6 Test Management

This chapter describes ways to organize test teams, which qualifications are important, the tasks of a test manager, and which supporting processes must be present for efficient testing.

6.1 Test Organization

6.1.1 Test Teams

Testing activities are necessary during the entire software product life cycle (see chapter 3). They should be well coordinated with the development activities. The easiest solution is to let the developer perform the testing.

However, because there is a tendency to be blind to our own errors, it is much more efficient to let different people perform testing and development and to organize testing as independently as possible from development.

Benefits of independent testing

Independent testing provides the following benefits:

- Independent testers are unbiased and thus find additional and different defects than developers find.
- An independent tester can verify (implicit) assumptions made by developers during specification and implementation of the system.

Possible drawbacks of independent testing

But there may also be drawbacks to independent testing:

- Too much isolation may impair the necessary communication between testers and developers.
- Independent testing may become a bottleneck if there is a lack of necessary resources.
- Developers may lose a sense of responsibility for quality because they may think, “the testers will find the →problems anyway.”

Models of independent testing

The following models or options for independence are possible:

1. The development team is responsible for testing, but developers test each other's programs, i.e., a developer tests the program of a colleague.
2. There are testers within the development team; these testers do all the test work for their team.
3. One or more dedicated testing teams exist within the project team (these teams are not responsible for development tasks). Typically, team members from the business or IT department work as independent testers.
4. Independent test specialists are used for specific testing tasks (such as performance test, usability test, security test, or for showing conformance to standards and regulatory rules).
5. A separate organization (testing department, external testing facility, test laboratory) takes over the testing (or important parts of it, such as the system test).

When to choose which model

For each of these models, it is advantageous to have testing consultants available. These consultants can support several projects and can offer methodical assistance in areas such as training, coaching, test automation, etc. Which of the previously mentioned models is appropriate depends on—among other things—the current test level.

Component Testing

Testing should be close to development. Although often used, it is definitely the worst choice to allow developers to test their own programs. Independent testing such as in model 1 is easy to organize and would certainly improve quality. Testing such as in model 2 is useful, if a sufficient number of testers relative to the number of developers can be made available. However, with both testing models, there is the risk that the participating people essentially consider themselves developers and thus will neglect their testing responsibilities.

To prevent this, the following measures are recommended:

Hint

- Project or test management should set testing standards and rules, and require test logs from the developers.
- To provide support for applying systematic testing methods, testing specialists should, at least temporarily, be called in as coaches.

Integration Testing

When the same team that developed the components also performs integration and integration testing, this testing can be organized as for component testing (models 1, 2).

If components originating from several teams are integrated, then either a mixed integration team with representatives from the involved development groups or an independent integration team should be responsible. The individual development team may have their own view about their own component and therefore may overlook faults. Depending on the size of the development project and the number of components, models 3, 4, and 5 should be considered here.

System Testing

The final product shall be considered from the point of view of the customer and the end user. Therefore, independence from the development team is crucial. This leaves only models 3, 4, and 5.

Example: Organization of the VSR tests

In the VSR project, each development team is responsible for component testing. These teams are organized according to the previously mentioned models 1 and 2. In parallel to these development teams, an independent testing group is established. This testing group is responsible for integration and system testing. Figure 6-1 shows the organization.

Two or three employees from each responsible user department (sales, marketing, etc.) are made available for the functional or business-process-based testing of every subsystem (*ContractBase*, *DreamCar*, etc.). These people are familiar with the business processes to be supported by the particular subsystem and know the requirements "their" test object should fulfill from the users' point of view. They are experienced PC users, but not IT experts. It is their task to support the test specialists in specifying business-related functional test cases and to perform these tests. When the testing activities are started, they have received training in basic testing procedures (test process, specification, execution, and logging).

→Test logging

Additionally, test personnel consists of three to five IT and test specialists, responsible for integration activities, nonfunctional tests, test automation, and support of test tools ("technical test"). A test manager, responsible for test planning and test control, is in charge of the test team. The manager's tasks also comprise coaching of the test personnel, especially instructing the staff on testing the business requirements.

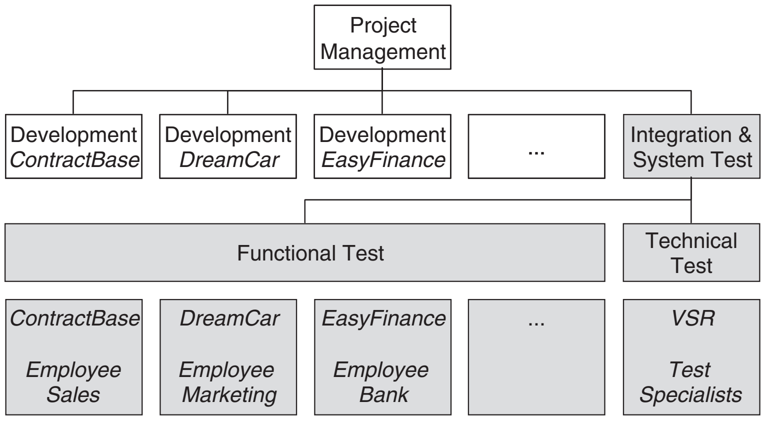

Figure 6–1
VSR project organization

6.1.2 Tasks and Qualifications

Specialists with knowledge covering the full scope of activities in the test process should be available. The following roles should be assigned, ideally to specifically qualified employees:

Roles and qualification profiles

- **Test manager** (test leader): Test planning and test control expert(s), possessing knowledge and experience in the fields of software testing, quality management, project management, and personnel management. Typical tasks may include the following:
  - Writing and coordinating the test policy for the organization
  - Developing the test approach and test plan as described in section 6.2.2
  - Representing the testing perspective in the project
  - Procuring testing resources
  - Selecting and introducing suitable test strategies and methods, introducing or improving testing tools, organizing tools training, deciding about test environment and test automation
  - Introducing or optimizing supporting processes (e.g., problem management, configuration management) in order to be able to trace back changes and securing reproducibility of the tests
  - Introducing, using, and evaluating metrics defined in the test plan
  - Regularly adapting test plans based on test results and test progress
  - Identifying suitable metrics for measuring test progress, and evaluating the quality of the testing and the product
  - Writing and communicating test reports

- **Test designer** (test analyst): Expert(s) in test methods and test specification, having knowledge and experience in the fields of software testing, software engineering, and (formal) specification methods. Typical tasks may include the following:
  - Reviewing requirements, specifications, and models for testability and in order to design test cases
  - Creating test specifications
  - Preparing and acquiring test data
- **Test automator:** Test automation expert(s) with knowledge of testing basics, programming experience, and deep knowledge of the testing tools and script languages. Automates tests as required, making use of the test tools available for the project.
- **Test administrator:** Expert(s) for installing and operating the test environment (system administrator knowledge). Sets up and supports the test environment (often coordinating with general system administration and network management).
- **Tester:**[1] Expert(s) for executing tests and reporting failures (IT basics, basic knowledge of testing, using the test tools, understanding the test object). Typical tasks are as follows:
  - Reviewing test plans and test cases
  - Using test tools and test monitoring tools (for example, to measure performance)
  - Executing and logging tests, including evaluating and documenting the results and detected deficiencies

Certified Tester

In this context, what does the Certified Tester training offer? The basic *Certified Tester* training (Foundation Level) qualifies for the "tester" role (without covering the required IT basics). This means that a Certified Tester knows why discipline and structured work are necessary. Under the supervision of a test manager, a Certified Tester can manually execute and document tests. He or she is familiar with basic techniques for test specification and test management. Every software developer should also know these foundations of software testing to be able to adequately execute the testing tasks required by organizational models 1 and 2. Before someone is able to fulfill the role of a test designer or test manager, appropriate experience as a tester should be gathered. The second educational level (Advanced Level) offers training for the tasks of the designer and manager.

1. The term tester is often also used as generic term for all the previously mentioned roles.

Social competence is important

To be successful, in addition to technical and test-specific skills, a tester needs social skills:

- Ability to work in a team, and political and diplomatic aptitude
- Skepticism (willingness to question apparent facts)
- Persistence and poise
- Accuracy and creativity
- Ability to get quickly acquainted with (complex fields of) application

Multidisciplinary team

Especially in system testing, it is often necessary to extend the test team by adding IT specialists, at least temporarily, to perform work for the test team. For example, these might be database administrators, database designers, or network specialists. Professional specialists from the application field of the software system currently being tested or the business are often indispensable. Managing such a multidisciplinary test team can be difficult even for experienced test managers.

Specialized software test service providers

If appropriate resources are not available within the company, test activities can be given to external software testing service providers. This is similar to letting an external software house develop software. Based on their experience and their use of predefined solutions and procedures, these test specialists are able to provide an optimal test for the project. They can also provide missing specialist skills from each of the previously mentioned qualification profiles for the project.

6.2 Planning

Testing should not be the only measure for quality assurance (QA). It should be used in combination with other quality assurance measures. Therefore, an overall plan for quality assurance is needed that should be documented in the quality assurance plan.

6.2.1 Quality Assurance Plan

Guidelines for structuring the quality assurance plan can be found in IEEE standard 730-2002 [IEEE 730-2002]. The following subjects shall be considered (additional sections may be added as required. Some of the material may also appear in other documents).

Contents of a Software Quality Assurance Plan as defined in IEEE 730-2002:[a]

1. Purpose
2. Reference documents
3. Management
4. Documentation
5. Standards, practices, conventions, and metrics
6. Software reviews
7. Test
8. Problem reporting and corrective action
9. Tools, techniques, and methodologies
10. Media control
11. Supplier control
12. Records collection, maintenance, and retention
13. Training
14. Risk management
15. Glossary
16. SQA Plan Change Procedure and History

a. IEEE Standard 730 in its new form from 2013 [IEEE 730-2013] has a new title, Standard for Software Quality Assurance Processes, and does not contain a standard layout for a software quality assurance plan anymore.

During quality assurance planning, the role the tests play as special, analytical measures of quality control is roughly defined. The details are then determined during test planning and documented in the test plan.

6.2.2 Test Plan

A task as extensive as testing requires careful planning. This planning and test preparation starts as early as possible in the software project. The test policy of the organization and the objectives, risks, and constraints of the project as well as the criticality of the product influence the test plan.

Test planning activities

The test manager might participate in the following planning activities:

- Defining the overall approach to and strategy for testing (see section 6.4)
- Deciding about the test environment and test automation
- Defining the test levels and their interaction, and integrating the testing activities with other project activities

- Deciding how to evaluate the test results
- Selecting metrics for monitoring and controlling test work, as well as defining test exit criteria
- Determining how much test documentation shall be prepared and determining templates
- Writing the test plan and deciding on what, who, when, and how much testing
- Estimating test effort and test costs; (re)estimating and (re)planning the testing tasks during later testing work

The results are documented in the test plan. IEEE Standard 829-1998 [IEEE 829] provides a template.

Test Plan according to IEEE 829-1998

1. Test plan identifier
2. Introduction
3. Test items
4. Features to be tested
5. Features not to be tested
6. Approach
7. Item pass/fail criteria (test exit criteria)
8. Suspension criteria and resumption requirements
9. Test deliverables
10. Testing tasks
11. Environmental needs
12. Responsibilities
13. Staffing and training needs
14. Schedule
15. Risk and contingencies
16. Approvals

This structure[2] works well in practice. The sections listed will be found in real test plans in many projects in the same, or slightly modified, form. The

2. A detailed description of the listed points in IEEE 829-1998 can be found in Appendix A. The new standard [IEEE 829-2008] shows an outline for a master test plan and a level test plan. IEEE Standard 1012 ([IEEE 1012]) gives another reference structure for a verification and validation plan. This standard can be used for planning the test strategy for more complex projects.

new edition of IEEE 829-2008 [IEEE 829-2008] differentiates between "Master Test Plan" and "Level Test Plan." The overall test plan ("Master Test Plan") is required for every project. The different level test plans are optional, depending on the criticality of the product developed. An existing test plan according to IEEE 829-1998 can be changed into the structure of the master test plan in IEEE 829-2008 using mapping or a cross-reference listing. The new standard also has a different approach: There is an explicit requirement for tailoring the test documentation depending on product risks and organizational needs. The standard encourages putting some information from the plans into tools or, if necessary, other plans.

When preparing for an exam using the Foundation syllabus version 2015, IEEE Standard 829-2008, not 1998, should be studied!

Test planning is a continuous activity for the test manager throughout all phases of the development project. The test plan and related plans must be updated regularly, based on feedback from test activities and reacting to changing project risks.

6.2.3 Prioritizing Tests

Even with good planning and control, it is possible that the time and budget for the total test, or for a certain test level, are not sufficient for executing all planned test cases. In this case, it is necessary to select test cases in a suitable way. Even with a reduced number of executable test cases, it must be assured that as many as possible critical faults are found. This means test cases must be prioritized.

Prioritization rule

Test cases should be prioritized so that if any test ends prematurely, the best possible test result at that point of time is achieved.

The most important test cases first

Prioritization also ensures that the most important test cases are executed first. This way important problems can be found early.

The criteria for prioritization, and thus for determining the order of execution of the test cases, are outlined next. Which criteria are used depends on the project, the application area, and the customer requirements.

Criteria for prioritization

The following criteria for prioritization of test cases may be used:

- The **usage frequency** of a function or the **probability** of failure in software use. If certain functions of the system are used often and they contain a fault, then the probability of this fault leading to a failure is high. Thus, test cases for this function should have a higher priority than test cases for a less-often-used function.

- **Failure risk**. Risk is the combination (mathematical product) of severity and failure probability. The severity is the expected damage. Such risks may be, for example, that the business of the customer using the software is impaired, thus leading to financial losses for the customer. Tests that may find failures with a high risk get higher priority than tests that may find failures with low risks (see also section 6.4.3).
- The **visibility** of a failure for the end user is a further criterion for prioritization of test cases. This is especially important in interactive systems. For example, a user of a city information service will feel unsafe if there are problems in the user interface and will lose confidence in the remaining information output.
- Test cases can be chosen depending on the **priority of the requirements**. The different functions delivered by a system have different importance for the customer. The customer may be able to accept the loss of some of the functionality if it behaves wrongly. For other parts, this may not be possible.
- Besides the functional requirements, the **quality characteristics** may have differing importance for the customer. Correct implementation of the important quality characteristics must be tested. Test cases for verifying conformance to required quality characteristics get a high priority.
- Prioritization can also be done from the perspective of development or system architecture. Components that lead to severe consequences when they fail (for example, a crash of the system) should be tested especially intensively.
- **Complexity** of the individual components and system parts can be used to prioritize test cases. Complex program parts should be tested more intensively because developers probably introduced more faults. However, it may happen that program parts seen as easy contain many faults because development was not done with the necessary care. Therefore, prioritization in this area should be based on experience data from earlier projects run within the organization.
- Failures having a high **project risk** should be found early. These are failures that require considerable correction work that in turn requires special resources and leads to considerable delays of the project (see section 6.4.3).

In the test plan, the test manager defines adequate priority criteria and priority classes for the project. Every test case in the test plan should get a priority class using these criteria. This helps in deciding which test cases can be left out if resource problems occur.

Where there are many defects, there are probably more

Where many faults were found before, more are present. This phenomenon occurs often in projects. To react appropriately, it must be possible to change test case priority. In the next test cycle (see section 6.5), additional test cases should be executed for such defect-prone test objects.

Without prioritizing test cases, it is not possible to adequately allocate limited test resources. Concentration of resources on high-priority test cases is a MUST.

6.2.4 Test Entry and Exit Criteria

Defining clear test entry and exit criteria is an important part of test planning. They define when testing can be started and stopped (totally or within a test level).

Test start criteria

Here are typical criteria, or checkpoints, that need to be fulfilled before executing the planned tests:

- The test environment is ready.
- The test tools are ready for use in the test environment.
- Test objects are installed in the test environment.
- The necessary test data is available.

These criteria are preconditions for starting test execution. They prevent the test team from wasting time trying to run tests that are not ready.

Exit criteria

Exit criteria are used to make sure test work is not stopped by chance or prematurely. They prevent tests from ending too early, for example, because of time pressure or because of resource shortages. But they also prevent testing from being too extensive. Here are some typical exit criteria and corresponding metrics or indicators:

- Achieved test coverage: Tests run, covered requirements, code coverage, etc.
- Product quality: Defect density, defect severity, failure rate, and reliability of the test object
- Residual risk: Tests not executed, defects not repaired, incomplete coverage of requirements or code, etc.
- Economic constraints: Allowed cost, project risks, release deadlines, and market chances

The test manager defines the project-specific test exit criteria in the test plan. During test execution, these criteria are then regularly measured and evaluated and serve as the basis for decisions by test and project management.

6.3 Cost and Economy Aspects

Testing can be very costly and can constitute a significant cost factor in software development. How much effort is adequate for testing a specific software product? When is the test cost higher than the possible benefit?

To answer these questions, one must understand the potential defect costs due to lack of checking and testing. Then, one has to compare defect costs and testing costs.

6.3.1 Costs of Defects

If testing activities are reduced or cut out completely, there will be more undetected faults and deficiencies in the product. These remain in the product and may lead to the following costs:

Costs due to product deficiencies

- **Direct defect costs:** Costs that arise for the customer due to failures during operation of the software product (and that the vendor may have to pay for). Examples are costs due to calculation mistakes (data loss, wrong orders, damage of hardware or parts of the technical installation, damage to personnel); costs because of the failure of software-controlled machines, installations, or business processes; and costs due to installation of new versions, which might also require training employees. Very few people think of these costs, but they can be huge. The impact from just the time it takes to install a new version at all customer sites can be enormous.
- **Indirect defect costs:** Costs or loss of sales for the vendor that occur because the customer is dissatisfied with the product. Some examples include penalties or reduction of payment for failure to meet contractual requirements, increased costs for the customer hotline and support, bad publicity, even legal costs such as loss of license (for example, for safety critical software).
- **Costs for defect correction:** Costs paid to vendors for fault correction. For example, time needed for failure analysis, correction, retest and regression test, redistribution and reinstallation, new customer and user training, delay of new products due to tying up the developers with maintenance of the existing product, decreasing competitiveness.