testing concept**.
- Informing about the **risk evaluation** – high or low error quotas in certain areas of a software product can be used for improving the quality of **risk analysis**

4.2 Risk management

4.2.1 Introduction to risk management

Explain the key activities of risk management:
- risk identification
- risk analysis
- risk avoidance

Ideally, all imembers of the project are involved in risk management during all phases and steps.

4.2.2 Risk identification

The following techniques and utilities can be used for risk identification:
- expert interviews
- independent assessments
- use of risk templates
- risk workshops
- brainstorming
- checklists
- calling on past experience

4.2.3 Risk analysis

Explain risk analysis as the study of the identified risks.

Explain that a risk can be quantified mathematically when the probability of the occurrence of the risk (P) and the corresponding damage (D) can be quantitatively represented. The risk is calculated from the formula P * D.

In most cases the probability and damage cannot be quantified, rather only the tendencies are assignable (e.g. high probability, low probability, higher damage, average damage).

The risk is defined as a graduation within a number of classes or categories.

If there are no dependable metrics available, then the analysis is based on probability perceptions and damage estimation.

Because these risk assessments are normally based on personal perceptions, the results differ, depending on the person making the judgement.

Explain the different perspectives of a project manager, a developer, a tester, and a user.

The degree of insecurity should be recognisable from the results of the risk analysis, which was used to evaluate the risk.

For the evaluation of **risks** in software products, rough risk categories must be set as the basis.

Normal categories refer to:
- functional aspects (especially critical functions or business processes)
- non functional aspects such as **performance**, **security**, **usablity** etc.

Explain that there are various procedures for **risk analysis** with different degrees of accuracy; from quantitative computation to the simple definition of a risk class.

Illustrate examples of **qualitative** and **quantitative risk evaluation**.

Risk analysis is a permanent process during the complete course of a project. Test results can be used as input to the **risk analysis** and lead to new evaluation of risks. Degrees of insecurity can be re-appraised.

If, for example, an increased number of faults is found in a software component, then the quality of this component is lower than expected. The probability of failure of this component increases and with it the failure risk of all the parts of the system which are connected with this component.

An expansion of the testing of the component affected could be an appropriate measure to reduce the risk.

4.2.4 Risk avoidance

Explain that risk avoidance covers activities which were unertaken as a result of the risk analysis

Possible reactions to a recognised risk are:
- do nothing
- distribution of the risk
- take preventive measures for circumventing or reduction of the risk
- set up an emergency plan in case damage ensues.

The selection of a suitable reaction to a **risk** is dependent on the benefits from the removal or reduction of the risk.

Explain that testing is a preventive measure to reduce **risks** by finding and curing errors.

Explain that knowledge about risks can be used to lay down the content of the **test concept**.

Because the budget is usually restricted, risks are normally prioritised. Tests for a high risk are executed first. This knowledge can be used in two ways:

- for defining the **intensity** of the test to be executed based on the degree of risk. Most **safety** standards use this method and define the **test case drafting technique** and **test completion criteria** to be used. For example, **components** which belong to the highest security level require a 100% **branch coverage**.
- For defining the test procedure according to the type of risk. In this way a risk associated with the user interface of a software product, could be minimised by extensive usability testing.

That can mean that low priority risks are not covered by testing, when the project's budget or time has run out.

The risks that are not covered are known to the project management. With this in mind, a decision must be made, whether to accept the remaining risks and release the software product.

5 Test techniques 1.200

5.1 Functional techniques

Introduction to the test case investigation techniques per BS 7925-2:1998[BS 7925-2]:

- Classification tree method [Grochtmann93]
- Equivalence class method
- Boundary value analysis
- Condition driven testing
- Testing for particular coverage
  - Assignment coverage
  - **Branch coverage**
  - **Path coverage**

Introduce **requirements based testing.**

Explain that systematic test case investigation is nearly always connected with a corresponding test termination criteria (end of the test case investigation).

5.2 Non functional testing

Explain the techniques for non functional quality characteristic testing such as reliability and usability and efficiency.

5.3 Dynamic analysis

Explain that dynamic analysis provides runtime information about the execution of a software program. In the main this is achieved through the instrumentation of the tested program. Dynamic analysis watches consumption, use and freeing of memory, memory leaks, non-allocated pointers, pointer arithmetic plus other fault sources which are difficult to investigate statically.

5.4 Static analysis

Explain that static analysis takes place without executing the software to be tested, and that possible errors such as unreachable code, undeclared variables, parameter type inconsistencies, non executed functions and procedures and possible array indexing violations can be detected.

Explain the basic principles of static analysis. A test object, built according to a predefined syntax, is read and analysed.
Often, fault sensitive situations such as

- Syntax errors
- Deviations from conventions or standards
- Control flow anomalies
- Data flow anomalies

are uncovered or measurements extracted using statistic analysis. For example, the number of lines of code in the source are determined by static analysis.

Explain that each fault that the compiler discovers can also be uncovered by static analysis. Many compilers also provide information about the use of variables. These can be useful during software maintenance.

Explain in detail the concept on which data flow analysis is based. Explain that data flow analysis oversees the use of data by paths in the program and looks for anomalies. Anomalies are, e.g. definition of variables without their use or the use of variables after their deletion.

Explain a control flow diagram and derivation of graphs using a program as an example. Control flow diagram notation – see [Hetzel85]

Explain the use of complexity metrics. Calculation of lines of code (LOC) and calculation of cyclical complexity,

5.6 Unsystematic test techniques

Illustrate the use of unsystematic test techniques such as undocumented ad-hoc testing (explorative testing), documented intuitive testing and weak point testing as additional techniques in the test process.

5.7 Selection of test techniques

Explain, which test technique is used and when

Explain that the selection of test techniques can be made based on standards or contractual requirements. Describe the typical general or branch specific standards that demand particular test techniques such as IEC/ISO 61508, DO-178B, standards for railway signalling equipment, nuclear industry standards, pharmaceutical industry standards, "MISRA" requirements for automobile software.
Explain the current day knowledge in regard to the effectiveness of the different test techniques.

6 Reviews 480

6.1 Introduction

Explain the benefits of reviews:
- discovery of faults in early phases of development
- consequent cost effectiveness.

Explain the advantages associated with reviews undertaken in the early development phases:
- avoiding fault propogation into further development phases.
- securing testable requirements

All types of documents can be subjected to a review, e.g. source code, requirements specifications, concepts, test plans, test documents etc.

All types of review have the same objective, fault detection.

Introduction to the standard IEEE 1029-1997 "Standard for Software Reviews"

6.2 Principles

Reviews are best executed as soon as all source documents (documents which describe the project requirements) and standards (to which the project must adhere) are available. If one of the documents or standards is missing then faults and inconsistencies across all documentation cannot be discovered, only those within one document can be discovered.

A review can lead to three possible results:
- the document can be used unchanged
- the document must be changed but a further review is not necessary
- the document must be extensively changed and a further review is necessary.

The people who take part in a review normally undertake different roles:
- manager: makes decisions about the execution of a review
- moderator: leads the review as a neutral person. He is there, if necessary, to moderate between the differing standpoints and is often the key person as regards the success of the review.

- author: is the originator or the person with overall responsibility for the document to be reviewed.
- consultant: technical expert (also called the reviewer or inspector) who takes part in the review after the necessary preparation.
- Secretary (or minute taker): documents clearly and precisely all the deviations, problems, open points etc. discovered during the review.

The consultant carries out the actual review. Some consultants can specialise in certain aspects under which they examine the documents. For this they often use a checklist.

A review is a type of static examination. Following a source code review, a dynamic examination normally follows, designed to find any faults that cannot be found by static examination.

There are different types of review, whose formats are designed differently. The review types should be introduced and compared with each other to show their relative strengths, weaknesses and fields of use. Different types of review pursue different secondary aims.

6.3 Informal reviews

Explain that in informal reviews, one or more consultants (Author excepted) work on a document and give their comments. A review meeting seldom takes place.

6.4 Walkthroughs

To enable the consultants to become familiar with the contents of the review document, it should be distributed to them in good time before the review meeting.
In a walkthrough the author presents the document. Often, scenarios (use cases) and their deployment in the technical design are acted out. Furthermore, dry runs through the code can be used.

In a walkthrough the following secondary aims are pursued:
- examination of alternatives and stylistic questions
- enhancing the consultant's know-how

6.5 Technical reviews Expert review (Peer review)

Explain how a review document is distributed to the consultants and that the consultants must study the document in preparation for the meeting.

The moderator goes through the document step by step. At the appropriate points the consultants introduce the comments that they have formulated during their preparation. The consultants discuss controversial points, reach agreement and decide on the next steps.

More consultants are involved in an expert review than in walkthroughs or inspections.

6.6 Inspection

The model introduced below is based on the review technique published by Michael Fagan (IBM) in 1976 (Fagan's inspections)

The inspection process is formally defined and is strictly followed. The results of a review are dependent on whether the product meets pre-defined requirements.

A preparatory meeting can be used to make all the participants familiar with the review procedures, explain their roles and introduce the review document. The areas where more intensive examination by a particular consultant are required, are defined.

The consultants carry out the inspection of all the review documents received on their own, document their comments and send these back to the moderator before the inspection meeting.
The reader likewise studies the review document to ensure that he is capable of presenting it sensibly at the inspection meeting.
The moderator must be sufficiently qualified to carry out an inspection (e.g. through the corresponding training). The moderator is responsible for the planning of the inspection and the selection of the consultants.

The moderator leads every inspection meeting and confirms that the actions noted therein are carried out. The author is not allowed to take the role of moderator in an inspection meeting and introduce his own document. During the inspection meeting the reader presents and interprets the review documents.
Should it come to a disagreement between consultants during the inspection meeting, the discussion points are noted and discussed at the end of the inspection meeting.

The IEEE Standard recommends that the moderator defines statistically the consultant's preparation time and the number of faults discovered. The cost of discovering the faults will be compared with the estimated cost of not discovering them. The comparison shows the work and cost savings that can be achieved through the review process.

Point to the IEEE standard for details. Michael Fagan's [Fagan 76] original contribution should be used as a secondary reference..

6.7 The introduction of reviews

Explain the steps that are necessary for the effective introduction of reviews:

- securing management support
- execution of pilot project reviews
- demonstrating the benefit of reviews through cost savings
- education and training in review procedures

6.8 Why reviews sometimes fail

Explain the most common causes that are responsible for the lack of success in reviews:

- roles and responsibilities are not understood or are not taken seriously
- insufficient training in the review procedures
- insufficient resources (time, budget, personnel) to carry out reviews
- no general improvement in software development process
- resistance to formal methods
- standards and processes to support reviews are either do not exist or are not available in sufficient quality

Explain that psychological factors can play a decisive role in the acceptance of reviews:

- the document is under scrutiny, not the author
- the consultants support the author in his effort to deliver a fault free product
- authors are not punished or judged when faults are found
- review meetings are confined to the search for faults and are not used to criticise the author.

7 Deviation management 120

Explain and illustrate **deviatiion management** as defined by IEEE Std. 1044-1993 (Standards Classification for Software Anomalies) und IEEE Std. 1044.1-1995 (Guide to Classification for Software Anomalies).

Point out the process steps for recognising, monitoring and removing deviations: Recognition, Investigation, Action, Disposal.

8 Procedure to optimise the testing process as described in [Koomen99] 180

Explain that **improvements** in **procedure** are relevant to the software develpment process as well as for the **testing process**.

Overview of **test process improvement model** SEI Capability Maturity Model (CMMI) and ISO/IEC 15504 (SPICE). Explanation of the SEI CMMI-Model and comparison with SEI CMM and ISO/IEC 15504.

Explain the 5 "degree of maturity" steps in CMMI. Illustrate the most important steps in executing process improvements.

Overview of the SPICE Reference and Assessment Model. Explain the 6 "degree of maturity" steps in SPICE.

Compare CMM-CMMI-SPICE

Explain in detail the Testing Maturiy Model (TMM, as defined by IIT) and procedure for test process optimisation (as described in [Koomen99]).

Explain the 5 "degree of maturity" steps in TMM.

Explain the Key areas and levels of the procedure for optimising test processes (as described in [Koomen99]). Explain the Test Maturity Matrix.

9 Test tools 420

9.1 Overview	There are tools available to support many activities in the test processes. Explain that tools for security critical software must be certified against the corresponding standards. Describe the following types of test tools and explain their advantages and disadvantages
9.2 Terms	Explain the difference between intrusive (i.e. those which affect the test environment) and non-intrusive tools
9.3 Types of tools	Tools for examining requirements support the verification and validation of requirement models e.g. by checking consistency and by animation.

Static analysis tools provide information about the quality of program code. This is accomplished by examination of the program code and not by execution of test cases. Such tools enable an objective measurement of various software characteristics e.g. complexity and other quality attributes.
Many static analysis tools support the compliance of programming guidelines/conventions. Infringements are highlighted with a warning about the rule violated in the source code. The tools come with a standard set of rules for a particular programming language and can be tailored to specific project requirements.

Tools for drafting test inputs generate these inputs from a specification. The specifications are available from CASE tools in the form of formally specified requirements.
There are also tools, which can generate test inputs from the analysis of program code. These tools can also automatically execute the test cases, e.g. in order to test the fault handling of disallowed data types.

Test robots (also called capture-replay) display the user inputs and the system's replies on a screen so that these can be used later for comparison. For applications with a graphical user interface (GUI) the tools can simulate mouse movements and keystrokes and recognise graphical objects such as windows, dialogues, fields and other control elements. The condition of graphical objects can be recorded for later comparison.

Sequences are normally recorded as programmable scripts, that are mostly modified manually to introduce verification steps and to improve the maintainability of the script. One of the most important aims here is often the separation of test script and test data. The user inputs are repeated during the script execution and the systems' responses are compared with those stored in the script. Differences between expected and actual responses are identified by the tool and sometimes automatically recorded in a fault management tool. (see test management tools). Test cases, test data, scripts and expected results can be stored in separate test files. Test robots are usually used for the automation of regression tests.

Test frames and drivers are used for executing software that has no user interface or where the user interface is not available. There are a few commercial products available, but most Test frames and drivers have to be individually developed for a particular project.

Test script generators generate scripts from specifications or direct from the source code.

Simulators are used to support tests if the program code is not available or is not useable in practice (e.g. when testing software in safety critical situations such as Nuclear plants).

Performance (test) tools have two main functions: generating a load and the measurement of test transactions.

Load generators can simulate multiple users or high data volumes. The load will usually be generated by a driver, which emulates the load from several simultaneous users. Different measurements of the transactions are made which are then used to analyse the response times. Normally performance tools deliver various reports and graphs showing the relationship between response time and load.

Tools for analysing web sites are used for collecting information about the number of users and their habits.

Dynamic analysis tools provide information about software during its execution. Such tools are mostly used for identifying the following possible error sources: un-allocated pointers, incorrect pointer arithmetic, incorrect assignment, use and release of memory, memory holes and other errors that cannot be found by static analysis.

The differences between expected and actual results can be recorded using comparison tools. Comparison tools can normally work with a variety of different file and database formats. They are often part of test robots and can handle character oriented user interfaces and/or graphical objects (GUI – objects, bitmaps). These tools support format independent possibilities for filtering out certain information and for masking out data rows and columns.

Test management tools often offer multiple functions. They support the creation, administration and control of documents, e.g. test plans, specifications and results. Some tools support the project management aspects of testing, e.g. chronological test planning, storing results, and recording problems that are uncovered during testing.
Deviation management tools (also known as fault management tools) can support Activities, such as, recording, tracing, controlling and eliminating deviations and repeat testing. Tools from the "test management " category can record test cases and their test coverage and link with other information and objects. Most test management tools offer a diversity of capabilities for report generation and analysis.

Tools for measuring and analysing test coverage measure the degree of coverage of internal program structures during test execution. The program to be tested will be instrumented before test execution. The test coverage will be evaluated by the counters embedded in the program. The tool tracks the detailed information and stores it in a file. These activities slow down the program and can influence the functionality. After the test execution, the stored data is analysed by the tool to create statistics of the test coverage. The majority of the tools provide the most popular test coverage statistics (e.g. pointer and branch coverage).

Hyperlink or link tools check that all links of a web site are present and list links to deleted pages.

Monitoring tools are mostly used for e-commerce or e-business applications and check whether a web site delivers the required user availability and performance. The tools run permanently in the background and give out warnings when a web site or a program is no longer available or the performance boundaries have been violated.

Test tools for checking security aspects are mostly used for e-commerce and e-business applications as well as for web sites. Such a tool checks all the aspects of a web based software system that is in danger of misuse by unauthorised access.

Test oracles supply the expected results. Because such tools must reproduce the functionality of the software to be tested, they are seldom available. A test oracle is for example, an old system that is being replaced by a newer system with the same functionality. The old system supplies the correct expected results for all test cases. Oracles can also be used for systems which must deliver high performance. An oracle with lower performance characteristics can be used or created to generate the functional results that a high performance software system must deliver.

9.4 Tool selection	Explain the necessity for a formal tool evaluation and a thorough selection process. Two objectives will be pursued with these activities • avoiding unnecessary purchase of a tool • guarantee that the tools purchased really will be used

Explain that the tool selection and evaluation process consists ideally of the following steps:

- identification and quantification of the problem area
- considering the alternatives
- cost / benefit analysis
- identification of licensing models and alternatives
- identification of costs which are connected with the ownership of the tool
- identification of limitations
- identification of the functionality and characteristics of the tool
- identification of tools for a detailed evaluation
- execution of a detailed evaluation
- execution of a parallel test run or test drive of competitive tools as necessary.

Explain that an evaluation and selection team normally consists of a project leader (full-time) and some team members (temporary). The team members are drawn from different areas of the company where the tool could be used.

9.5 Tool implementation

Explain that a formal tool implementation procedure is necessary to ensure the long term success of the tool's deployment

Describe the implementation process: it begins with a small pilot project of 3 to 6 months designed to establish the procedure for use of the tool within the company. This procedure includes the architecture of the test article, the script generation method, naming conventions and the configuration management of the test article as required. Ideally the implementation team works full-time on the pilot project and performs the following roles:

The initiator:
- initiates the implementation of the test tool
- understands the different problems and aspects that can arise
- is convinced of the potential advantages of the tool
- radiates enthusiasm and works gladly and constructively with colleagues.

The change manager
- plans, organises and drives the tool implementation (including the plot project)
- can also take on the role of initiator

The person responsible for the tool
- is responsible for the technical support
- provides help and advice internally regarding the use of the tool.

Explain that in addition to these roles and the efforts of further team members, a management sponsor is required who visibly supports the implementation process.

Describe that at the end of the pilot project the results should be compared with the original cost /benefit analysis. If the outcome is positive, the tools and the procedures developed in the pilot project will be available for use in future projects.

10 Team composition 180

10.1 Individual skills

The capability to test software can be obtained through experience or training in different work areas:

- through the use of software systems
- through activities in software development
- through activities in software testing.

The users of software systems know the user side of the system well and have a good knowledge of how the system is operated, where errors have the most damaging effect and what should be the normal reaction of the system. Also, users with a less extensive system knowledge can provide useful tips for testing.

Explain the problem of staff selection, how, for example, to ensure that new staff who have experience of working in a team are included to enhance the team. Illustrate the advantages of having a variety of different personality types in a test team.

Illustrate the concept of role distribution as explanation for the different behaviour of team members, their contribution to team results and their relationships within a team.

As a basis the Team Role Concept from Dr Merdith Belben, Margison-McCann' s Team Management Wheel or Ruth C Cohn's Group Concept Theme Centred Interaction can be used. ([URL:bergdander],[URL: tms-zentrum],[URL:tzi-forum])

10.2 Organisational structure Testers and other project members

Explain that test tasks in different companies can be organised into different structures. Thus the responsibility for testing can lie with the developer himself, with a development team or with a special person from the development team. The responsibility for testing can also be given to a separate test team (which has no development responsibilities), company internal test specialists (who support more projects on a consultative basis) or an external organisation.

Describe the communications methods between:

Testers and developers: testers find faults in software and convey them in as diplomatic a manner as possible, in order to raise the quality of the software. Developers should in turn inform the tester which parts of the system require greater attention because they are more complex or are newly built.

Testers and project management: testers report to the project management on the test progress and the software quality. Project management inform the testers about changes to the extent, environment, project plans and delivery timescales of the testing.

Testers and users: testers receive information about aspects of the system which are particularly important for the users and therefore can then set priorities for the tests. Testers receive background information about the area of application of the system. Users receive test results, which, if necessary are explained by the testers in more detail.

10.3 Motivation

Handling motivational factors like recognition and approval. Handling de-motivational factors such as lack of promotional opportunities.

Recognition and respect are acquired when it is clear that the tester contributes to the incremental value of the project. In an individual project this is most rapidly achieved by immediately involving the tester in the review process. In the course of time the testers will win recognition and respect by their contribution to the positive development of the project. This means that they have documented and explained clearly the project in which they were involved.

Appendix and Literature

Extended reading about this curriculum and its international „relations":

[Spillner 02] Spillner, A.; Linz, T.: Basiswissen Softwaretest, Aus- und Weiterbildung zum Certified-Tester, 1. Aufl., Heidelberg, dpunkt-Verl., 2003, ISBN 3-89864-178-3
Keywords: Testen, Testprozess, Testmanagement, Testtechniken, Abweichungsmanagement, Reviews

[Veenendaal02] Veenendaal, Erik van: The Testing Practioner; UTN Publishers, 2002.
Keywords: test process, risk oriented testing, test techniques, deviation management, reviews, Procedures for Optimising the Test Processes as described in [Koomen99].

Further reading about software testing generally:

[Balzert 00] Balzert, H.: Lehrbuch der Software-Technik; Band 1 und 2; Spektrum Akademischer Verlag, 1998.
Keywords: software development and software quality generally.

[Beizer99] Beizer, B.: Software Testing Techniques, Van Nostrand Reinhold, 1999.
Keywords: test techniques.

[Dustin99] Dustin, E.; Rashaka, J.; Paul, J.: Automated Software Testing, Introduction, Management and Performance, Addison Wesley, 1999
Keywords: test tools

[Fagan76] Fagan, M.E.: „Design and Code Inspections to Reduce Errors in Program Development",
in "IBM Systems Journal", Vol. 15, No. 3, 1976, S. 182-211.
Keyword: reviews

[Fewster99] Fewster, M.; Graham, D.: Software Test Automation; Effective use of test execution tools, Addison Wesley, 1999
Keyword: test tools

[Grochtmann93] Grochtmann, M. and Grimm, K., Classification Trees for Partition Testing, Software Testing, Verification and Reliability, 3:63-82, 1993;
Keyword: test techniques

[Hetzel85] Hetzel, W.C.: The Complete Guide to Software Testing, Collings

[Myers82] Myers, G.J.: Methodisches Testen von Programmen, Oldenbourg, München, Wien 1982. (7. Auflage 2001).

[Koomen99] Koomen, T., Pol, M.: „Vorgehen zur Optimierung des Testprozesses",1999, ACM Press - ISBN 0-201-59624-5.
Keyword: Procedure for Optimising the Test Process as described in [Koomen99]

[URL: bergdander] http://www.bergander.de
Keywords: team assembly, team role concepts.

[URL: tms-zentrum] http://www.tms-zentrum.de
Keywords: team assembly, team management wheel

[URL:tzi-forum] http://www.tzi-forum.com
Keywords: team assembly, theme centered interaction.

Norms and standards:

Non functional Tests:
[URL: BCS SIGIST] http://www.testingstandards.co.uk; The BCS SIGIST Standards Working Party;

[BS 7925-2] British Standard 7925-2, Software testing, Part 2; Software component, 1998

[DIN66272] DIN 66272, Ausgabe:1994-10, Informationstechnik - Bewerten von Softwareprodukten - Qualitätsmerkmale und Leitfaden zu ihrer Verwendung; Identiical withISO/IEC 9126:1991 Original language: German.

[DO-178B] Software Considerations in Airbone Systems and Equipment Certification, RTCA, 1992.

[IEEE 829] IEEE Std. 829-1998, IEEE Standard for Software Test Documentation (Rev. of IEEE Std. 829-1983).

[IEEE1028] IEEE Std 1028-1997, IEEE Standard for Software Reviews.

[IEEE1044] IEEE Std 1044-1993, IEEE Standard Classification for Software Anomalies.

[IEEE1044.1] IEEE Std 1044.1-1995, IEEE Guide to Classification for Software Anomalies.

[IEC 61508] IEC 61508:1998, Functional safefy of electrical/electronic/programmable electronic safety related systems, Industrial Electrical Commitee.

[ISO 9172] ISO/IEC 9126:1991, Bewerten von Softwareprodukten, Qualitätsmerkmale und Leitfaden zu ihrer Verwendung (identisch mit [DIN 66272, 94]).

[ISO/ICE 15504] ISO/IEC 1504 (1998): „Information technology – Software process assessment", International Organization for Standardization.