Certified Tester

Advanced Level Syllabus
Test Manager

Version 2012

International Software Testing Qualifications Board

ISTQB™

Copyright Notice
This document may be copied in its entirety, or extracts made, if the source is acknowledged.

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB

Inter
Software
Qualifications

Revision History

Version	Date	Remarks
ISEB v1.1	04SEP01	ISEB Practitioner Syllabus
ISTQB 1.2E	SEP03	ISTQB Advanced Level Syllabus from EOQ-SG
V2007	12OCT07	Certified Tester Advanced Level syllabus version 2007
D100626	26JUN10	Incorporation of changes as accepted in 2009, separation of each chapters for the separate modules
D101227	27DEC10	Acceptance of changes to format and corrections that have no impact on the meaning of the sentences.
D2011	31OCT11	Change to split syllabus, re-worked LOs and text changes to match LOs. Addition of BOs.
Alpha 2012	09Feb12	Incorporation of all comments from NBs received from October release.
Beta 2012	26Mar12	Incorporation of comments from NBs received on time from Alpha release.
Beta 2012	07APR12	Beta version submitted to GA
Beta 2012	08JUN12	Copy edited version released to NBs
Beta 2012	27JUN12	EWG and Glossary comments incorporated
RC 2012	15AUG12	Release candidate version - final NB edits included
GA 2012	19OCT12	Final edits and cleanup for GA release

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB®

Inte...
Software
Qualifications

Table of Contents

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB

Inter...
Software...
Qualifications...

Acknowledgements

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB

Inter...
Software
Qualifications

0. Introduction to this Syllabus

0.1 Purpose of this Document

This syllabus forms the basis for the International Software Testing Qualification at the Advanced Level for the Test Manager. The ISTQB® provides this syllabus as follows:

1. To National Boards, to translate into their local language and to accredit training providers. National Boards may adapt the syllabus to their particular language needs and modify the references to adapt to their local publications.
2. To Exam Boards, to derive examination questions in their local language adapted to the learning objectives for each syllabus.
3. To training providers, to produce courseware and determine appropriate teaching methods.
4. To certification candidates, to prepare for the exam (as part of a training course or independently).
5. To the international software and systems engineering community, to advance the profession of software and systems testing, and as a basis for books and articles.

The ISTQB® may allow other entities to use this syllabus for other purposes, provided they seek and obtain prior written permission.

0.2 Overview

The Advanced Level is comprised of three separate syllabi:

- Test Manager
- Test Analyst
- Technical Test Analyst

The Advanced Level Overview document [ISTQB_AL_OVIEW] includes the following information:

- Business Outcomes for each syllabus
- Summary for each syllabus
- Relationships between the syllabi
- Description of cognitive levels (K-levels)
- Appendices

0.3 Examinable Learning Objectives

The Learning Objectives support the Business Outcomes and are used to create the examination for achieving the Advanced Test Manager Certification. In general all parts of this syllabus are examinable at a K1 level. That is, the candidate will recognize, remember and recall a term or concept. The learning objectives at K2, K3 and K4 levels are shown at the beginning of the pertinent chapter.

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB®

Inte...
Software...
Qualifications...

1. Testing Process – 420 mins.

Keywords

exit criteria, test case, test closure, test condition, test control, test design, test execution, test implementation, test log, test planning, test procedure, test script, test summary report

Learning Objectives for Testing Process

1.2 Test Planning, Monitoring and Control

TM-1.2.1 (K4) Analyze the test needs for a system in order to plan test activities and work products that will achieve the test objectives

1.3 Test Analysis

TM-1.3.1 (K3) Use traceability to check completeness and consistency of defined test conditions with respect to the test objectives, test strategy, and test plan

TM-1.3.2 (K2) Explain the factors that might affect the level of detail at which test conditions may be specified and the advantages and disadvantages for specifying test conditions at a detailed level

1.4 Test Design

TM-1.4.1 (K3) Use traceability to check completeness and consistency of designed test cases with respect to the defined test conditions

1.5 Test Implementation

TM-1.5.1 (K3) Use risks, prioritization, test environment and data dependencies, and constraints to develop a test execution schedule which is complete and consistent with respect to the test objectives, test strategy, and test plan

1.6 Test Execution

TM-1.6.1 (K3) Use traceability to monitor test progress for completeness and consistency with the test objectives, test strategy, and test plan

1.7 Evaluating Exit Criteria and Reporting

TM-1.7.1 (K2) Explain the importance of accurate and timely information collection during the test process to support accurate reporting and evaluation against exit criteria

1.8 Test Closure Activities

TM-1.8.1 (K2) Summarize the four groups of test closure activities

TM-1.8.2 (K3) Implement a project retrospective to evaluate processes and discover areas to improve

1.1 Introduction

The ISTQB® Foundation Level syllabus describes a fundamental test process which includes the following activities:
- Planning and control
- Analysis and design
- Implementation and execution
- Evaluating exit criteria and reporting
- Test closure activities

The Foundation Level syllabus states that although logically sequential, the activities in the process may overlap or take place concurrently. Tailoring these main activities within the context of the system and the project is usually required.

For the Advanced Level syllabi some of these activities are considered separately in order to provide additional refinement and optimization of the processes, better fit with the software development lifecycle, and to facilitate effective test monitoring and control. The activities are now considered as follows:
- Planning, monitoring and control
- Analysis
- Design
- Implementation
- Execution
- Evaluating exit criteria and reporting
- Test closure activities

1.2 Test Planning, Monitoring and Control

This section focuses on the processes of planning, monitoring and controlling testing. As discussed at the Foundation Level, these activities are test management roles.

1.2.1 Test Planning

For each test level, test planning starts at the initiation of the test process for that level and continues throughout the project until the completion of closure activities for that level. It involves the identification of the activities and resources required to meet the mission and objectives identified in the test strategy. Test planning also includes identifying the methods for gathering and tracking the metrics that will be used to guide the project, determine adherence to plan and assess achievement of the objectives. By determining useful metrics during the planning stages, tools can be selected, training can be scheduled and documentation guidelines can be established.

The strategy (or strategies) selected for the testing project help to determine the tasks that should occur during the planning stages. For example, when using the risk-based testing strategy (see Chapter 2), risk analysis is used to guide the test planning process regarding the mitigating activities required to reduce the identified product risks and to help with contingency planning. If a number of likely and serious potential defects related to security are identified, a significant amount of effort should be spent developing and executing security tests. Likewise, if it is identified that serious defects are usually found in the design specification, the test planning process could result in additional static testing (reviews) of the design specification.

Risk information may also be used to determine the priorities of the various testing activities. For example, where system performance is a high risk, performance testing may be conducted as soon as

integrated code is available. Similarly, if a reactive strategy is to be employed, planning for the creation of test charters and tools for dynamic testing techniques such as exploratory testing may be warranted.

In addition, the test planning stage is where the approach to testing is clearly defined by the Test Manager, including which test levels will be employed, the goals and objectives of each level, and what test techniques will be used at each level of testing. For example, in risk-based testing of certain avionics systems, a risk assessment prescribes what level of code coverage is required and thereby which testing techniques should be used.

Complex relationships may exist between the test basis (e.g., specific requirements or risks), test conditions and the tests that cover them. Many-to-many relationships often exist between these work products. These need to be understood to enable effective implementation of test planning, monitoring and control. Tool decisions may also depend on the understanding of the relationships between the work products.

Relationships may also exist between work products produced by the development team and the testing team. For example, the traceability matrix may need to track the relationships between the detailed design specification elements from the system designers, the business requirements from the business analysts, and the test work products defined by the testing team. If low-level test cases are to be designed and used, there may be a requirement defined in the planning stages that the detailed design documents from the development team are to be approved before test case creation can start. When following an Agile lifecycle, informal transfer-of-information sessions may be used to convey information between teams prior to the start of testing.

The test plan may also list the specific features of the software that are within its scope (based on risk analysis, if appropriate), as well as explicitly identifying features that are not within its scope. Depending on the levels of formality and documentation appropriate to the project, each feature that is within scope may be associated with a corresponding test design specification.

There may also be a requirement at this stage for the Test Manager to work with the project architects to define the initial test environment specification, to verify availability of the resources required, to ensure that the people who will configure the environment are committed to do so, and to understand cost/delivery timescales and the work required to complete and deliver the test environment.

Finally, all external dependencies and associated service level agreements (SLAs) should be identified and, if required, initial contact should be made. Examples of dependencies are resource requests to outside groups, dependencies on other projects (if working within a program), external vendors or development partners, the deployment team, and database administrators.

1.2.2 Test Monitoring and Control

In order for a Test Manager to provide efficient test control, a testing schedule and monitoring framework needs to be established to enable tracking of test work products and resources against the plan. This framework should include the detailed measures and targets that are needed to relate the status of test work products and activities to the plan and strategic objectives.

For small and less complex projects, it may be relatively easy to relate test work products and activities to the plan and strategic objectives, but generally more detailed objectives need to be defined to achieve this. This can include the measures and targets to meet test objectives and coverage of the test basis.

Of particular importance is the need to relate the status of test work products and activities to the test basis in a manner that is understandable and relevant to the project and business stakeholders.

Defining targets and measuring progress based on test conditions and groups of test conditions can be used as a means to achieve this by relating other testing work products to the test basis via the test conditions. Properly configured traceability, including the ability to report on traceability status, makes the complex relationships that exist between development work products, the test basis, and the test work products more transparent and comprehensible.

Sometimes, the detailed measures and targets that stakeholders require to be monitored do not relate directly to system functionality or a specification, especially if there is little or no formal documentation. For example, a business stakeholder may be more interested in establishing coverage against an operational business cycle even though the specification is defined in terms of system functionality. Involvement of business stakeholders at an early stage in a project can help define these measures and targets which not only can be used to help provide better control during the project, but can also help to drive and influence the testing activities throughout the project. For example, stakeholder measures and targets may result in the structuring of test design and test implementation work products and/or test execution schedules to facilitate the accurate monitoring of testing progress against these measures. These targets also help to provide traceability for a specific test level and have the potential to help provide information traceability across different test levels.

Test control is an ongoing activity. It involves comparing actual progress against the plan and implementing corrective actions when needed. Test control guides the testing to fulfill the mission, strategies, and objectives, including revisiting the test planning activities as needed. Appropriate reactions to the control data depend on detailed planning information.

The content of test planning documents and test control activities are covered in Chapter 2.

1.3 Test Analysis

Rather than consider test analysis and design together as described in the Foundation Level syllabus, the Advanced syllabi consider them as separate activities, albeit recognizing that they can be implemented as parallel, integrated, or iterative activities to facilitate the production of test design work products.

Test analysis is the activity that defines "what" is to be tested in the form of test conditions. Test conditions can be identified by analysis of the test basis, test objectives, and product risks. They can be viewed as the detailed measures and targets for success (e.g., as part of the exit criteria) and should be traceable back to the test basis and defined strategic objectives, including test objectives and other project or stakeholder criteria for success. Test conditions should also be traceable forward to test designs and other test work products as those work products are created.

Test analysis for a given level of testing can be performed as soon as the basis for testing is established for that level. Formal test techniques and other general analytical techniques (e.g., analytical risk-based strategies and analytical requirements-based strategies) can be used to identify test conditions. Test conditions may or may not specify values or variables depending on the level of testing, the information available at the time of carrying out the analysis and the chosen level of detail (i.e., the degree of granularity of documentation).

There are a number of factors to consider when deciding on the level of detail at which to specify test conditions, including:
- Level of testing
- Level of detail and quality of the test basis
- System/software complexity
- Project and product risk
- The relationship between the test basis, what is to be tested and how it is to be tested
- Software development lifecycle in use

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB®
Inter...
Software...
Qualifications...

- Test management tool being utilized
- Level at which test design and other test work products are to be specified and documented
- Skills and knowledge of the test analysts
- The level of maturity of the test process and the organization itself (note that higher maturity may require a greater level of detail, or allow a lesser level of detail)
- Availability of other project stakeholders for consultation

Specifying test conditions in a detailed fashion will tend to result in a larger number of test conditions. For example, you might have a single general test condition, "Test checkout," for an e-commerce application. However, in a detailed test condition document, this might be split into multiple test conditions, with one condition for each supported payment method, one condition for each possible destination country, and so forth.

Some advantages of specifying test conditions at a detailed level include:
- Facilitates more flexibility in relating other test work products (e.g., test cases) to the test basis and test objectives, thus providing better and more detailed monitoring and control for a Test Manager
- Contributes to defect prevention, as discussed in the Foundation Level, by occurring early in a project for higher levels of testing, as soon as the test basis is established and potentially before system architecture and detailed design are available
- Relates testing work products to stakeholders in terms that they can understand (often, test cases and other testing work products mean nothing to business stakeholders and simple metrics such as number of test cases executed mean nothing to the coverage requirements of stakeholders)
- Helps influence and direct not just other testing activities, but also other development activities
- Enables test design, implementation and execution, together with the resulting work products to be optimized by more efficient coverage of detailed measures and targets
- Provides the basis for clearer horizontal traceability within a test level

Some disadvantages of specifying test conditions at a detailed level include:
- Potentially time-consuming
- Maintainability can become difficult in a changing environment
- Level of formality needs to be defined and implemented across the team

Specification of detailed test conditions can be particularly effective in the following situations:
- Lightweight test design documentation methods, such as checklists, are being used due to accommodate the development lifecycle, cost and/or time constraints or other factors
- Little or no formal requirements or other development work products are available as the test basis
- The project is large-scale, complex or high risk and requires a level of monitoring and control that cannot be delivered by simply relating test cases to development work products

Test conditions may be specified with less detail when the test basis can be related easily and directly to test design work products. This is more likely to be the case for the following:
- Component level testing
- Less complex projects where simple hierarchical relationships exist between what is to be tested and how it is to be tested
- Acceptance testing where use cases can be utilized to help define tests

1.4 Test Design

Test design is the activity that defines "how" something is to be tested. It involves the identification of test cases by the stepwise elaboration of the identified test conditions or test basis using test techniques identified in the test strategy and/or the test plan.

Depending on the approaches being used for test monitoring, test control, and traceability, test cases may be directly related (or indirectly related via the test conditions) to the test basis and defined objectives. These objectives include strategic objectives, test objectives and other project or stakeholder criteria for success.

Test design for a given test level can be performed once test conditions are identified and enough information is available to enable the production of either low or high-level test cases, according to the employed approach to test design. For higher levels of testing, it is more likely that test design is a separate activity following earlier test analysis. For lower levels of testing, it is likely that test analysis and design will be conducted as an integrated activity.

It is also likely that some tasks that normally occur during test implementation will be integrated into the test design process when using an iterative approach to building the tests required for execution; e.g., the creation of test data. In fact, this approach can optimize the coverage of test conditions, either creating low-level or high-level test cases in the process.

1.5 Test Implementation

Test implementation is the activity during which tests are organized and prioritized by the Test Analysts. In formally-documented contexts, test implementation is the activity in which test designs are implemented as concrete test cases, test procedures, and test data. Some organizations following the IEEE 829 [IEEE829] standard define inputs and their associated expected results in test case specifications and test steps in test procedure specifications. More commonly, each test's inputs, expected results, and test steps are documented together. Test implementation also includes the creation of stored test data (e.g., in flat files or database tables).

Test implementation also involves final checks to ensure the test team is ready for test execution to take place. Checks could include ensuring delivery of the required test environment, test data and code (possibly running some test environment and/or code acceptance tests) and that all test cases have been written, reviewed and are ready to be run. It may also include checking against explicit and implicit entry criteria for the test level in question (see Section 1.7). Test implementation can also involve developing a detailed description of the test environment and test data.

The level of detail and associated complexity of work done during test implementation may be influenced by the detail of the test work products (e.g., test cases and test conditions). In some cases, particularly where tests are to be archived for long-term re-use in regression testing, tests may provide detailed descriptions of the steps necessary to execute a test, so as to ensure reliable, consistent execution regardless of the tester executing the test. If regulatory rules apply, tests should provide evidence of compliance to applicable standards (see section 2.9).

During test implementation, the order in which manual and automated tests are to be run should be included in a test execution schedule. Test Managers should carefully check for constraints, including risks and priorities, that might require tests to be run in a particular order or on particular equipment. Dependencies on the test environment or test data must be known and checked.

There may be some disadvantages to early test implementation. With an Agile lifecycle, for example, the code may change dramatically from iteration to iteration, rendering much of the implementation work obsolete. Even without a lifecycle as change-prone as Agile, any iterative or incremental

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB

Inter...
Software
Qualifications

lifecycle may result in significant changes between iterations, making scripted tests unreliable or subject to high maintenance needs. The same is true for poorly-managed sequential lifecycles where the requirements change frequently, even late into the project. Before embarking on an extensive test implementation effort, it is wise to understand the software development lifecycle and the predictability of the software features that will be available for testing.

There may be some advantages in early test implementation. For example, concrete tests provide worked examples of how the software should behave, if written in accordance with the test basis. Business domain experts are likely to find verification of concrete tests easier than verification of abstract business rules, and may thereby identify further weaknesses in software specifications. Such verified tests may provide illuminating illustrations of required behavior for software designers and developers.

1.6 Test Execution

Test execution begins once the test object is delivered and the entry criteria to test execution are satisfied. Tests should be designed or at least defined prior to test execution. Tools should be in place, particularly for test management, defect tracking and (if applicable) test execution automation. Test results tracking, including metrics tracking, should be working and the tracked data should be understood by all team members. Standards for test logging and defect reporting should be available and published. By ensuring these items are in place prior to test execution, the execution can proceed efficiently.

Tests should be executed according to the test cases, although the Test Manager should consider allowing some amount of latitude so that the tester can cover additional interesting test scenarios and behaviors that are observed during testing. When following a test strategy that is at least in part reactive, some time should be reserved for test sessions using experience-based and defect-based techniques. Of course, any failure detected during such unscripted testing must describe the variations from the written test case that are necessary to reproduce the failure. Automated tests will follow their defined instructions without deviation.

The main role of a Test Manager during test execution is to monitor progress according to the test plan and, if required, to initiate and carry out control actions to guide testing toward a successful conclusion in terms of mission, objectives, and strategy. To do so, the Test Manager can use traceability from the test results back to the test conditions, the test basis, and ultimately the test objectives, and also from the test objectives forward to the test results. This process is described in detail in Section 2.6.

1.7 Evaluating Exit Criteria and Reporting

Documentation and reporting for test progress monitoring and control are discussed in detail in Section 2.6.

From the point of view of the test process, it is important to ensure that effective processes are in place to provide the source information necessary for evaluating exit criteria and reporting.

Definition of the information requirements and methods for collection are part of test planning, monitoring and control. During test analysis, test design, test implementation and test execution, the Test Manager should ensure that members of the test team responsible for those activities are providing the information required in an accurate and timely manner so as to facilitate effective evaluation and reporting.

The frequency and level of detail required for reporting are dependent on the project and the organization. This should be negotiated during the test planning phase and should include consultation with relevant project stakeholders.

1.8 Test Closure Activities

Once test execution is determined to be complete, the key outputs should be captured and either passed to the relevant person or archived. Collectively, these are test closure activities. Test closure activities fall into four main groups:

1. Test completion check - ensuring that all test work is indeed concluded. For example, all planned tests should be either run or deliberately skipped, and all known defects should be either fixed and confirmation tested, deferred for a future release, or accepted as permanent restrictions.
2. Test artifacts handover - delivering valuable work products to those who need them. For example, known defects deferred or accepted should be communicated to those who will use and support the use of the system. Tests and test environments should be given to those responsible for maintenance testing. Regression test sets (either automated or manual) should be documented and delivered to the maintenance team.
3. Lessons learned - performing or participating in retrospective meetings where important lessons (both from within the test project and across the whole software development lifecycle) can be documented. In these meetings, plans are established to ensure that good practices can be repeated and poor practices are either not repeated or, where issues cannot be resolved, they are accommodated within project plans. Areas to be considered include the following:
   a. Was the user representation in the quality risk analysis sessions a broad enough cross-section? For example, due to late discovery of unanticipated defect clusters, the team might have discovered that a broader cross-section of user representatives should participate in quality risk analysis sessions on future projects.
   b. Were the estimates accurate? For example, estimates may have been significantly misjudged and therefore future estimation activities will need to account for this together with the underlying reasons, e.g., was testing inefficient or was the estimate actually lower than it should have been.
   c. What are the trends and the results of cause and effect analysis of the defects? For example, assess if late change requests affected the quality of the analysis and development, look for trends that indicate bad practices, e.g., skipping a test level which would have found defects earlier and in a more cost effective manner, for perceived savings of time. Check if defect trends could be related to areas such as new technologies, staffing changes, or the lack of skills.
   d. Are there potential process improvement opportunities?
   e. Were there any unanticipated variances from the plan that should be accommodated in future planning?
4. Archiving results, logs, reports, and other documents and work products in the configuration management system. For example, the test plan and project plan should both be stored in a planning archive, with a clear linkage to the system and version they were used on.

These tasks are important, often missed, and should be explicitly included as part of the test plan.
It is common for one or more of these tasks to be omitted, usually due to premature reassignment or dismissal of project team members, resource or schedule pressures on subsequent projects, or team burnout. On projects carried out under contract, such as custom development, the contract should specify the tasks required.

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB

Inter
Software
Qualifications

2. Test Management – 750 mins.

Keywords

level test plan, master test plan, product risk, project risk, quality risk, risk, risk analysis, risk assessment, risk identification, risk level, risk management, risk mitigation, risk-based testing, test approach, test conditions, test control, test director, test estimation, test leader, test level, test management, test monitoring, test plan, test policy, test strategy, Wide Band Delphi

Learning Objectives for Test Management

2.2 Test Management in Context
TM-2.2.1   (K4) Analyze the stakeholders, circumstances, and needs of a software project or program, including the software development lifecycle model, and identify the optimal test activities
TM-2.2.2   (K2) Understand how software development lifecycle activities and work products affect testing, and how testing affects software development lifecycle activities and work products
TM-2.2.3   (K2) Explain ways to manage the test management issues associated with experience-based testing and non-functional testing

2.3 Risk-Based Testing and Other Approaches for Test Prioritization and Effort Allocation
TM-2.3.1   (K2) Explain the different ways that risk-based testing responds to risks
TM-2.3.2   (K2) Explain, giving examples, different techniques for product risk analysis
TM-2.3.3   (K4) Analyze, identify, and assess product quality risks, summarizing the risks and their assessed level of risk based on key project stakeholder perspectives
TM-2.3.4   (K2) Describe how identified product quality risks can be mitigated and managed, appropriate to their assessed level of risk, throughout the lifecycle and the test process
TM-2.3.5   (K2) Give examples of different options for test selection, test prioritization and effort allocation

2.4 Test Documentation and Other Work Products
TM-2.4.1   (K4) Analyze given samples of test policies and test strategies, and create master test plans, level test plans, and other test work products that are complete and consistent with these documents
TM-2.4.2   (K4) For a given project, analyze project risks and select appropriate risk management options (i.e., mitigation, contingency, transference, and/or acceptance)
TM-2.4.3   (K2) Describe, giving examples, how test strategies affect test activities
TM-2.4.4   (K3) Define documentation norms and templates for test work products that will fit organization, lifecycle, and project needs, adapting available templates from standards bodies where applicable

2.5 Test Estimation
TM-2.5.1   (K3) For a given project, create an estimate for all test process activities, using all applicable estimation techniques
TM-2.5.2   (K2) Understand and give examples of factors which may influence test estimates

2.6 Defining and Using Test Metrics
TM-2.6.1   (K2) Describe and compare typical testing related metrics
TM-2.6.2   (K2) Compare the different dimensions of test progress monitoring

TM-2.6.3   (K4) Analyze and report test results in terms of the residual risk, defect status, test execution status, test coverage status, and confidence to provide insight and recommendations that enable project stakeholders to make release decisions

2.7 Business Value of Testing

TM-2.7.1   (K2) Give examples for each of the four categories determining the cost of quality

TM-2.7.2   (K3) Estimate the value of testing based on cost of quality, along with other quantitative and qualitative considerations, and communicate the estimated value to testing stakeholders

2.8 Distributed, Outsourced, and Insourced Testing

TM-2.8.1   (K2) Understand the factors required for successful use of distributed, outsourced, and insourced test team staffing strategies

2.9 Managing the Application of Industry Standards

TM-2.9.1   (K2) Summarize sources and uses of standards for software testing

---

2.1 Introduction

At the Advanced Level, career specialization has begun to occur for the test professional. This chapter focuses on areas of knowledge required by test professionals as they move into Test Leader, Test Manager, and Test Director positions. In this syllabus, we refer collectively to these professionals as Test Managers, understanding that different organizations will have different definitions for the titles and levels of responsibility of people in such positions.

2.2 Test Management in Context

A central responsibility of a manager is to secure and utilize resources (people, software, hardware, infrastructure, etc.) to carry out value-adding processes. For software and IT managers, the processes are often part of a project or a program aimed at delivering software or a system for internal or external use. For Test Managers, the processes are those involved with testing, specifically the fundamental test process activities described in the Foundation Level syllabus and in Chapter 1 of this syllabus. Since test processes add value only by contributing to the overall success of the project or program (or by preventing a more severe type of failure), the Test Manager must plan and control the test processes accordingly. In other words, the Test Manager must appropriately arrange the test processes, including the associated activities and work products, according to the needs and circumstances of the other stakeholders, their activities (e.g., the software development lifecycle in which testing occurs), and their work products (e.g., requirements specifications).

2.2.1 Understanding Testing Stakeholders

People are stakeholders of testing when they have an interest in the testing activities, the testing work products, or the quality of the final system or deliverable. The stakeholder's interest can be direct or indirect involvement in the testing activities, direct or indirect receipt of testing work products, or direct or indirect effect by the quality of the deliverables produced by the project or program.

While the testing stakeholders vary, depending on the project, the product, the organization, and other factors, they can include the following roles:
- Developers, development leads, and development managers. These stakeholders implement the software under test, receive test results, and often must take action based on those results (e.g., fix reported defects).
- Database architects, system architects, and designers. These stakeholders design the software, receive test results, and often must take action on those results.
- Marketing and business analysts. These stakeholders determine the features, and the level of quality inherent in those features, that must be present in the software. They are also often involved in defining needed test coverage, reviewing test results, and making decisions based on test results.
- Senior management, product managers and project sponsors. These stakeholders are often involved in defining needed test coverage, reviewing test results, and making decisions based on test results.
- Project managers. These stakeholders are responsible for managing their projects to success, which requires balancing quality, schedule, feature, and budget priorities. They often procure the resources required for the test activities and collaborate with the Test Manager in test planning and control.
- Technical support, customer support, and help desk staff. These stakeholders support the users and customers who benefit from the features and quality of the delivered software.
- Direct and indirect users. These stakeholders use the software directly (i.e., they are the end-users), or receive outputs or services produced or supported by the software.

For more on testing stakeholders, see Chapter 2 of [Goucher09].

This list of stakeholders is not comprehensive. Test Managers must identify the specific testing stakeholders for their project or program. The Test Manager must also understand the precise nature of the stakeholder relationship with testing and how the test team serves the needs of the stakeholders. In addition to identifying the test stakeholders as described above, the Test Manager should identify the other software development lifecycle activities and work products that affect testing and/or are affected by testing. Without this, the testing process might not achieve optimal effectiveness and efficiency (see Section 2.2.3).

2.2.2 Additional Software Development Lifecycle Activities and Work Products

Since software testing is an evaluation of the quality of one or more work products produced outside of the testing activities, it usually exists in the context of a larger set of software development lifecycle activities. The Test Manager must plan and guide the testing activities with an understanding of how these other activities and their work products affect testing, as was discussed in the Foundation Level syllabus, and how testing affects these other activities and their work products.

For example, in organizations using Agile development practices, developers often perform test-driven development, create automated unit tests, and continuously integrate code (along with the tests for that code) into the configuration management system. The Test Manager should work with the development manager to ensure that testers are integrated into and aligned with these activities. Testers may review the unit tests both to contribute suggestions for increased coverage and effectiveness of these tests and to gain a deeper understanding of the software and its implementation. Testers may evaluate ways to integrate their own automated tests, especially functional regression tests, into the configuration management system. [Crispin09]

While the specific relationship between testing activities, the other test stakeholders, software development lifecycle work activities, and work products varies depending on the project, the chosen software development lifecycle and a variety of other factors, testing is closely interconnected and related to the following:

- Requirements engineering and management. The Test Manager needs to consider requirements during the scoping and estimation of test effort, as well as remaining aware of changes to the requirements and exercising test control actions to adjust to those changes. Technical Test Analysts and Test Analysts should participate in requirements reviews.
- Project management. The Test Manager, working with Test Analysts and Technical Test Analysts, must provide schedule and resource requirements to the Project Manager. The Test Manager must work with the Project Manager to understand changes in the project plan and exercise test control actions to adjust to those changes.
- Configuration management, release management, and change management. The Test Manager, working with the test team, must establish the test object delivery processes and mechanisms, and capture those in the test plan. The Test Manager may ask Test Analysts and Technical Test Analysts to create build verification tests and to ensure version control during test execution.
- Software development and maintenance. The Test Manager should work with Development Managers to coordinate the delivery of test objects, including content and dates of each test release, as well as participating in defect management (see Chapter 4).
- Technical support. The Test Manager should work with the Technical Support Manager to ensure proper delivery of test results during test closure so that those involved in supporting the product after release are aware of known failures and workarounds. In addition, the Test Manager should work with the Technical Support Manager to analyze production failures in order to implement test process improvements.
- Production of technical documentation. The Test Manager should work with the Technical Documentation Manager to ensure delivery of documentation for testing in a timely fashion, as well as the management of defects found in those documents.

In addition to identifying the test stakeholders as described above, the Test Manager must identify the other software development lifecycle activities and work products that affect testing and/or are affected by testing. If not, the testing process will not achieve optimal effectiveness and efficiency.

2.2.3 Alignment of Test Activities and Other Lifecycle Activities

Testing should be an integral part of the project, regardless of the software development models used. This includes:

- Sequential models, such as the waterfall model, V-model and W-model. In a sequential model, all of the work products and activities for a given phase (e.g., requirements, design, implementation, unit testing, integration testing, system testing, and acceptance testing) are completed before the next phase begins. Test planning, test analysis, test design, and test implementation proceeds in an overlapping fashion with project planning, business/requirements analysis, software and database design, and programming, with the precise nature of the overlap depending on the test level in question. Test execution proceeds sequentially according to the test levels discussed in the Foundation Level syllabus and this syllabus.
- Iterative or incremental models, such as Rapid Application Development (RAD) and the Rational Unified Process (RUP). In an iterative or incremental model, the features to be implemented are grouped together (e.g., according to business priority or risk), and then the various project phases, including their work products and activities, occur for each group of features. The phases may be done either sequentially or in an overlapped fashion, and the iterations themselves may be sequential or overlapping. During project initiation, high-level test planning and test analysis occurs in parallel with the project planning and business/requirements analysis. Detailed test planning, test analysis, test design, and test implementation occurs at the beginning of each iteration, in an overlapping fashion. Test execution often involves overlapping test levels. Each test level begins as early as possible and may continue after subsequent, higher test levels have started.
- Agile, such as SCRUM and Extreme Programming (XP). These are iterative lifecycles where the iterations are very short (often two to four weeks). The work products and activities for each iteration are concluded before the next iteration starts (i.e., the iterations are sequential). Testing proceeds similarly to iterative models, but with a higher degree of overlap of the various testing activities with the development activities, including considerable overlap of test execution (at various levels) with the development activities. All of the activities in an iteration, including the test activities, should be complete before the next iteration starts. In an Agile project, the role of the Test Manager usually changes from a direct managerial role to a technical authority/advisory role.
- Spiral. In a spiral model, prototypes are used early in the project to confirm feasibility and to experiment with design and implementation decisions, using the level of business priority and technical risk to select the order in which the prototyping experiments are carried out. These prototypes are tested to determine what aspects of the technical problems remain unsolved. Once the main technical problems are resolved, the project proceeds according to either a sequential or iterative model.

In order to properly align testing activities within the lifecycle, the Test Manager must have a detailed understanding of the lifecycle models used in their organization. For example, in the V-model, the ISTQB fundamental test process applied to the system test level could align as follows:

- System test planning activities occur concurrently with project planning, and test control continues until system test execution and closure are complete.
- System test analysis and design activities occur concurrently with requirements specification, system and architectural (high-level) design specification, and component (low-level) design specification.

- System test implementation activities might start during system design, though the bulk of these activities would typically occur concurrently with coding and component test, with work on system test implementation activities stretching often until just days before the start of system test execution.
- System test execution activities begin when the system test entry criteria are all met (or waived), which typically means that at least component testing and often also component integration testing are complete. System test execution continues until the system test exit criteria are met.
- Evaluation of the system test exit criteria and reporting of system test results occur throughout system test execution, generally with greater frequency and urgency as project deadlines approach.
- System test closure activities occur after the system test exit criteria are met and system test execution is declared complete, though they can sometimes be delayed until after acceptance testing is over and all project activities are finished.

In an iterative or incremental lifecycle, the same tasks must be performed but the timing and the extent may vary. For example, rather than being able to implement the entire test environment at the beginning of the project, it may be more efficient to implement only the part needed for the current iteration. With any of the iterative or incremental lifecycle models, the farther ahead the planning occurs, the farther ahead the scope of the fundamental test process can extend.

In addition to the planning phases that occur for each project, test execution and reporting may also be influenced by the lifecycle being used by the team. For example, in an iterative lifecycle, it may be effective to produce complete reports and to conduct post-iteration review sessions before the start of the next iteration. By treating each iteration as a mini-project, the team is given an opportunity to correct and adjust based on what occurred during the previous iteration. Because iterations may be short and time constrained, it may make sense to abbreviate the time and effort spent on this reporting and assessment, but the tasks should be conducted as a way to track the overall testing progress and to identify any problem areas as quickly as possible. Process issues experienced in one iteration can easily affect and even recur in the next iteration if corrective measures are not taken.

General information about how to align testing with other lifecycle activities may be captured in the test strategy (see Section 2.4.2). The Test Manager should perform project-specific alignment, for each test level and for any selected combination of software development lifecycle and test process, during test planning and/or project planning.

Depending on the needs of the organization, project, and product, additional test levels beyond those defined in the Foundation Level syllabus may be required, such as:
- Hardware-software integration testing
- System integration testing
- Feature interaction testing
- Customer product integration testing

Each test level should have the following elements clearly defined:
- Test objectives, with achievable goals
- Test scope and test items
- Test basis, along with a means of measuring coverage of that basis (i.e., traceability)
- Entry and exit criteria
- Test deliverables, including results reporting
- Applicable test techniques, along with a way of ensuring the appropriate degrees of coverage using those techniques
- Measurements and metrics relevant to the test objectives, entry and exit criteria, and results reporting (including coverage measurements)

- Test tools to be applied for specific test tasks (if and where applicable)
- Resources (e.g., test environments)
- Responsible individuals and groups, both inside and outside the test team
- Compliance with organizational, regulatory, or other standards (if and where applicable)

As discussed later in this chapter, the best practice is to define these elements coherently across all test levels to avoid wasteful and dangerous gaps across different levels of similar tests.

2.2.4 Managing Non-Functional Testing

Failure to plan for non-functional tests can result in the discovery of serious, sometimes disastrous, quality problems in a system after release. However, many types of non-functional tests are expensive so the Test Manager must select which non-functional tests to perform based on risk and constraints. In addition, there are many different types of non-functional tests, some of which might not be appropriate to a given application.

Since the Test Manager may not have sufficient expertise to handle all the planning considerations, the Test Manager needs to delegate some of test planning responsibilities to the Technical Test Analysts (and in some cases Test Analysts) assigned to the non-functional testing activities. The Test Manager should ask the analysts to consider the following general factors:
- Stakeholder requirements
- Required tooling
- Test environment
- Organizational factors
- Security

For more details, see Chapter 4 of the Advanced Technical Test Analyst syllabus [ISTQB ATTA SYL].

Another important consideration for test managers is how to integrate non-functional tests into the software development lifecycle. A common mistake is to wait until all functional tests are complete prior to starting non-functional tests, which can result in the late discovery of critical non-functional defects. Instead, non-functional tests should be prioritized and sequenced according to risk. There are often ways to mitigate non-functional risks during early levels of testing or even during development. For example, usability reviews of user interface prototypes during system design can be quite effective at identifying usability defects that would create significant schedule problems if discovered toward the end of system testing.

In iterative lifecycles, the pace of change and of iterations can make it difficult to focus on certain non-functional tests that require construction of sophisticated test frameworks. Test design and implementation activities that take longer than the timescales of a single iteration should be organized as separate work activities outside of the iterations.

2.2.5 Managing Experience-Based Testing

While experience-based testing provides benefits by efficiently finding defects that other test techniques may miss and serving as a check on the completeness of those techniques, it provides some challenges for test management. The Test Manager should be aware of the challenges as well as the benefits of the experience-based techniques, particularly exploratory testing. It is difficult to determine the coverage attained during such testing, given the typical light-weight logging and minimal advanced preparation of tests. Reproducibility of the test results requires particular management attention, especially when multiple testers are involved.

One way to manage experience-based testing, especially exploratory testing, is to break the testing work into small, 30 to 120 minute periods sometimes called test sessions. This time boxing limits and

focuses the work to be done in a session and provides a level of monitoring and scheduling. Each session covers a test charter, which is communicated in writing or verbally to the tester by the test manager. The test charter gives the test condition(s) to be covered in the test session, which further helps to maintain focus and prevent overlap if multiple people are carrying out exploratory testing simultaneously.

Another technique to manage experience-based testing is by integrating such self-directed and spontaneous testing into more traditional pre-designed testing sessions. For example, testers can be given permission (and allocated time) to explore beyond the explicit steps, inputs, and expected results in their pre-defined tests. Testers may also be assigned such self-directed testing sessions as part of their daily testing, before, during, or after a day of running pre-defined tests. If such testing sessions identify defects or interesting areas for future testing, the pre-defined tests may be updated.

At the beginning of the exploratory session, the tester ascertains and performs the necessary set up tasks for the tests. During the session, the tester learns about the application being tested, designs and executes tests according to the technique being applied and what has been learned about the application, investigates any defects, and captures the results of the test in a log. (If repeatability of the tests is required, the testers should also log the test inputs, actions, and events.) After the session, a debriefing may occur, which sets the direction for subsequent sessions.

2.3 Risk-Based Testing and Other Approaches for Test Prioritization and Effort Allocation

A universal test management challenge is the proper selection, allocation, and prioritization of tests. That is, out of a practically infinite number of test conditions and combinations of conditions that could be covered, the test team must select a finite set of conditions, determine the appropriate amount of effort to allocate in order to cover each condition with test cases, and sequence the resulting test cases in a prioritized order that optimizes the effectiveness and efficiency of the testing work to be done. The identification and analysis of risk, along with other factors, can be used by the Test Manager to help solve this problem, although many interacting constraints and variables may require a compromised solution.

2.3.1 Risk-Based Testing

Risk is the possibility of a negative or undesirable outcome or event. Risks exist whenever some problem may occur which would decrease customer, user, participant, or stakeholder perceptions of product quality or project success. Where the primary effect of the potential problem is on product quality, potential problems are referred to as quality risks, product risks, or product quality risks. Where the primary effect of the potential problem is on project success, potential problems are referred to as project risks or planning risks.

In risk-based testing, quality risks are identified and assessed during a product quality risk analysis with the stakeholders. The test team then designs, implements, and executes tests to mitigate the quality risks. Quality includes the totality of features, behaviors, characteristics, and attributes that affect customer, user, and stakeholder satisfaction. Therefore, a quality risk is a potential situation where quality problems might exist in a product. Examples of quality risks for a system include incorrect calculations in reports (a functional risk related to accuracy), slow response to user input (a non-functional risk related to efficiency and response time), and difficulty in understanding screens and fields (a non-functional risk related to usability and understandability). When tests reveal defects, testing has mitigated quality risk by providing the awareness of defects and opportunities to deal with them before release. When tests do not find defects, testing has mitigated quality risk by ensuring that, under the tested conditions, the system operates correctly.

Risk-based testing uses product quality risks to select test conditions, to allocate test effort for those conditions, and to prioritize the resulting test cases. A variety of techniques exists for risk-based testing, with significant variation both in the type and level of documentation gathered and in the level of formality applied. Whether explicitly or implicitly, risk-based testing has the objective of using testing to reduce the overall level of quality risk, and, specifically to reduce that level of risk to an acceptable level.

Risk-based testing consists of four main activities:
- Risk identification
- Risk assessment
- Risk mitigation
- Risk management

These activities overlap. The following subsections will discuss each of these activities.

To be most effective, risk identification and assessment should include representatives of all project and product stakeholder groups, though sometimes project realities result in some stakeholders acting as surrogates for other stakeholders. For example, in mass-market software development, a small sample of potential customers may be asked to help identify potential defects that would impact their use of the software most heavily; in this case the sample of potential customers serves as a surrogate for the entire eventual customer base. Because of their particular expertise with product quality risks and failures, testers should be actively involved in the risk identification and assessment process.

2.3.1.1 Risk Identification

Stakeholders can identify risks through one or more of the following techniques:
- Expert interviews
- Independent assessments
- Use of risk templates
- Project retrospectives
- Risk workshops
- Brainstorming
- Checklists
- Calling on past experience

By involving the broadest possible sample of stakeholders, the risk identification process is most likely to identify most of the significant product quality risks.

The risk identification often produces by-products, i.e., identification of issues which are not product quality risks. Examples include general questions or issues about the product or project, or problems in referenced documents such as requirements and design specifications. Project risks are also often identified as a by-product of quality risk identification but are not the main focus of risk-based testing. However, project risk management is important for all testing, not just risk-based testing, and is discussed further in Section 2.4.

2.3.1.2 Risk Assessment

Once risk identification has occurred, risk assessment can begin, being the study of these identified risks. Specifically, risk assessment involves categorizing each risk and determining the likelihood and impact associated with each risk. Risk assessment may also involve evaluating or assigning other properties of each risk, such as risk owner.

Categorization of risk means placing each risk into an appropriate type, such as performance, reliability, functionality, and so forth. Some organizations use the ISO 9126 standard [ISO9126] (which is being replaced by the ISO 25000 standard [ISO25000]) quality characteristics for categorization, but many organizations use other categorization schemes. The same checklist used for risk identification

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB

Inter...
Software...
Qualifications...

is often used to categorize the risks. Generic quality risk checklists exist and many organizations customize these checklists. When using checklists as a foundation of risk identification, categorization of the risk often occurs during identification.

Determining the level of risk typically involves assessing, for each risk item, the likelihood of occurrence and the impact upon occurrence. The likelihood of occurrence is the probability that the potential problem exists in the system under test. In other words, likelihood is an assessment of the level of technical risk. Factors influencing likelihood for product and project risks include:

- Complexity of technology and teams
- Personnel and training issues among the business analysts, designers, and programmers
- Conflict within the team
- Contractual problems with suppliers
- Geographically distributed team
- Legacy versus new approaches
- Tools and technology
- Weak managerial or technical leadership
- Time, resource, budget and management pressure
- Lack of earlier quality assurance activities
- High change rates
- High earlier defect rates
- Interfacing and integration issues

The impact upon occurrence is the severity of the effect on the users, customers, or other stakeholders. Factors influencing impact in project and product risks include:

- Frequency of use of the affected feature
- Criticality of the feature to accomplishing a business goal
- Damage to reputation
- Loss of business
- Potential financial, ecological or social losses or liability
- Civil or criminal legal sanctions
- Loss of license
- Lack of reasonable workarounds
- Visibility of failure leading to negative publicity
- Safety

The level of risk can be assessed either quantitatively or qualitatively. If likelihood and impact can be ascertained quantitatively, one can multiply the two values together to calculate a quantitative risk priority number. Typically, though, the level of risk can only be ascertained qualitatively. That is, one can speak of likelihood being very high, high, medium, low, or very low, but one cannot express likelihood as a percentage with any real precision; similarly, one can speak of impact being very high, high, medium, low, or very low, but one cannot express impact in financial terms in a complete or precise fashion. This qualitative assessment of risk levels should not be seen as inferior to quantitative approaches; indeed, when quantitative assessments of risk levels are used inappropriately, the results mislead the stakeholders about the extent to which one actually understands and can manage risk. Qualitative assessments of risk levels are often combined, through multiplication or addition, to create an aggregate risk score. This aggregate risk score may be expressed as a risk priority number but should be interpreted as a qualitative, relative rating on an ordinal scale.

In addition, unless the risk analysis is based upon extensive and statistically valid risk data, the risk analysis will be based on the stakeholders' subjective perceptions of likelihood and impact. Project managers, programmers, users, business analysts, architects and testers typically have different perceptions and thus possibly different opinions on the level of risk for each risk item. The risk analysis process should include some way of reaching consensus or, in the worst case, establishing an agreed

upon level of risk (e.g., through management dictate or by taking the average, median, or mode level for the risk item). Furthermore, the risk levels should be checked for good distribution through the range, to ensure that the risk rates provide meaningful guidance in terms of test sequencing, prioritization, and effort allocation. Otherwise, risk levels cannot be used as a guide for risk mitigation activities.

2.3.1.3 Risk Mitigation

Risk-based testing starts with a quality risk analysis (identifying and assessing product quality risks). This analysis is the foundation of the master test plan and the other test plans. As specified in the plan(s), tests are designed, implemented, and executed in order to cover the risks.   The effort associated with developing and executing a test is proportional to the level of risk, which means that more meticulous test techniques (such as pairwise testing) are used for higher risks, while less meticulous test techniques (such as equivalence partitioning or time-boxed exploratory testing) are used for lower risks. In addition, the priority of the development and execution of the test is based on the level of risk. Some safety related standards (e.g., FAA DO-178B/ED 12B, IEC 61508), prescribe the test techniques and degree of coverage based on the level of risk. In addition, the level of risk should influence decisions such as the use of reviews of project work products (including tests), the level of independence, the level of experience of the tester, and the degree of confirmation testing (re-testing) and regression testing performed.

During the project, the test team should remain aware of additional information that changes the set of quality risks and/or the level of risk associated with known quality risks. Periodic adjustment of the quality risk analysis, which results in adjustments to the tests, should occur. These adjustments should occur at least at major project milestones. Adjustments include identifying new risks, re-assessing the level of existing risks, and evaluating the effectiveness of risk mitigation activities. To take one example, if a risk identification and assessment session occurred based on the requirements specification during the requirements phase, once the design specification is finalized, a re-evaluation of the risks should occur. To take another example, if during testing a component is found to contain considerably more than the expected number of defects, one can conclude that the likelihood of defects in this area was higher than anticipated and thus adjust the likelihood and overall level of risk upward. This could result in an increase in the amount of testing to be performed against this component.

Product quality risks can also be mitigated before test execution starts. For example, if problems with the requirements are located during risk identification, the project team can implement thorough requirements specification reviews as a mitigating action. This can reduce the level of risk, which can mean that fewer tests are needed to mitigate the remaining quality risk.

2.3.1.4 Risk Management in the Lifecycle

Ideally, risk management occurs throughout the entire lifecycle. If an organization has a test policy document and/or test strategy document, then these should describe the general process by which product and project risks are managed in testing, and show how risk management is integrated into and affects all stages of testing.

In a mature organization where risk awareness pervades the project team, risk management takes place at many levels, and not just for testing.   Important risks are not only addressed earlier in particular test levels, but also in earlier test levels. (For example, if performance is identified as a key quality risk area, not only will performance testing begin early in system test, but performance tests will be run during unit and integration testing.) Mature organizations not only identify risks, but also the sources of risk and the consequence of those risks should they become outcomes. For those defects that do occur, root cause analysis is used to understand sources of risk more deeply and to implement process improvements that prevent defects in the first place. Mitigation exists throughout the software development lifecycle. Risk analysis is fully informed, considering related work activity, analysis of system behavior, cost-based risk assessment, product risk analysis, end user risk analysis, and

liability risk analysis. Risk analysis transcends testing, with the test team participating in and influencing a program-wide risk analysis.

Most risk-based testing methods also include techniques for using the level of risk to sequence and prioritize the testing, thus ensuring early coverage of the most important areas and discovery of the most important defects during test execution. In some cases, all of the highest-risk tests are run before any lower risk tests, and tests are run in strict risk order (often called "depth-first"); in other cases, a sampling approach is used to select a sample of tests across all the identified risk items using risk to weight the selection while at the same time ensuring coverage of every risk item at least once (often called "breadth-first").

Whether risk-based testing proceeds depth-first or breadth-first, it is possible that the time allocated for testing might be consumed without all tests being run. Risk-based testing allows testers to report to management in terms of the remaining level of risk at this point, and allows management to decide whether to extend testing or to transfer the remaining risk onto the users, customers, help desk/technical support, and/or operational staff.

During test execution, the most sophisticated risk-based testing techniques—which need not be the most formal or heavy-weight—allow project participants, project and product managers, executives, senior managers, and project stakeholders, to monitor and control the software development lifecycle, including making release decisions, based on the residual level of risk. This requires the Test Manager to report test results in terms of risk in a way each test stakeholder can understand.

2.3.2 Risk-Based Testing Techniques

There are a number of techniques for risk-based testing. Some of these techniques are highly informal. Examples include approaches where the tester analyzes quality risks during exploratory testing [Whittaker09]. This can help guide the testing, but can lead to an excessive focus on the likelihood of defects, not their impact, and does not include cross-functional stakeholder input. In addition, such approaches are subjective and dependent on the skills, experience, and preferences of the individual tester. As such, these approaches seldom achieve the full benefits of risk-based testing.

In an attempt to capture the benefits of risk-based testing while minimizing the costs, many practitioners employ lightweight risk-based approaches. These blend the responsiveness and flexibility of informal approaches with the power and consensus building of more formal approaches. Examples of lightweight approaches include: Pragmatic Risk Analysis and Management (PRAM) [Black09], Systematic Software Testing (SST) [Craig02], and Product Risk Management (PRisMa) [vanVeenendaal12]. In addition to the usual attributes of risk-based testing, these techniques typically have the following attributes:

- Evolved over time based on industry experience with risk-based testing, especially in industries where questions of efficiency are important
- Predicated on the extensive involvement of a cross-functional team of stakeholders, representing both business and technical perspectives, during the initial risk identification and assessment
- Optimized when introduced during the earliest phases of a project, when the options to mitigate quality risks are maximized and when main work products and by-products of the risk analysis can help to influence the specification and implementation of the product in a way that minimizes risk
- Use the generated output (the risk matrix or risk analysis table) as the basis for the test plan and the test conditions, and thus all subsequent test management and analysis activities
- Support the reporting of test results in terms of residual risk to all levels of testing stakeholders

Some of these techniques (e.g., SST) require requirements specifications as an input into the risk analysis and cannot be used except when requirements specifications are provided. Other techniques

(e.g., PRisMa and PRAM) encourage the use of a blended risk- and requirements-based strategy, using the requirements and/or other specifications as an input to the risk analysis, but can function entirely based on stakeholder input. The use of requirements as an input helps ensure good coverage of the requirements, but the Test Manager must ensure that important risks not suggested by the requirements—especially in non-functional areas—are not missed. When good, prioritized requirements are provided as an input, one typically sees a strong correlation between risk levels and requirement priority.

Many of these techniques also encourage the use of the risk identification and assessment process as a way to create consensus across stakeholders on the test approach. This is a powerful benefit, but does require the stakeholders to dedicate time to participate either in group brainstorming sessions or in one-on-one interviews. Insufficient stakeholder participation leads to gaps in the risk analysis. Of course, stakeholders do not always agree on the level of risk, so the person leading a quality risk analysis effort must work creatively and positively with stakeholders to achieve the best possible degree of agreement. All the skills of a trained moderator leading review meetings are applicable to the person leading a quality risk analysis.

Like more formal techniques, lightweight techniques allow the use of weighting of the likelihood and impact factors to emphasize business or technical risks (depending on the level of testing, for example). Unlike more formal techniques, though, lightweight techniques:
- use only two factors, likelihood and impact; and,
- use simple, qualitative judgments and scales.

The lightweight nature of these approaches provides flexibility, applicability in a range of domains, and accessibility across teams of all experience and skill levels (even with non-technical and junior people).

At the formal, heavy-weight end of the scale, the Test Manager has a number of options available:
- Hazard analysis, which extends the analytical process upstream, attempting to identify the hazards that underlie each risk.
- Cost of exposure, where the risk assessment process involves determining, for each quality risk item, three factors: 1) the likelihood (expressed as a percentage) of a failure related to the risk item; 2) the cost of a loss (expressed as a financial quantity) associated with a typical failure related to the risk item, should it occur in production; and, 3) the cost of testing for such failures.
- Failure Mode and Effect Analysis (FMEA) and its variants [Stamatis03], where quality risks, their potential causes, and their likely effects are identified and then severity, priority and detection ratings are assigned.
- Quality Function Deployment (QFD), which is a quality risk management technique with testing implications, specifically, being concerned with quality risks that arise from an incorrect or insufficient understanding of the customers' or users' requirements.
- Fault Tree Analysis (FTA), where various actual observed failures (from testing or from production), or potential failures (quality risks), are subjected to a root cause analysis starting with defects that could cause the failure, then with errors or defects that could cause those defects, continuing on until the various root causes are identified.

The specific techniques that should be used for risk-based testing, and the degree of formality of each technique, depend on project, process, and product considerations. For example, an informal approach, such as Whittaker's exploratory technique, might apply to a patch or quick fix. In Agile projects, the quality risk analysis is fully integrated into the early period of each sprint and the documentation of risks is blended into the user story tracking. Systems of systems require risk analysis on each system as well as on the overall system of systems. Safety-critical and mission-critical projects require higher levels of formality and documentation.

The inputs, processes, and outputs for risk-based testing will tend to be determined by the selected technique. Common inputs include stakeholder insights, specifications, and historical data. Common processes include risk identification, risk assessment, and risk control. Common outputs include a list of quality risks (with the associated level of risk and the recommended allocation of test effort), defects discovered in input documents such as specifications, questions or issues related to the risk items, and project risks affecting testing or the project as a whole.

Usually the most critical success factor for risk-based testing is the involvement of the right team of stakeholders in the risk identification and assessment. All stakeholders have their own understanding of what constitutes quality for the product, and their own set of priorities and concerns about quality. However, stakeholders tend to fall into two broad categories: business stakeholders and technical stakeholders.

Business stakeholders include customers, users, operation staff, help desk and technical support staff, among others. These stakeholders understand the customer and user, and thus can identify risks and assess impact from a business perspective.

Technical stakeholders include developers, architects, database administrators, and network administrators, among others. These stakeholders understand the underlying ways in which software can fail, and thus can identify risks and assess likelihood from a technical perspective.

Some stakeholders have both a business and a technical perspective. For example, subject matter experts who are in testing or business analysis roles often have a broader view on the risks due to their technical and business expertise.

The process of identifying risk items is one that generates a substantial list of risks. There is no need for stakeholders to argue about the risk items; so long as one stakeholder perceives something as a risk to the quality of the system, it is a risk item. However, it is important that stakeholders achieve consensus on their ratings for the level of risk. For example, in lightweight approaches that use likelihood and impact as the rating factors, part of the process must include finding a common ranking scheme for likelihood and impact. All stakeholders, including the testing group, must use this same scale, and must be able to converge on a single likelihood and impact rating for each quality risk item.

If risk-based testing is to be used long-term, then the Test Manager must successfully advocate and initiate risk-based testing with the stakeholders. The cross-functional group must see the value of the risk analysis to ensure on-going use of the technique. This requires that the Test Manager understand the stakeholders' needs, expectations, and time availability to participate in the process.

Good stakeholder engagement with the quality risk analysis process offers an important benefit to the Test Manager. This benefit is that, on under-specified projects with weak or missing requirements, the stakeholders can still identify risks when guided with the proper checklist. This benefit can be seen when, after the implementation of risk-based testing, the defect detection effectiveness of the test team improves. This happens because a more complete test basis—in this case the list of quality risk items—is being used.

During test closure for risk-based testing, test teams should measure the extent to which they realized the benefits. In many cases, this involves answering some or all of the following four questions through metrics and consultation with the team:

- Did the test team detect a greater percentage of important defects than of less-important defects?
- Did the test team find most of the important defects early in the test execution period?
- Was the test team able to explain the test results to stakeholders in terms of risk?
- Did the tests the test team skipped (if any) have a lower level of associated risk than those executed?

In most cases, successful risk-based testing results in an affirmative answer for all four questions. In the long term, the Test Manager should set process improvement goals for these metrics, along with striving to improve the efficiency of the quality risk analysis process. Of course, other metrics and success criteria can be applied to risk-based testing, and the Test Manager should consider carefully the relationship between these metrics, the strategic objectives the test team serves, and the behaviors that will result from management based on a particular set of metrics and success criteria.

2.3.3 Other Techniques for Test Selection

While many Test Managers employ risk-based testing as one element of their test strategy, many also include other techniques.

One of the most prominent alternative techniques for developing and prioritizing test conditions is requirements-based testing. Requirements based testing may utilize several techniques, such as ambiguity reviews, test condition analysis, and cause-effect graphing. Ambiguity reviews identify and eliminate ambiguities in the requirements (which serve as the test basis), often by using a checklist of common requirements defects (see [Wiegers03]).

As described in [Craig02], test condition analysis involves a close reading of the requirements specification to identify the test conditions to cover. If those requirements have an assigned priority, this can be used to allocate effort and prioritize the test cases. In the absence of an assigned priority, it is difficult to determine the appropriate effort and order of testing without blending requirements-based testing with risk-based testing.

Cause-effect graphing is discussed in the Advanced Test Analyst module in the context of covering combinations of test conditions as part of test analysis. However, it has a broader usage in that it can reduce an extremely large testing problem to a manageable number of test cases and still provide 100% functional coverage of the test basis. Cause-effect graphing also identifies gaps in the test basis during test case design, which can identify defects early in the software development lifecycle when test design is started against the draft requirements. One major impediment to the adoption of cause-effect graphing is the complexity of creating these graphs. There are tools to support this method which can be complicated to do manually.

A general obstacle to requirements-based testing is that often the requirements specifications are ambiguous, untestable, incomplete, or non-existent. Not all organizations are motivated to solve these problems, so testers confronted with such situations must select another test strategy.

Another method which is sometimes used to augment the use of the existing requirements is the creation of usage or operational profiles, a model-based approach, which utilizes a mix of use cases, users (sometimes called personas), inputs, and outputs, so as to accurately depict the real-world use of the system. This allows testing not only of functionality but also of usability, interoperability, reliability, security and performance.

During test analysis and planning, the test team identifies the usage profiles and attempts to cover them with test cases. The usage profile is an estimate, based on available information, of realistic usage of the software. This means that, as with risk-based testing, the usage profiles may not perfectly model the eventual reality. However, if enough information and stakeholder input is available, then the model will be adequate (see [Musa04]).

Some Test Managers also apply methodical approaches such as checklists to determine what to test, how much, and in what order. For products that are very stable, a checklist of major functional and non-functional areas to test, combined with a repository of existing test cases, can be sufficient. The checklist provides heuristics for the allocation of effort and the sequencing of tests, usually based on

the type and amount of changes that have occurred. Such approaches tend to become less valid when used to test more than minor changes.

Finally, another common method is to use a reactive approach. In a reactive approach, very few test analysis, design, or implementation tasks occur prior to test execution. The test team is focused on reacting to the product as actually delivered. Bug clusters, as discovered, become the focus of further testing. Prioritization and allocation are completely dynamic. The reactive approach can work as a complement to other approaches, but when employed exclusively the reactive approach tends to miss major areas of the application that are important but not suffering from a large number of bugs.

2.3.4 Test Prioritization and Effort Allocation in the Test Process

Whatever technique—or, better yet, mix of techniques—that a Test Manager uses, the Test Manager must incorporate that technique into the project and test processes. For example, in sequential lifecycles (e.g., V-model), the test team selects tests, allocates test effort, and initially prioritizes tests during the requirements phase, with periodic adjustments, whereas iterative or Agile lifecycles require an iteration-by-iteration approach. Test planning and control must consider the degree to which the risks, requirements, and/or usage profiles will evolve, and respond accordingly.

During test analysis, design, and implementation, the allocation and prioritization determined during test planning must be applied. It is a common breakdown in the test process for careful analysis and/or modeling to occur, only for that information not to be used to guide the test process going forward. This breakdown typically occurs during design and implementation.

In test execution, the prioritization determined during test planning should be carried out, although it is important to update the prioritization periodically based on information gained after the plan was initially written. When evaluating and reporting the test results and exit criteria status, the Test Manager must also evaluate and report in terms of the risks, requirements, usage profiles, checklists, and other guides used to select and prioritize tests. If necessary, test triage should occur based on the test prioritization scheme.

As part of results reporting and exit criteria evaluation, the Test Manager can measure the degree to which testing is complete. This should include tracing test cases and discovered defects back to the relevant test basis. For example, in risk-based testing, as tests are run and defects found, testers can examine the remaining, residual level of risk. This supports the use of risk-based testing in determining the right moment to release. Test reporting should address risks covered and still open, as well as benefits achieved and not yet achieved. For an example of reporting test results based on risk coverage, see [Black03].

Finally, during test closure, the Test Manager should evaluate metrics and success criteria which are pertinent to the needs and expectations of the testing stakeholders, including the customers' and users' needs and expectations in terms of quality. Only when testing satisfies these needs and expectations can a test team claim to be truly effective.

2.4 Test Documentation and Other Work Products

Documentation is often produced as part of the test management activities. While the specific names of the test management documents and the scope of each document tend to vary, the following are common types of test management documents found in organizations and on projects:
- Test policy - describes the organization's objectives and goals for testing
- Test strategy - describes the organization's general, project-independent methods for testing
- Master test plan (or project test plan) - describes the implementation of the test strategy for a particular project

- Level test plan (or phase test plan) - describes the particular activities to be carried out within each test level

The physical arrangements of these types of document may differ from context to context. In some organizations and on some projects, they may be combined into a single document; in others, they may be found in separate documents; and in some, their contents may be manifested as intuitive, unwritten, or traditional methodologies for testing. Larger and more formal organizations and projects tend to have all of these types of documents as written work products, while smaller and less formal organizations and projects tend to have fewer such written work products. This syllabus describes each of these types of documents separately, though in practice the organizational and project context determines the correct utilization of each type.

2.4.1 Test Policy

The test policy describes why the organization tests. It defines the overall objectives for testing that the organization wants to achieve. This policy should be developed by senior test management staff in the organization in collaboration with senior managers for the testing stakeholder groups.

In some cases, the test policy will be complementary to or a component of a broader quality policy. This quality policy describes the overall values and goals of management related to quality.

Where a written test policy exists, it may be a short, high-level document that:
- Summarizes the value that the organization derives from testing
- Defines the objectives of testing, such as building confidence in the software, detecting defects in the software, and reducing the level of quality risk (see Section 2.3.1)
- Describes how to evaluate the effectiveness and efficiency of testing in meeting these objectives
- Outlines the typical test process, perhaps using the ISTQB fundamental test process as a basis
- Specifies how the organization will improve its test processes (see Chapter 5)

The test policy should address test activities for new development as well as for maintenance. It may also reference internal and/or external standards for the testing work products and terminology to be used throughout the organization.

2.4.2 Test Strategy

The test strategy describes the organization's general test methodology. This includes the way in which testing is used to manage product and project risks, the division of testing into levels, and the high-level activities associated with testing. (The same organization may have different strategies for different situations, such as different software development lifecycles, different levels of risk, or different regulatory requirements.) The test strategy, and the processes and activities described in it, should be consistent with the test policy. It should provide the generic test entry and exit criteria for the organization or for one or more programs.

As mentioned above, test strategies describe general test methodologies, which typically include:
- Analytical strategies, such as risk-based testing, where the test team analyzes the test basis to identify the test conditions to cover. For example, in requirements-based testing, test analysis derives test conditions from the requirements, tests are then designed and implemented to cover those conditions. The tests are subsequently executed, often using the priority of the requirement covered by each test to determine the order in which the tests will be run. Test results are reported in terms of requirements status, e.g., requirement tested and passed, requirement tested and failed, requirement not yet fully tested, requirement testing blocked, etc.

- Model-based strategies, such as operational profiling, where the test team develops a model (based on actual or anticipated situations) of the environment in which the system exists, the inputs and conditions to which the system is subjected, and how the system should behave. For example, in model-based performance testing of a fast-growing mobile device application, one might develop models of incoming and outgoing network traffic, active and inactive users, and resulting processing load, based on current usage and project growth over time. In addition, models might be developed considering the current production environment's hardware, software, data capacity, network, and infrastructure. Models may also be developed for ideal, expected, and minimum throughput rates, response times, and resource allocation.

- Methodical strategies, such as quality characteristic-based, where the test team uses a predetermined set of test conditions, such as a quality standard (e.g., ISO 25000 [ISO2500] which is replacing ISO 9126 [ISO9126]), a checklist or a collection of generalized, logical test conditions which may relate to a particular domain, application or type of testing (e.g., security testing), and uses that set of test conditions from one iteration to the next or from one release to the next. For example, in maintenance testing a simple, stable e-commerce website, testers might use a checklist that identifies the key functions, attributes, and links for each page. The testers would cover the relevant elements of this checklist each time a modification is made to the site.

- Process- or standard-compliant strategies, such as medical systems subject to U.S. Food and Drug Administration standards, where the test team follows a set of processes defined by a standards committee or other panel of experts, where the processes address documentation, the proper identification and use of the test basis and test oracle(s), and the organization of the test team. For example, in projects following Scrum Agile management techniques, in each iteration testers analyze user stories that describe particular features, estimate the test effort for each feature as part of the planning process for the iteration, identify test conditions (often called acceptance criteria) for each user story, execute tests that cover those conditions, and report the status of each user story (untested, failing, or passing) during test execution.

- Reactive strategies, such as using defect-based attacks, where the test team waits to design and implement tests until the software is received, reacting to the actual system under test. For example, when using exploratory testing on a menu-based application, a set of test charters corresponding to the features, menu selections, and screens might be developed. Each tester is assigned a set of test charters, which they then use to structure their exploratory testing sessions. Testers periodically report results of the testing sessions to the Test Manager, who may revise the charters based on the findings.

- Consultative strategies, such as user-directed testing, where the test team relies on the input of one or more key stakeholders to determine the test conditions to cover. For example, in outsourced compatibility testing for a web-based application, a company may give the outsourced testing service provider a prioritized list of browser versions, anti-malware software, operating systems, connection types, and other configuration options that they want evaluated against their application. The testing service provider can then use techniques such as pairwise testing (for high priority options) and equivalence partitioning (for lower priority options) to generate the tests.

- Regression-averse testing strategies, such as extensive automation, where the test team uses various techniques to manage the risk of regression, especially functional and/or non-functional regression test automation at one or more levels. For example, if regression testing a web-based application, testers can use a GUI-based test automation tool to automate the typical and exception use cases for the application. Those tests are then executed any time the application is modified.

Different strategies may be combined. The specific strategies selected should be appropriate to the organization's needs and means, and organizations may tailor strategies to fit particular operations and projects.

The test strategy may describe the test levels to be carried out. In such cases, it should give guidance on the entry criteria and exit criteria for each level and the relationships among the levels (e.g., division of test coverage objectives).

The test strategy may also describe the following:
- Integration procedures
- Test specification techniques
- Independence of testing (which may vary depending on level)
- Mandatory and optional standards
- Test environments
- Test automation
- Test tools
- Reusability of software work products and test work products
- Confirmation testing (re-testing) and regression testing
- Test control and reporting
- Test measurements and metrics
- Defect management
- Configuration management approach for testware
- Roles and responsibilities

Different test strategies for the short term and the long term might be necessary. Different test strategies are suitable for different organizations and projects. For example, where security- or safety-critical applications are involved, a more intensive strategy may be more appropriate than in other cases. In addition, the test strategy also differs for the various development models.

2.4.3 Master Test Plan

The master test plan covers all of the testing work to be done on a particular project, including the particular levels to be carried out and the relationships among those levels, and between test levels and corresponding development activities. The master test plan should discuss how the testers will implement the test strategy for this project (i.e., the test approach). The master test plan should be consistent with the test policy and strategy, and, in specific areas where it is not, should explain those deviations and exceptions, including any potential impact resulting from the deviations. For example, if it is an organization's testing strategy to conduct one complete pass of regression testing on an unchanging system immediately prior to release, but the current project will have no regression testing, the test plan should explain why this is planned and what will be done to mitigate any risk due to this variance from the usual strategy. The test plan should also include an explanation of any other effects that are expected from this variance. For example, skipping regression testing might require scheduling a maintenance release one month after the initial project release. The master test plan should complement the project plan or operations guide in that it should describe the testing effort that is part of the larger project or operation.

While the specific content and structure of the master test plan varies depending on the organization, its documentation standards, and the formality of the project, typical topics for a master test plan include:
- Items to be tested and not to be tested
- Quality characteristics to be tested and not to be tested
- Testing schedule and budget (which should be aligned with the project or operational budget)
- Test execution cycles and their relationship to the software release plan
- Relationships and deliverables among testing and other people or departments
- Definition of what test items are in scope and out of scope for each level described

- Specific entry criteria, continuation (suspension/resumption) criteria, and exit criteria for each level and the relationships among the levels
- Test project risks
- Overall governance of the testing effort
- Responsibilities for executing each of the test levels
- Inputs to and outputs from each of the test levels

On smaller projects or operations, where only one level of testing is formalized, the master test plan and the test plan for that formalized level will often be combined into one document. For example, if system test is the only formalized level, with informal component and integration testing performed by developers and informal acceptance testing performed by customers as part of a beta test process, then the system test plan may include the elements mentioned in this section.

In addition, testing typically depends on other activities in the project. Should these activities not be sufficiently documented, particularly in terms of their influence and relationship with testing, topics related to those activities may be covered in the master test plan (or in the appropriate level test plan). For example, if the configuration management process is not documented, the test plan should specify how test objects are to be delivered to the test team.

2.4.4 Level Test Plan

Level test plans describe the particular activities to be carried out within each test level or, in some cases, test type. Level test plans expand, where necessary, on the master test plan for the specific level or test type being documented. They provide schedule, task, and milestone details not necessarily covered in the master test plan. In addition, to the extent that different standards and templates apply to specification of tests at different levels, these details would be covered in level test plans.

On less formal projects or operations, a single test plan is often the only test management document written. In such situations, some of the informational elements mentioned earlier in this section could be covered in this test plan document.

For Agile projects, sprint or iteration test plans may take the place of level test plans.

2.4.5 Project Risk Management

An important part of proper planning includes dealing with project risks. Project risks can be identified using processes similar to those described in Section 2.3. When project risks are identified, they should be communicated to and acted upon by the project manager. Such risks are not always within the power of the testing organization to reduce. However, some project risks can and should be mitigated successfully by the Test Manager, such as:
- Test environment and tools readiness
- Test staff availability and qualification
- Lack of standards, rules and techniques for the testing effort

Approaches to project risk management include preparing the testware earlier, pre-testing of test environments, pre-testing of early versions of the product, applying tougher entry criteria to testing, enforcing requirements for testability, participating in reviews of early project work products, participating in change management, and monitoring the project progress and quality.

Once a project risk has been identified and analyzed, there are four main options to manage that risk:
1. Mitigate the risk through preventive measures to reduce likelihood and/or impact
2. Make contingency plans to reduce impact if the risk becomes an actuality
3. Transfer the risk to some other party to handle

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB
Inte...
Software
Qualifications

4. Ignore or accept the risk

Selecting the best option depends on the benefits and opportunities created by the option, as well as the cost and, potentially, any additional risks associated with the option. When a contingency plan is identified for a project risk, the best practice is to identify a trigger (which will determine when and how the contingency plan is invoked) and an owner (who will carry out the contingency plan).

2.4.6 Other Test Work Products

Testing involves the creation of a number of other work products, such as defect reports, test case specifications, and test logs. Most of these work products are produced by Test Analysts and Technical Test Analysts. In the Test Analyst syllabus, considerations for producing and documenting these work products are discussed. The Test Manager should ensure consistency and quality of these work products through the following activities:
- Establishing and monitoring metrics that monitor the quality of these work products, such as the percentage of rejected defect reports
- Working with the Test Analysts and Technical Test Analysts to select and customize appropriate templates for these work products
- Working with the Test Analysts and Technical Test Analysts to establish standards for these work products, such as the degree of detail necessary in tests, logs, and reports
- Reviewing testing work products using the appropriate techniques and by the appropriate participants and stakeholders

The extent, type, and specificity of test documentation can be influenced by the chosen software development lifecycle, applicable standards and regulations, and the product quality and project risks associated with the particular system being developed, among other considerations.

There are various sources of templates for testing work products, such as IEEE 829 [IEEE829]. It is important for the Test Manager to remember that the IEEE 829 documents are designed for use in any industry. As such, the templates contain a high level of detail that may or may not be applicable to a particular organization. It is a best practice to tailor the IEEE 829 documents to create standard templates for use in a particular organization. The consistent application of the templates reduces training requirements and helps to unify processes across an organization.

Testing also involves the creation of test results reports, which are typically produced by the Test Manager and are described later in this chapter.

2.5 Test Estimation

Estimation, as a management activity, is the creation of an approximate target for costs and completion dates associated with the activities involved in a particular operation or project. The best estimates:
- Represent the collective wisdom of experienced practitioners and have the support of the participants involved
- Provide specific, detailed catalogs of the costs, resources, tasks, and people involved
- Present, for each activity estimated, the most likely cost, effort and duration

Estimation of software and system engineering has long been known to be fraught with difficulties, both technical and political, though project management best practices for estimation are well established. Test estimation is the application of these best practices to the testing activities associated with a project or operation.

Test estimation should cover all activities involved in the test process as described in Chapter 1. The estimated cost, effort, and, especially, duration of test execution is often of the most interest to

management, as test execution is typically on the project's critical path. However, test execution estimates tend to be difficult to generate and unreliable when the overall quality of the software is low or unknown. In addition, familiarity and experience with the system will likely affect the quality of the estimate. A common practice is to estimate the number of test cases required during test execution, but this only works if one can assume the number of defects in the software to be tested is low. Assumptions made during estimation should always be documented as part of the estimation.

Test estimation should consider all factors that can influence the cost, effort, and duration of the testing activities. These factors include (but are not limited to) the following:
- Required level of quality of the system
- Size of the system to be tested
- Historical data from testing for previous test projects which may be augmented with industry data or benchmark data from other organizations
- Process factors, including the test strategy, development or maintenance lifecycle and process maturity, and the accuracy of the project estimate
- Material factors, including test automation and tools, test environment(s), test data, development environment(s), project documentation (e.g., requirements, designs, etc.), and reusable test work products
- People factors, including managers and technical leaders, executive and senior management commitment and expectations, skills, experience, and attitudes in the project team, stability of the project team, project team relationships, test and debugging environment support, availability of skilled contractors and consultants, and domain knowledge
- Complexity of the process, technology, organization, number of testing stakeholders, composition and location of sub teams
- Significant ramp up, training, and orientation needs
- Assimilation or development of new tools, technology, processes, techniques, custom hardware, or a large quantity of testware
- Requirements for a high degree of detailed test specification, especially to comply with an unfamiliar standard of documentation
- Complex timing of component arrival, especially for integration testing and test development
- Fragile test data (e.g., data that is time sensitive)

The quality of the software delivered for testing is also a major factor that Test Managers should consider in their estimation. For example, if the developers have embraced best practices such as automated unit testing and continuous integration, then as many as 50% of the defects will be removed prior to delivery of the code to the test team (see [Jones11] for more on defect removal effectiveness of these practices). Some have reported that Agile methodologies, including test-driven development, result in higher levels of quality being delivered for testing.

Estimation can be done either bottom-up or top-down. Techniques which can be used in test estimation, singly or in combination, include the following:
- Intuition, guesses, and past experience
- Work Breakdown Structures (WBS)
- Team estimation sessions (e.g., Wide Band Delphi)
- Company standards and norms
- Percentages of the overall project effort or staffing levels (e.g., tester-developer ratios)
- Organizational history and metrics, including metrics-derived models that estimate the number of defects, the number of test cycles, the number of test cases, each test's average effort, and the number of regression cycles involved
- Industry averages and predictive models such as function points, lines of code, estimated developer effort, or other project parameters (e.g., see [Jones07]).

In most cases, the estimate, once prepared, must be delivered to management, along with a justification (see Section 2.7). Frequently, some negotiation ensues, often resulting in a reworking of the estimate. Ideally, the final test estimate represents the best-possible balance of organizational and project goals in the areas of quality, schedule, budget, and features.

It is important to remember that any estimate is based on the information available at the time it was prepared. Early in a project, the information may be quite limited. In addition, information that is available early in a project may change over time. In order to remain accurate, estimates should be updated to reflect new and changed information.

2.6 Defining and Using Test Metrics

A management cliché says that what gets measured gets done. In addition, what does not get measured does not get done, because what does not get measured is easy to ignore. Therefore, it is important that the proper set of metrics be established for any endeavor, including testing.

Testing metrics can be classified as belonging to one or more of the following categories:
- Project metrics, which measure progress toward established project exit criteria, such as the percentage of test cases executed, passed, and failed
- Product metrics, which measure some attribute of the product, such as the extent to which it has been tested or the defect density
- Process metrics, which measure the capability of the testing or development process, such as the percentage of defects detected by testing
- People metrics, which measure the capability of individuals or groups, such as the implementation of test cases within a given schedule

Any given metric may belong in two, three, or even four categories. For example, a trend chart showing the daily arrival rate of defects can be associated with an exit criterion (zero new defects found for one week), the quality of the product (testing cannot locate further defects in it), and the capability of the test process (testing finds a large number of defects early in the test execution period).

People metrics are particularly sensitive. Managers sometimes mistake metrics which are primarily process metrics for people metrics, leading to disastrous results when people act to skew the metrics in a way which is favorable to them. The proper motivation and assessment of test people is discussed in Chapter 7 of this syllabus and in the Expert Test Management syllabus [ISTQB ETL SYL].

At the Advanced Level, we are concerned mostly with the use of metrics to measure the progress of testing, i.e., project metrics. Some of the project metrics used for test progress measurement also relate to the product and process. More information on the management use of product and process metrics is found in the Expert Test Management syllabus. More information on the use of process metrics is found in the Expert Improving the Test Process syllabus [ISTQB ITP SYL].

Using metrics enables testers to report results in a consistent way and enables coherent tracking of progress over time. Test Managers are frequently required to present metrics at various meetings which may be attended by multiple levels of stakeholders, ranging from technical staff to executive management. Because metrics are sometimes used to determine the overall success of a project, great care should be taken when determining what to track, how often to report it, and the method to be used to present the information. In particular, a Test Manager must consider the following:
- Definition of metrics. A limited set of useful metrics should be defined. Metrics should be defined based on specific objective(s) for the project, process, and/or product. Metrics should be defined for balance, as a single metric may give a misleading impression of status or trends. Once these metrics have been defined, their interpretation must be agreed upon by all

stakeholders in order to avoid confusion when the measurements are discussed. There is often a tendency to define too many metrics instead of concentrating on the most pertinent ones.

- Tracking of metrics. Reporting and merging metrics should be as automated as possible to reduce the time spent in taking and processing measurements. Variations of measurements over time for a specific metric may reflect information other than the interpretation agreed upon in the metric definition phase. The Test Manager should be prepared to carefully analyze possible divergence in measurements from expectations, and the reasons for that divergence.
- Reporting of metrics. The objective is to provide an immediate understanding of the information, for management purposes. Presentations may show a snapshot of a metric at a certain time or show the evolution of a metric over time so that trends can be evaluated.
- Validity of metrics. A Test Manager must also verify the information that is being reported. The measurements taken for a metric may not reflect the true status of a project or may convey an overly positive or negative trend. Before any data is presented, the Test Manager must review it for both accuracy and for the message that it is likely to convey.

There are five primary dimensions on which test progress is monitored:
- Product (quality) risks
- Defects
- Tests
- Coverage
- Confidence

Product risks, defects, tests, and coverage can be and often are measured and reported in specific ways during the project or operation. If these measurements are related to defined exit criteria in the test plan, they can provide an objective standard by which to judge completion of the test effort. Confidence is measurable through surveys or by using coverage as a surrogate metric; however, confidence is often reported subjectively as well.

Metrics related to product risks include:
- Percentage of risks completely covered by passing tests
- Percentage of risks for which some or all tests fail
- Percentage of risk not yet completely tested
- Percentage of risks covered, sorted by risk category
- Percentage of risks identified after the initial quality risk analysis

Metrics related to defects include:
- Cumulative number reported (found) versus cumulative number resolved (fixed)
- Mean time between failure or failure arrival rate
- Breakdown of the number or percentage of defects categorized by the following:
  - Particular test items or components
  - Root causes
  - Source of defect (e.g., requirement specification, new feature, regression, etc.)
  - Test releases
  - Phase introduced, detected, and removed
  - Priority/severity
  - Reports rejected or duplicated
- Trends in the lag time from defect reporting to resolution
- Number of defect fixes that introduced new defects (sometimes called daughter bugs)

Metrics related to tests include:
- Total number of tests planned, specified (implemented), run, passed, failed, blocked, and skipped

- Regression and confirmation test status, including trends and totals for regression test and confirmation test failures
- Hours of testing planned per day versus actual hours achieved
- Availability of the test environment (percentage of planned test hours when the test environment is usable by the test team)

Metrics related to test coverage include:
- Requirements and design elements coverage
- Risk coverage
- Environment/configuration coverage
- Code coverage

It is important that Test Managers understand how to interpret and use coverage metrics in order to understand and report test status. For higher levels of testing such as system testing, system integration testing, and acceptance testing, the primary test bases are usually work products such as requirements specifications, design specifications, use cases, user stories, product risks, supported environments, and supported configurations. Structural code coverage metrics apply more to lower levels of testing such as unit testing (e.g., statement and branch coverage) and component integration testing (e.g., interface coverage). While Test Managers may use code coverage metrics to measure the extent to which their tests exercise the structure of the system under test, reporting of higher level test results typically should not involve code coverage metrics. In addition, Test Managers should understand that, even if unit testing and component integration testing achieve 100% of their structural coverage goals, defects and quality risks remain to be addressed in higher levels of testing.

Metrics can also be linked to the activities of the fundamental test process (described in the Foundation syllabus and in this syllabus). By doing so, metrics can be used throughout the testing process to monitor the test process itself as well as progress toward project objectives.

Metrics to monitor test planning and control activities may include:
- Risk, requirements, and other test basis element coverage
- Defect discovery
- Planned versus actual hours to develop testware and execute test cases

Metrics to monitor test analysis activities may include:
- Number of test conditions identified
- Number of defects found during test analysis (e.g., by identifying risks or other test conditions using the test basis)

Metrics to monitor test design activities may include:
- Percentage of test conditions covered by test cases
- Number of defects found during test design (e.g., by developing tests against the test basis)

Metrics to monitor test implementation activities may include:
- Percentage of test environments configured
- Percentage of test data records loaded
- Percentage of test cases automated

Metrics to monitor test execution activities may include:
- Percentage of planned test cases executed, passed, and failed
- Percentage of test conditions covered by executed (and/or passed) test cases
- Planned versus actual defects reported/resolved
- Planned versus actual coverage achieved

Metrics to monitor test progress and completion activities will include a mapping to the milestones, entry criteria, and exit criteria (defined and approved in test planning), which may include the following:

- Number of test conditions, test cases or test specifications planned and those executed broken down by whether they passed or failed
- Total defects, often broken down by severity, priority, current state, affected subsystem, or other classification (see Chapter 4)
- Number of changes required, accepted, built, and tested
- Planned versus actual cost
- Planned versus actual duration
- Planned versus actual dates for testing milestones
- Planned versus actual dates for test-related project milestones (e.g., code freeze)
- Product (quality) risk status, often broken down by those mitigated versus unmitigated, major risk areas, new risks discovered after test analysis, etc.
- Percentage loss of test effort, cost, or time due to blocking events or planned changes
- Confirmation and regression test status

Metrics to monitor test closure activities may include:

- Percentage of test cases executed, passed, failed, blocked, and skipped during test execution
- Percentage of test cases checked into re-usable test case repository
- Percentage of test cases automated, or planned versus actual test cases automated
- Percentage of test cases integrated into the regression tests
- Percentage of defect reports resolved/not resolved
- Percentage of test work products identified and archived

In addition, standard project management techniques such as work breakdown structures are often used to monitor the test process. In Agile teams, testing is part of a user story's progress on the burn-down chart. Where Lean management techniques are used, testing progress on a story-by-story basis is often monitored by having the user story card move through a column on the Kanban board.

Given a defined set of metrics, measurements may be reported verbally in narrative form, numerically in tables, or pictorially in graphs. The measurements may be used for a number of purposes, including:

- Analysis, to discover what trends and causes may be discernible via the test results
- Reporting, to communicate test findings to interested project participants and stakeholders
- Control, to change the course of the testing or the project as a whole and to monitor the results of that course correction

Proper ways to gather, analyze, and report these test measures depend on the specific information needs, goals, and abilities of the people who will use the measurements. In addition, the specific content of test reports should vary based on the audience.

For purposes of test control, it is essential that metrics throughout the test process (once test planning is complete) provide the Test Manager with the information needed to guide the test effort toward successful completion of the test mission, strategies, and objectives. Therefore, planning must consider these information needs, and monitoring must include gathering any work product metrics needed. The volume of information needed and the effort expended to collect it depend on various project factors including size, complexity and risk.

Test control must respond to information generated by the testing as well as to changing conditions in which a project or endeavor exists. For example, if dynamic testing reveals defect clusters in areas that were deemed unlikely to contain many defects, or if the test execution period is shortened due to a delay in starting testing, the risk analysis and the plan must be revised. This could result in the re-prioritization of tests and re-allocation of the remaining test execution effort.

If divergence from the test plan is discovered via the test progress report, test control should be performed. Test control is aimed at redirecting the project and/or the testing in a more successful direction. When using test results to influence or measure control efforts on the project, the following options should be considered:

- Revising the quality risk analysis, test priorities, and/or test plans
- Adding resources or otherwise increasing the project or test effort
- Delaying the release date
- Relaxing or strengthening the test exit criteria
- Changing the scope (functional and/or non-functional) of the project

Implementing such options typically requires consensus among project or operation stakeholders and consent by project or operation managers.

The information delivered in a test report should depend largely on the information needs of the target audience, e.g., project management or business management. For a project manager, having detailed information on defects is likely to be of interest; for the business manager, the status of the product risks could be the key reporting issue.

2.7 Business Value of Testing

The Test Manager must work to optimize testing so that good business value is delivered. Testing excessively does not deliver good business value, because the testing will impose unreasonable delays and cost more than it saves. Testing too little does not deliver good business value because too many defects will be delivered to users. The optimum lies between those two extremes. The Test Manager must help testing stakeholders understand this optimum and the value delivered by testing.

While most organizations consider testing valuable in some sense, few managers, including Test Managers, can quantify, describe, or articulate that value. In addition, many Test Managers, test leads, and testers focus on the tactical details of testing (aspects specific to the task or test level), while ignoring the larger strategic (higher level) issues related to testing that other project participants, especially managers, care about.

Testing delivers value to the organization, project, and/or operation in both quantitative and qualitative ways:

- Quantitative values include finding defects that are prevented or fixed prior to release, finding defects that are known prior to release (not fixed but documented, perhaps with workarounds), reducing risk by running tests, and delivering information on project, process, and product status
- Qualitative values include improved reputation for quality, smoother and more-predictable releases, increased confidence, protection from legal liability, and reducing risk of loss of whole missions or even lives

Test Managers should understand which of these values apply for their organization, project, and/or operation, and be able to communicate about testing in terms of these values.

A well-established method for measuring the quantitative value and efficiency of testing is called cost of quality (or, sometimes, cost of poor quality). Cost of quality involves classifying project and operational costs into four categories related to product defect costs:

- Costs of prevention, e.g., training developers to write more maintainable or secure code
- Costs of detection, e.g., writing test cases, configuring test environments, and reviewing requirements
- Costs of internal failure, e.g., fixing defects detected during testing or reviews, prior to delivery

- Costs of external failure, e.g., support costs associated with defective software delivered to customers

A portion of the testing budget is a cost of detection (i.e., money that would be spent even if testers find no defects, such as money spent developing tests), while the remainder is a cost of internal failure (i.e., the actual cost associated with the found defects). The total costs of detection and internal failure are typically well below the costs of external failure, which makes testing an excellent value. By determining the costs in these four categories, Test Managers can create a convincing business case for testing.

More information on the business value of testing, including cost of quality, is found in [Black03].

2.8 Distributed, Outsourced, and Insourced Testing

In many cases, some or perhaps even all of the test effort is done by people in different locations, employed by different companies, or separated from the project team. If the test effort occurs at multiple locations, that test effort is distributed. If the test effort is carried out at one or more locations by people who are not employees of the company and who are not co-located with the project team, that test effort is outsourced. If the test effort is carried out by people who are co-located with the project team but who are not fellow employees, that test effort is insourced.

Common across all such test efforts is the need for clear channels of communication and well-defined expectations for missions, tasks, and deliverables. The project team must rely less on informal communication channels such as hallway conversations and colleagues spending social time together. It is important to have defined ways in which communication should occur, including addressing topics such as escalation of problems, types of information to be communicated, and methods of communication to be used. Everyone, on all sides of the team relationships, must have a clear understanding of their roles and responsibilities as well as the other parties' roles and responsibilities to avoid misunderstandings and unrealistic expectations. Location, time zone, cultural and language differences make communication and expectation issues even more likely to arise.

Also common across all such test efforts is the need for alignment of methodologies. While mis-alignment of methodologies can occur in any project, it is more likely to arise in situations where the work is distributed and/or performed by outside entities. If two testing groups use different methodologies or the testing group uses a different methodology than development or project management, this can result in significant problems, especially during test execution. For example, if a client is using Agile development, while the testing services provider has a pre-defined test methodology that assumes a sequential lifecycle, the timing and nature of delivery of test items to the testing services provider will be a point of contention.

For distributed testing, the division of the test work across the multiple locations must be explicit and intelligently decided. Without such guidance, the most competent group may not do the test work for which they are qualified. In addition, if each team doesn't know what they are responsible for, they may not do what they are supposed to do. Expectations for each of the teams must be clearly communicated. Without careful management, the test work as a whole may suffer from gaps (which increase residual quality risk on delivery) and overlap (which reduces efficiency).

Finally, for all such test efforts, it is critical that the entire project team develop and maintain trust that all of the test team(s) will carry out their roles properly in spite of organizational, cultural, language, and geographical boundaries. Lack of trust leads to inefficiencies and delays associated with verifying activities, apportioning blame for problems, and playing organizational politics.

2.9 Managing the Application of Industry Standards

In both the Foundation and Advanced Level syllabi, references are made to a number of standards. These referenced standards cover software development lifecycles, software testing, software quality characteristics, reviews, and defect management. Test Managers should be aware of standards, their organization's policy on the use of standards, and whether or not standards are required, necessary or useful for them to use.

Standards can come from different sources, such as:
- International or with international objectives
- National, such as national applications of international standards
- Domain specific, such as when international or national standards are adapted to particular domains, or developed for specific domains

International standards bodies include ISO and IEEE. ISO is the International Standards Organization, also called IOS, the International Organization for Standardization. It is made up of members representing, for their country, the national body most relevant to the area being standardized. This international body has promoted a number of standards useful for software testers, such as ISO 9126 (being replaced by ISO 25000), ISO 12207 [ISO12207], and ISO 15504 [ISO15504].

IEEE is the Institute of Electrical and Electronics Engineers, a professional organization based in the U.S., but with national representatives in more than one hundred countries. This organization has proposed a number of standards that are useful for software testers, such as IEEE 829 [IEEE829] and IEEE 1028 [IEEE1028].

Many countries have their own national standards. Some of these standards are useful for software testing. One example is the UK standard BS 7925-2 [BS7925-2], that provides information related to many of the test design techniques described in the Advanced Test Analyst and Advanced Technical Test Analyst syllabi.

Some standards are specific to particular domains, and some of these standards have implications for software testing, software quality and software development. For example, in the avionics domain, the U.S. Federal Aviation Administration's standard DO-178B (and its EU equivalent ED 12B) applies to software used in civilian aircraft. This standard prescribes certain levels of structural coverage criteria based on the level of criticality of the software being tested.

Another example of a domain-specific standard is found in medical systems with the U.S. Food and Drug Administration's Title 21 CFR Part 820 [FDA21]. This standard recommends certain structural and functional test techniques. The standard also recommends testing strategies and principles which are consistent with the ISTQB syllabi.

In some cases, testing is influenced by standards or widespread methodologies that are not primarily concerned with testing, but which strongly influence the software process context in which testing occurs. One example is the CMMI® software process improvement framework. It includes two key process areas, verification and validation, which are often interpreted as referring to levels of testing (such as system testing and acceptance testing, respectively). It also has implications in terms of the test strategy, often being interpreted as requiring that analytical requirements-based testing be included as part of the test strategy.

Three other important examples are PMI's PMBOK, PRINCE2®, and ITIL®. PMI and PRINCE2 are commonly used project management frameworks in North America and Europe, respectively. ITIL is a framework for ensuring that an IT group delivers valuable services to the organization in which it exists. The terminology and activities specified in these frameworks differ significantly from the ISTQB

syllabi and glossary. When working in an organization that uses PMI's PMBOK, PRINCE2, and/or ITIL, the Test Manager must understand the chosen frameworks, their implementations, and their terminology sufficiently well to work effectively in that context.

Whatever standards or methodologies are being adopted, it is important to remember that they are created by groups of professionals. Standards reflect the source group's collective experience and wisdom, but also their weaknesses, too. Test Managers should be aware of the standards that apply to their environment and context, whether formal standards (international, national or domain specific) or in-house standards and recommended practices.

When considering the use of multiple standards, keep in mind that some standards are inconsistent with other standards, or even provide conflicting definitions. The Test Manager should determine the usefulness of the different standards for the specific context in which testing is occurring. The information stated in a standard may be useful for a project, or may hinder it. However, standards can provide a reference to proven best practices, and provide a basis for organizing the test process.

In some cases, standards compliance is mandated and has implications for testing. The Test Manager must be aware of any such requirement to adhere to standards and ensure that adequate compliance is maintained.

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB

Inter...
Software...
Qualifications...

3. Reviews – 180 mins.

Keywords
audit, informal review, inspection, management review, moderator, review, review plan, reviewer, technical review, walkthrough

Learning Objectives for Reviews

3.2 Management Reviews and Audits
TM-3.2.1   (K2) Understand the key characteristics of management reviews and audits

3.3 Managing Reviews
TM-3.3.1   (K4) Analyze a project to select the appropriate review type and to define a plan for conducting reviews, in order to ensure proper execution, follow up, and accountability
TM-3.3.2   (K2) Understand the factors, skills, and time required for participation in reviews

3.4 Metrics for Reviews
TM-3.4.1   (K3) Define process and product metrics to be used in reviews

3.5 Managing Formal Reviews
TM-3.5.1   (K2) Explain, using examples, the characteristics of a formal review

3.1 Introduction

Reviews were introduced in the ISTQB Foundation Level syllabus as static testing activities for products. Audits and management reviews focus more on the software process rather than the software work products.

Since reviews are a form of static testing, Test Managers may be responsible for their overall success, particularly with respect to testware products. In the wider context of software projects, however, this responsibility should be a matter of organizational policy. Given the possible widespread application of formal reviews across many disciplines, both before and within software projects, the responsible party may be a Test Manager, or a Quality Assurance Manager, or a trained Review Coordinator. In this syllabus, the responsible party (whoever that is) is referred to as the review leader.

The review leader should ensure that an environment exists that is conducive to the implementation of the success factors as defined in the ISTQB Foundation Level syllabus. In addition, the review leader should devise a measurement plan to ensure that the reviews provide effective value.

Because testers have a strong understanding of the operational behavior and the required characteristics of the software system, tester involvement in the review process is important.

Participants in reviews should have review training to better understand their respective roles in any review process. All review participants must be committed to the benefits of a well-conducted review.

When done properly, reviews are the single biggest, and most cost-effective, contributor to overall delivered quality. It is thus of paramount importance that review leaders are able to implement efficient reviews in their projects and demonstrate the benefits of these reviews.

Possible reviews within a project include:
- Contractual reviews, initiated at project inception and at major project milestones
- Requirements reviews, initiated when the requirements are available for review, which ideally cover both functional and non-functional requirements
- Top level design reviews, initiated when the overall architectural design is available for review
- Detailed design reviews, initiated when the detailed design is available for review
- Code reviews, carried out as individual modules of software are created, which may include the unit tests and their results as well as the code itself
- Test work product reviews, which may cover the test plan(s), test conditions, quality risk analysis results, tests, test data, test environments, and test results
- Test entry (test readiness) reviews and test exit reviews for each test level, which respectively check the test entry criteria prior to starting test execution and the test exit criteria prior to concluding testing
- Acceptance reviews, used to obtain customer or stakeholder approval for a system

In addition to applying multiple review types to a product, it is important for a review leader to remember that while reviews can find defects in the static document, reviews should be augmented with other forms of static testing (such as static analysis) and dynamic testing of the code. Using a combination of these techniques improves test coverage and will locate more defects.

Different techniques have different focuses. For example, a review can eliminate an issue at the requirements level before the problem is implemented into the code. Static analysis can help to enforce coding standards and check for problems that might be too laborious for the team to find by examination of the work product. Inspections can not only lead to the discovery and removal of defects, but also can train authors in how to avoid creating defects in their work products.

The ISTQB Foundation Level syllabus introduced the following types of reviews:
- Informal review
- Walkthrough
- Technical review
- Inspection

In addition to these, Test Managers may also be involved in:
- Management reviews
- Audits

3.2 Management Reviews and Audits

Management reviews are used to monitor progress, assess status, and make decisions about future actions. These reviews support decisions about the future of the project, such as adapting the level of resources, implementing corrective actions or changing the scope of the project.

The following are key characteristics of management reviews:
- Conducted by or for managers having direct responsibility for the project or system
- Conducted by or for a stakeholder or decision maker, e.g., a higher level manager or director
- Check consistency with and deviations from plans
- Check adequacy of management procedures
- Assess project risks
- Evaluate impact of actions and ways to measure these impacts
- Produce lists of action items, issues to be resolved and decisions made

Management reviews of processes, such as project retrospectives (i.e., lessons learned), are an integral part of process improvement activities.

Test Managers should participate in and may initiate management reviews of testing progress.

Audits are usually performed to demonstrate conformance to a defined set of criteria, most likely an applicable standard, regulatory constraint, or a contractual obligation. As such, audits are intended to provide independent evaluation of compliance to processes, regulations, standards, etc. The following are key characteristics of audits:
- Conducted and moderated by a lead auditor
- Evidence of compliance collected through interviews, witnessing and examining documents
- Documented results include observations, recommendations, corrective actions and a pass/fail assessment

3.3 Managing Reviews

Reviews should be planned to take place at natural break points or milestones within the software project. Typically, reviews should be held after requirements and design definitions, with associated reviews starting with the business objectives and working down to the lowest level of design. Management reviews should take place at major project milestones, often as part of a verification activity before, during, and after test execution and other significant project phases. The review strategy must be coordinated with the test policy and the overall test strategy.

Before formulating an overall review plan at the project level, the review leader (who may be a Test Manager) should take into account:
- What should be reviewed (product and processes)
- Who should be involved in specific reviews
- Which relevant risk factors to cover

Early in the project planning phase, the review leader should identify the items to be reviewed and select the appropriate review type (informal review, walkthrough, technical review or inspection, or a mixture of two or three types) and level of formality. This is the point at which additional review training might be recommended. From there, a budget (time and resources) for the review process can be allocated. Determination of the budget should include a risk evaluation and return on investment computation.

The return on investment for reviews is the difference between the cost of conducting the review, and the cost of dealing with the same defects at a later stage (or missing them altogether) if the review had not been done. The Cost of Quality calculation explained in Section 2.7 can be used to help determine this number.

Determining the optimal time to perform reviews depends on the following:
- The availability of the items to review in a sufficiently final format
- The availability of the right personnel for the review
- The time when the final version of the item should be available
- The time required for the review process of that specific item

Adequate metrics for review evaluation should be defined by the review leader during test planning. If inspections are used, then brief inspections should be conducted at the author's request, as document fragments are completed (e.g., individual requirements or sections).

The objectives of the review process must be defined during test planning. This includes conducting effective and efficient reviews and reaching consensus decisions regarding review feedback.

Project reviews are frequently held for the overall system and may also be necessary for subsystems and even individual software elements. The number of reviews, the type of reviews, the organization of the reviews, and the people involved all depend on project size and complexity, and on product risks.

In order to be efficient, participants in reviews must have the appropriate level of knowledge, both technical and procedural. Thoroughness and attention to detail are some of the other required skills for the reviewers, in order to have effective reviews. Clarity and correct prioritization are attributes to look for in good review comments. The need for procedural knowledge may mean that some training is necessary to ensure that reviewers understand their roles and responsibilities in the review process.

Review planning should address the risks associated with technical factors, organizational factors and people issues when performing reviews. The availability of reviewers with sufficient technical knowledge is critical to a successful review. All teams within the project should be involved in planning for the reviews, which should ensure that each team is committed to the success of the review process. Planning must ensure that each organization is allocating sufficient time for required reviewers to prepare for, and participate in, the reviews at appropriate points in the project schedule. Time should also be planned for any required technical or process training for the reviewers. Backup reviewers should be identified in case key reviewers become unavailable due to changes in personal or business plans.

During the actual execution of formal reviews, the review leader must ensure that:
- Adequate measurements are provided by the participants in the reviews to allow evaluation of review efficiency
- Checklists are created, and maintained to improve future reviews
- Defect severity and priority evaluation are defined for use in defect management of issues found during reviews (see Chapter 4)

After each review, the review leader should:

- Collect the review metrics and ensure that the issues identified are sufficiently resolved to meet the specific test objectives for the review
- Use the review metrics as input when determining the return on investment (ROI) for reviews
- Provide feedback information to the relevant stakeholders
- Provide feedback to review participants

To evaluate the effectiveness of reviews, Test Managers can compare actual results found in subsequent testing (i.e., after the reviews) with the results from the review reports. In the case where a work product is reviewed, approved based on the review, but later found defective, the review leader should consider ways in which the review process might have allowed the defects to escape. Likely causes include problems with the review process (e.g., poor entry/exit criteria), improper composition of the review team, inadequate review tools (checklists, etc.), insufficient reviewer training and experience, and too little preparation and review meeting time.

A pattern of escaped defects (especially major defects), repeated across several projects, indicates that there are significant problems with the conduct of reviews. In such a situation, the review leader needs to review the process and take appropriate action. It's also possible that, for various reasons, reviews may lose their effectiveness over time. Such an effect will be revealed in project retrospectives by reduced defect detection effectiveness for the reviews. Here again, the review leader must investigate and fix the causes. In any case, review metrics should not be used to punish or reward individual reviewers or authors, but should focus on the review process itself.

3.4 Metrics for Reviews

Review leaders (who, as mentioned in the previous sections, may be Test Managers) must ensure that metrics are available to:

- Evaluate the quality of the reviewed item
- Evaluate the cost of conducting the review
- Evaluate the downstream benefit of having conducted the review

Review leaders can use the measurements to determine the return on investment and efficiency of the reviews. These metrics can also be used for reporting and for process improvement activities.

For each work product reviewed, the following metrics can be measured and reported for product evaluation:

- Work-product size (pages, lines of code, etc.)
- Preparation time (prior to the review)
- Time to conduct the review
- Rework time to fix defects
- Duration of the review process
- Number of defects found and their severity
- Identification of defect clusters within the work product (i.e., areas that have a higher defect density)
- Type of review (informal review, walkthrough, technical review or inspection)
- Average defect density (e.g., defects per page or per thousand lines of code)
- Estimated residual defects (or residual defect density)

For each review the following metrics can be measured and reported for process evaluation:

- Defect detection effectiveness (taking into account defects found later in the lifecycle)
- Improvement of review process effort and timing
- Percent coverage of planned work products
- Types of defects found and their severity

- Participant surveys about effectiveness and efficiency of the review process
- Cost of quality metrics for review defects versus dynamic test defects and production defects
- Correlation of review effectiveness (review type versus defect detection effectiveness)
- Number of reviewers
- Defects found per work-hour expended
- Estimated project time saved
- Average defect effort (i.e., the total detection and fix time divided by the number of defects)

In addition, the metrics mentioned for product evaluation above are also useful in process evaluation.

3.5 Managing Formal Reviews

The ISTQB Foundation Level syllabus describes the different phases of a formal review: planning, kick-off, individual preparation, review meeting, rework and follow-up. To correctly implement formal reviews, review leaders need to ensure that all steps in the review process are followed.

Formal reviews have a number of characteristics such as:
- Defined entry and exit criteria
- Checklists to be used by the reviewers
- Deliverables such as reports, evaluation sheets or other review summary sheets
- Metrics for reporting on the review effectiveness, efficiency, and progress

Prior to initiating a formal review, fulfillment of the review prerequisites (as defined in the procedure or in the list of entry criteria) should be confirmed by the review leader.

If the prerequisite conditions for the formal review are not fulfilled, the review leader may propose one of the following to the review authority for final decision:
- Redefinition of the review with revised objectives
- Corrective actions necessary for the review to proceed
- Postponement of the review

As part of controlling a formal review, these reviews are monitored in the context of the overall (higher level) program, and are associated with project quality assurance activities. Control of formal reviews includes feedback information using product and process metrics.

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB

Inter...
Software
Qualifications

4. Defect Management – 150 mins.

Keywords

anomaly, defect, defect triage committee, failure, false-negative result, false-positive result, phase containment, priority, root cause, severity

Learning Objectives for Defect Management

4.2 The Defect Lifecycle and the Software Development Lifecycle

TM-4.2.1   (K3) Develop a defect management process for a testing organization, including the defect report workflow, that can be used to monitor and control a project's defects throughout the testing lifecycle

TM-4.2.2   (K2) Explain the process and participants required for effective defect management.

4.3 Defect Report Information

TM-4.3.1   (K3) Define the data and classification information that should be gathered during the defect management process

4.4 Assessing Process Capability with Defect Report Information

TM-4.4.1   (K2) Explain how defect report statistics can be used to evaluate the process capability of the testing and software development processes

4.1 Introduction

An organization's defect management process and the tool used to manage this work are of critical importance not only to the test team but to all teams involved in the development of software. Information gathered by the effective management of defects allows the Test Manager and other project stakeholders to gain insight on the state of a project throughout the development lifecycle, and by collecting and analyzing data over time, can help locate areas of potential improvement for testing and development processes.

In addition to understanding the overall defect lifecycle and how it is used to monitor and control both the testing and software development processes, the Test Manager must also be familiar with what data is critical to capture and must be an advocate of proper usage of both the process and the selected defect management tool.

4.2 The Defect Lifecycle and the Software Development Lifecycle

As explained in the Foundation Level syllabus, defects are introduced when a person makes a mistake during the creation of a work product. This work product can be a requirements specification, a user story, a technical document, a test case, the program code, or any other work product produced during a software development or maintenance process.

Defects may be introduced at any point in the software development lifecycle and in any software-related work product. Therefore, each phase of the software development lifecycle should include activities to detect and remove potential defects. For example, static testing techniques (i.e., reviews and static analysis) can be used on design specifications, requirements specifications, and code prior to delivering those work products to subsequent activities. The earlier each defect is detected and removed, the lower the overall cost of quality for the system; cost of quality for a given level of defects is minimized when each defect is removed in the same phase in which it was introduced (i.e., when the software process achieves perfect phase containment). Furthermore, as explained in the Foundation Level syllabus, static testing finds defects directly, rather than finding failures, and thus the cost of removing the defect is lower because debugging activities are not required to isolate the defect.

During dynamic testing activities such as unit testing, integration testing, and system testing, the presence of a defect is revealed when it causes a failure, which results in a discrepancy between the actual results and the expected results of a test (i.e., an anomaly). In some cases, a false-negative result occurs when the tester does not observe the anomaly. If the tester does observe the anomaly, a situation has occurred which requires further investigation. This investigation starts with the filing of a defect report.

In Test Driven Development, automated unit tests are used as a form of executable design specifications. As code is developed, it is immediately exercised using these tests. Until the development of the unit is complete, some or all of the tests will fail. Therefore, the failure of such a test does not constitute a defect, and is typically not tracked.

4.2.1 Defect Workflow and States

Most testing organizations use a tool to manage defect reports through the defect lifecycle. A defect report typically progresses through a workflow and moves through a sequence of states as it continues through the defect lifecycle. In most of these states, a single defect lifecycle participant owns the report and is responsible for carrying out a task which, when completed, will cause the defect report to be moved to the next state (and assigned to the next responsible party). In terminal states, such as when the defect report is closed (usually meaning that the underlying defect is fixed

and the fix verified through a confirmation test), cancelled (usually meaning that the defect report is invalid), irreproducible (usually meaning that the anomaly can no longer be observed), or deferred (usually meaning that the anomaly relates to a real defect, but that defect will not be fixed during the project), the report does not have an owner, because no further actions are required.

For defects discovered by testers during testing, there are three states in particular where the action resides with the test team:

- The initial state
  - o In this state, one or more testers gather the information necessary for the person responsible for resolving the defect to reproduce the anomaly (see Section 4.3 for more on the information to be included in the defect report).
  - o This may also be referred to as the "open" or "new" state.
- The returned state
  - o In this state, the receiver of the report has rejected the report or is asking the tester to supply further information. This state may indicate a shortfall in the initial information-gathering process or of the testing itself, and Test Managers should monitor for excessive rates of return. The tester(s) must provide the additional information, or confirm that the report indeed should be rejected.
  - o This may also be referred to as the "rejected" or "clarification" state.
- The confirmation test state
  - o In this state, the tester will run a confirmation test (often following the steps to reproduce the failure from the defect report itself) to determine whether the fix has indeed solved the problem. If the confirmation test indicates that the defect is repaired, the tester should close the report. If the confirmation test indicates that the defect is not repaired, the tester should re-open the report, causing it to be reassigned to the previous owner, who can then complete the work necessary to repair the defect.
  - o This may also be referred to as the "resolved" or "verification" state.

4.2.2 Managing Invalid and Duplicate Defect Reports

In some cases, an anomaly occurs not as the symptom of a defect, but rather due to a problem with the test environment, the test data, some other element of the testware, or the tester's own misunderstanding. If the tester opens a defect report that subsequently is found not to relate to a defect in the work product under test, that is a false-positive result. Such reports are typically cancelled or closed as invalid defect reports. In addition, in some cases a defect can exhibit different symptoms which may appear to the tester(s) as being entirely unrelated. If two or more defect reports are filed which subsequently are found to relate to the same root cause, one of the defect reports is typically retained while the others are closed as duplicate defect reports.

While invalid and duplicate defect reports represent a certain level of inefficiency, some amount of such reports is inevitable and should be accepted as such by the Test Manager. When managers attempt to eliminate all invalid and duplicate defect reports, the number of false-negatives typically increases, since testers are being discouraged from filing defect reports. This decreases the testing organization's defect detection effectiveness, which is related to a key testing organization objective in most cases.

4.2.3 Cross-Functional Defect Management

Although the testing organization and Test Manager typically own the overall defect management process and the defect management tool, a cross-functional team is generally responsible for managing the reported defects for a given project. In addition to the Test Manager, participants in the defect management (or defect triage) committee typically include development, project management, product management and other stakeholders who have an interest in the software under development.

As anomalies are discovered and entered into the defect management tool, the defect management committee should meet to determine whether each defect report represents a valid defect, and whether it should be fixed or deferred. This decision requires the defect management committee to consider the benefits, risks and costs associated with fixing or not fixing the defect. If the defect is to be fixed, the team should establish the priority of fixing the defect relative to other project tasks. The Test Manager and test team may be consulted regarding the relative importance of a defect and should provide available objective information.

A defect tracking tool should not be used as a substitute for good communication nor should defect management committee meetings be used as a substitute for effective use of a good defect tracking tool. Communication, adequate tool support, a well-defined defect lifecycle, and an engaged defect management committee are all necessary for effective and efficient defect management.

4.3 Defect Report Information

When a defect is detected (as part of static testing), or a failure is observed (as part of dynamic testing), data should be gathered by the person(s) involved and included in the defect report. This information should suffice for three purposes:
- Management of the report through the defect lifecycle
- Assessment of project status, especially in terms of product quality, and test progress
- Assessment of process capability (as discussed in Section 4.4 below)

The data needed for defect report management and project status can vary depending on when the defect is detected in the lifecycle, typically with less information needed earlier (e.g., requirements reviews and unit test). However, the core information gathered should be consistent across the lifecycle and ideally across all projects to allow for meaningful comparison of process defect data throughout the project and across all projects.

The collection of defect data can assist in test progress monitoring, control, and exit criteria evaluation. For example, defect information should support defect density analysis, trend analysis of defects detected and resolved, average time from defect detection to resolution, and failure intensity (e.g., MTBF analysis).

Defect data to be collected may include the following:
- The name of the person who discovered the defect
- The role of the person (e.g., end user, business analyst, developer, technical support person)
- The type of testing being performed (e.g., usability testing, performance testing, regression testing)
- A summary of the problem
- A detailed description of the problem
- Steps to reproduce the failure (for a defect), along with the actual and expected results (highlighting the anomaly), including screen shots, database dumps, and logs where applicable
- The lifecycle phase of introduction, detection, and removal for the defect, including the test level if applicable
- The work product in which the defect was introduced
- The severity of the impact on the system and/or the product stakeholders (usually determined by the technical behavior of the system)
- The priority to fix the problem (usually determined by the business impact of the failure)
- The subsystem or component in which the defect lies (for defect cluster analysis)
- The project activity occurring when the problem was detected
- The identification method which revealed the problem (e.g., review, static analysis, dynamic testing, production use)

- The type of defect (usually corresponding to a defect taxonomy where used)
- The quality characteristic affected by the defect
- The test environment in which the defect was observed (for dynamic testing)
- The project and product in which the problem exists
- The current owner; i.e., the person currently assigned to work on the problem, assuming the report is not in a final state
- The current state of the report (usually managed by the defect tracking tool as part of the lifecycle)
- The specific work products (e.g., test items and their release numbers) in which the problem was observed, along with the specific work products in which the problem was ultimately resolved
- The impact on project and product stakeholders' interests
- Conclusions, recommendations and approvals for the action taken or not taken to resolve the problem
- Risks, costs, opportunities, and benefits associated with fixing or not fixing the defect
- The dates on which various defect lifecycle transitions occurred, the owners of the report based on each transition, and the actions taken by project team members to isolate, repair, and verify the defect fix
- A description of how the defect was ultimately resolved and recommendations for testing the fix (if the defect was resolved by a change to the software)
- Other references, such as the test that revealed the defect and the risk, requirement, or other test basis element related to the defect (for dynamic testing)

Various standards and documents, such as ISO 9126 [ISO9126] (being replaced by ISO 25000), IEEE 829 [IEEE829], IEEE 1044 [IEEE1044], and Orthogonal Defect Classification exist to help the Test Manager determine which information to gather for defect reporting.

Whatever the specific information determined as necessary for defect reports, it is critical that testers enter information that is complete, concise, accurate, objective, relevant and timely. Even when manual intervention and face to face communication overcome problems with defect report data in terms of resolving an individual defect, problems with defect report data can pose insurmountable obstacles to proper assessment of project status, test progress, and process capability.

4.4 Assessing Process Capability with Defect Report Information

As discussed in Chapter 2, defect reports can be useful for project status monitoring and reporting. While the process implications of metrics are primarily addressed in the Expert Test Management syllabus [ISTQB ETM SYL], at the Advanced Level, Test Managers should be aware of what defect reports mean in terms of assessing the capability of the testing and software development processes.

In addition to the test progress monitoring information mentioned in Chapter 2 and in Section 4.3, defect information needs to support process improvement initiatives. Examples include:
- Using the phase of introduction, detection, and removal information, on a phase-by-phase basis, to assess phase containment and suggest ways to improve defect detection effectiveness in each phase
- Using the phase of introduction information for Pareto analysis of the phases in which the largest number of defects are introduced, to enable targeted improvements to reduce the total number of defects
- Using the defect root cause information to determine the underlying reasons for defect introduction, to enable process improvements that reduce the total number of defects
- Using the phase of introduction, detection, and removal information to perform cost of quality analysis, to minimize the cost associated with defects

Certified Tester

Advanced Level Syllabus - Test Manager

ISTQB

Inter...
Software
Qualifications

- Using defect component information to perform defect cluster analysis, to better understand technical risks (for risk-based testing) and to enable re-engineering of troublesome components

The use of metrics to assess the effectiveness and efficiency of the test process is discussed in the Expert Test Management syllabus [ISTQB ETM SYL].

In some cases, teams may elect not to track defects found during some or all of the software development lifecycle. While this is often done in the name of efficiency and for the sake of reducing process overhead, in reality it greatly reduces visibility into the process capabilities of testing and software development. This makes the improvements suggested above difficult to carry out due to a lack of reliable data.

5. Improving the Testing Process – 135 mins.

Keywords
Capability Maturity Model Integration (CMMI), Critical Testing Processes (CTP), Systematic Test and Evaluation Process (STEP), Test Maturity Model integration (TMMi), TPI Next

Learning Objectives for Improving the Testing Process

5.2 Test Improvement Process
TM-5.2.1   (K2) Explain, using examples, why it is important to improve the test process

5.3 Improving the Test Process
TM-5.3.1   (K3) Define a test process improvement plan using the IDEAL model

5.4 Improving the Test Process with TMMi
TM-5.4.1   (K2) Summarize the background, scope and objectives of the TMMi test process improvement model

5.5 Improving the Test Process with TPI Next
TM-5.5.1   (K2) Summarize the background, scope and objectives of the TPI Next test process improvement model

5.6 Improving the Test Process with CTP
TM-5.6.1   (K2) Summarize the background, scope and objectives of the CTP test process improvement model

5.7 Improving the Test Process with STEP
TM-5.7.1   (K2) Summarize the background, scope and objectives of the STEP test process improvement model

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB
Inter...
Software...
Qualifications...

5.1 Introduction

Once established, an organization's overall test process should undergo continuous improvement. In this chapter, generic improvement issues are first covered, followed by an introduction to some specific models which can be used for test process improvement. Test Managers should assume they will be the driving force behind test process changes and improvements and so should be familiar with the industry-accepted techniques discussed in this chapter. Further information on test improvement processes is discussed in the Expert Level Improving the Test Process syllabus.

5.2 Test Improvement Process

Just as organizations use testing to improve software, process improvement techniques can be selected and used to improve the process of developing software and the resulting software deliverables. Process improvement can also be applied to the testing processes. Different ways and methods are available to improve the testing of software and of systems containing software. These methods aim at improving the process, and hence the deliverables, by providing guidelines and areas for improvement.

Testing often accounts for a major part of the total project costs. However, only limited attention is given to the test process in the various software process improvement models, such as CMMI® (see below for details).

Test improvement models such as the Test Maturity Model integration (TMMi®), Systematic Test and Evaluation Process (STEP), Critical Testing Processes (CTP) and TPI Next® were developed to address the lack of attention to testing in most software process improvement models. Properly used, these models can provide a degree of cross-organization metrics that can be used for benchmark comparisons.

The models presented in this syllabus are not intended to be a recommendation for use, but are presented here to provide a representative view of how the models work and what they include.

5.2.1 Introduction to Process Improvement

Process improvements are relevant to the software development process as well as to the testing process. Learning from one's own mistakes makes it possible to improve the process that organizations are using to develop and test software. The Deming improvement cycle: Plan, Do, Check, Act, has been used for many decades, and is still relevant when testers need to improve the process in use today.

One premise for process improvement is the belief that the quality of a system is highly influenced by the quality of the process used to develop the software. Improved quality in the software industry reduces the need for resources to maintain the software and thus provides more time for creating more and better solutions in the future. Process models provide a place to start improving, by measuring the organization's process capabilities against the model. The models also provide a framework for improving the organization's processes based on the outcome of an assessment.

A process assessment leads to a process capability determination, which motivates a process improvement. This may invoke a subsequent process assessment to measure the effect of the improvement.

5.2.2 Types of Process Improvement

The use of assessment models is a common method which ensures a standardized approach to improving test processes using tried and trusted practices.

Process improvement models are categorized into two types:
1. The process reference model which provides a maturity measurement as part of the assessment in order to evaluate an organization's capability compared with the model, to evaluate the organization within the framework, and to provide a roadmap for improving the process.
2. The content reference model which provides business-driven evaluations of an organization's opportunities to improve including, in some cases, benchmarking against industry averages using objective measurements. This evaluation can be used to create a roadmap for improving the process.

Test process improvement can also be accomplished without models by using, for example, analytical approaches and retrospective meetings.

5.3 Improving the Testing Process

The IT industry can work with test process improvement models to reach a higher level of maturity and professionalism. Industry standard models are helping to develop cross-organization metrics and measures that can be used for comparison. Out of the need for process improvement in the testing industry, several sets of recommended processes have materialized. These include STEP, TMMi, TPI Next and CTP. The staged models, such as TMMi and CMMI, provide standards for comparison across different companies and organizations. The continuous models, such as CTP, STEP and TPI Next, allow an organization to address its highest priority issues with more freedom in the order of implementation. These are each discussed further in this section.

All of these models allow an organization to determine where it stands in terms of its current test processes. Once an assessment is performed, TMMi and TPI Next suggest a roadmap for improving the test process. Alternatively, STEP and CTP provide the organization with means to determine where its greatest process improvement return on investment will come from and leave it to the organization to select the appropriate roadmap.

Once it has been agreed that test processes should be reviewed and improved, the process improvement implementation steps to be adopted for this activity could be as defined in the IDEAL[SM] model [IDEAL96]:
- Initiating the improvement process
- Diagnosing the current situation
- Establishing a test process improvement plan
- Acting to implement improvement
- Learning from the improvement program

Initiating the improvement process
Before the process improvement activities start, the objectives, goals, scope and coverage of the process improvements are agreed on by the stakeholders. The choice of the process improvement model is also made at this time. The model may either be selected from publically available options (such as CTP, STEP, TMMi, and TPI Next) or developed internally. In addition, success criteria should be defined and a method by which they will be measured throughout the improvement activity should be determined.

Diagnosing the current situation

The agreed assessment approach is undertaken and a test assessment report is created which contains an appraisal of current testing practice and a list of possible process improvements.

Establishing a test process improvement plan

The list of possible process improvements is prioritized. The prioritization could be based on return on investment, risks, alignment with organizational strategy, and/or measurable quantitative or qualitative benefits. Having established the priority order, a plan for the delivery of the improvements is developed.

Acting to implement improvement

The test process improvement plan for the delivery of the improvements is implemented. This could include any training or mentoring required, piloting of processes and ultimately their full deployment.

Learning from the improvement program

Having fully deployed the process improvements, it is essential to verify which benefits (of those established earlier in addition to potentially unexpected benefits) were received. It is also important to check which of the success criteria for the process improvement activity have been met.

Depending on the process model used, this stage of the process is where monitoring of the next level of maturity starts and a decision is made to either start the improvement process again, or to stop the activity at this point.

5.4 Improving the Testing Process with TMMi

The Testing Maturity Model integration (TMMi) is composed of five maturity levels and is intended to complement CMMI. Each of the maturity levels contains defined process areas that must be 85% complete by achieving specific and generic goals before the organization can advance to the next level.

The TMMi maturity levels are:
- Level 1: Initial
  The initial level represents a state where there is no formally documented or structured testing process. Tests are typically developed in an ad hoc way after coding, and testing is seen as the same as debugging. The aim of testing is understood to be proving that the software works.
- Level 2: Managed
  The second level is attained when testing processes are clearly separated from debugging. It can be reached by setting testing policy and goals, introducing the steps found in a fundamental test process (e.g., test planning), and implementing basic testing techniques and methods.
- Level 3: Defined
  The third level is reached when a testing process is integrated into the software development lifecycle, and documented in formal standards, procedures, and methods. Reviews take place and there should be a distinct software testing function that can be controlled and monitored.
- Level 4: Measured
  Level four is achieved when the testing process is capable of being effectively measured and managed at an organizational level to the benefit of specific projects.
- Level 5: Optimized
  The final level represents a state of test process maturity where data from the testing process can be used to help prevent defects, and the focus is on optimizing the established process.

For more information on TMMi, see [vanVeenendaal11] and [www.tmmi.org].

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB

Inter...
Software
Qualifications

5.5 Improving the Testing Process with TPI Next

The TPI Next model defines 16 key areas, each of which covers a specific aspect of the test process, such as test strategy, metrics, test tools and test environment.

Four maturity levels are defined in the model:
- Initial
- Controlled
- Efficient
- Optimizing

Specific checkpoints are defined to assess each key area at each of the maturity levels. Findings are summarized and visualized by means of a maturity matrix which covers all key areas. The definition of improvement objectives and their implementation can be tailored according to the needs and capacity of the testing organization.

The generic approach makes TPI Next independent of any software process improvement model. It covers both the test engineering aspects as well as support for managerial decision making [deVries09].

For more information on TPI Next, see [www.tpinext.com].

5.6 Improving the Testing Process with CTP

The basic premise of the Critical Testing Processes (CTP) assessment model is that certain testing processes are critical. These critical processes, if carried out well, will support successful test teams. Conversely, if these activities are carried out poorly, even talented individual testers and Test Managers are unlikely to be successful. The model identifies twelve critical testing processes. CTP is primarily a content reference model.

The CTP model is a context-sensitive approach that allows for tailoring the model including:
- Identification of specific challenges
- Recognition of attributes of good processes
- Selection of the order and importance of implementation of process improvements

The CTP model is adaptable within the context of all software development lifecycle models.

In addition to participant interviews, the CTP model includes the use of metrics to benchmark organizations against industry averages and best practices.

For more information on CTP, see [Black03].

5.7 Improving the Testing Process with STEP

STEP (Systematic Test and Evaluation Process), like CTP and unlike TMMi and TPI Next, does not require that improvements occur in a specific order.

STEP is primarily a content reference model which is based upon the idea that testing is a lifecycle activity that begins during requirements formulation and continues until retirement of the system. The STEP methodology stresses "test then code" by using a requirements-based testing strategy to ensure that early creation of test cases validates the requirements specification prior to design and coding.

Basic premises of the methodology include:
- A requirements-based testing strategy
- Testing starts at the beginning of the lifecycle
- Tests are used as requirements and usage models
- Testware design leads software design
- Defects are detected earlier or prevented altogether
- Defects are systematically analyzed
- Testers and developers work together

In some cases the STEP assessment model is blended with the TPI Next maturity model.

For more information on STEP, see [Craig02].

ISTQB

Inter...
Software...
Qualifications...

6. Test Tools and Automation – 135 min

Keywords
open-source tool, custom tool

Learning Objectives for Test Tools and Automation

6.2 Tool Selection
TM-6.2.1   (K2) Describe management issues when selecting an open-source tool
TM-6.2.2   (K2) Describe management issues when deciding on a custom tool
TM-6.2.3   (K4) Assess a given situation in order to devise a plan for tool selection, including risks, costs and benefits

6.3 Tool Lifecycle
TM-6.3.1   (K2) Explain the different phases in the lifecycle of a tool

6.4 Tool Metrics
TM-6.4.1   (K2) Describe how metric collection and evaluation can be improved by using tools

6.1 Introduction

This section expands on the Foundation Level syllabus by covering a number of general concepts that the Test Manager must consider regarding tools and automation.

6.2 Tool Selection

There are a variety of different issues that a Test Manager must consider when selecting test tools.

The most common choice, historically, is to purchase a tool from a commercial vendor. In some cases, that may be the only viable choice. However, there are other possibilities, such as open-source tools and custom tools, which are also feasible options.

Regardless of the type of tool, a Test Manager must be careful to investigate the total cost of ownership throughout the expected life of the tool by performing a cost-benefit analysis. This topic is covered below in the return on investment (ROI) section.

6.2.1 Open-Source Tools

Open-source tools are available for almost any facet of the testing process, from test case management to defect tracking to test case automation, just to name a few. An important distinction of open-source tools is that while the tool itself generally does not have a high initial purchase cost, there may not be any formal support available for the tool. Many open-source tools, however, do have a devoted following that is willing to supply non-traditional or informal support for users.

In addition, many open-source tools were originally created to solve a specific problem or address a single issue; therefore, the tool may not perform all of the functions of a similar vendor tool. Because of this, careful analysis of the actual needs of the testing group should be performed prior to selecting an open-source tool.

One benefit of using open-source tools is that they usually can be modified or extended by their users. If the testing organization has the core competencies, the tool can be modified to work with other tools or changed to fit the test team's needs. Multiple tools can be combined to solve problems that vendor tools cannot address. Of course, the more tools used and the more modifications made, the more complexity and overhead added. A Test Manager must ensure that the team does not start using open-source tools just for the sake of using them; as with other tools, the effort must always be targeted at deriving a positive ROI.

A Test Manager must understand the licensing scheme of the selected tool. Many open-source tools come with a variation of the GNU General Public License which specifies that the distribution of the software must always be under the same terms as it was received. Should the test team make changes in the tool to better support their testing, these changes may need to be made available to all external users of the tool under the tool license. A Test Manager should check into the legal ramifications of redistributing the software for their organization.

Organizations which develop safety-critical or mission-critical software, or which are subject to regulatory compliance, may have issues with using open-source tools. While many open-source tools are of very high quality, the accuracy of any given open-source tool has likely not been certified. Vendor tools are often certified as to their accuracy and fitness for a particular task (e.g., DO-178B). While the open-source tool may be just as good, the certification may be the responsibility of the group using it, creating extra overhead.

6.2.2 Custom Tools

The testing organization may find that they have a specific need for which no vendor or open-source tool is available. Reasons may be a proprietary hardware platform, a customized environment or a process that has been modified in an unusual way. In such cases, if the core competence exists on the team, the Test Manager may wish to consider developing a custom tool.

The benefits of developing a custom tool are that the tool can meet the team's needs precisely and can operate efficiently in the context the team requires. The tool can be written to interface with other tools in use and to generate data in the exact form needed by the team. In addition, the tool may have applications in the organization beyond the immediate project. However, before planning on releasing the tool for other projects, it is important to review the purpose, aim, benefits and possible downsides first.

A Test Manager considering creating custom tools must also consider the possible negative issues. Custom tools are often dependent on the person creating the tool. Therefore, custom tools must be adequately documented so that they can be maintained by others. If this is not done, they may be orphaned and fall into disuse when the tool creator leaves the project. Over time, custom tools may have their scope expanded beyond the initial intent, which may cause quality problems in the tool, leading to false-positive defect reports or creation of inaccurate data. The Test Manager must keep in mind that a custom tool is just another software product, and as such, is subject to the same sorts of development issues as any other software product. Custom tools should be designed and tested to ensure that they work as expected.

6.2.3 Return on Investment (ROI)

It is the responsibility of the Test Manager to ensure that all tools introduced into the testing organization add value to the team's work and can show a positive ROI to the organization. To ensure that a tool will achieve real and lasting benefits, a cost-benefit analysis should be performed before acquiring or building a tool. In this analysis, the ROI should consider both recurring and non-recurring costs, some of which are monetary and some of which are resource or time costs, and the risks that may reduce the value of the tool.

Non-recurring costs include the following:
- Defining tool requirements to meet the objectives and goals
- Evaluating and selecting the correct tool and tool vendor
- Purchasing, adapting or developing the tool
- Performing the initial training for the tool
- Integrating the tool with other tools
- Procuring the hardware/software needed to support the tool

Recurring costs include the following:
- Owning the tool
  - Licensing and support fees
  - Maintenance costs for the tool itself
  - Maintenance of artifacts created by the tool
  - Ongoing training and mentoring costs
- Porting the tool to different environments
- Adapting the tool to future needs
- Improving the quality and processes to ensure optimal use of the selected tools

The Test Manager should also consider the opportunity costs inherent in any tool. The time spent on acquiring, administering, training and using the tool could have been spent on actual testing tasks; therefore, more testing resources may be needed up front until the tool goes on line.

There are many risks to tool use; not all tools actually supply benefits that outweigh the risks. Tool risks were discussed in the Foundation Level syllabus. The Test Manager should also consider the following risks when considering ROI:

- Immaturity of the organization (it is not ready to use the tool)
- Artifacts created by the tool may be difficult to maintain, requiring multiple revisions when the software under test is changed
- Reduction of Test Analysts' involvement in testing tasks may reduce the value of testing (e.g., defect detection effectiveness may be reduced when only automated scripts are run)

Finally, a Test Manager must look at the benefits that may accrue from the use of the tool. Benefits of tool introduction and use may include:

- Reduction in repetitive work
- Reduction in test cycle time (e.g., by using automated regression testing)
- Reduction in test execution costs
- Increase in certain types of testing (e.g., regression testing)
- Reduction in human error in different phases of testing.   Examples include:
    - Test data may be more valid using data generation tools
    - Test result comparisons may be more accurate using comparison tools
    - Test data entry may be more correct by entering with a scripting tool
- Reduction in effort required to access information about tests. Examples include:
    - Tool generated reports and metrics
    - Reuse of test assets such as test cases, test scripts and test data
- Increase in testing that was not possible without tools (e.g., performance tests, load tests)
- Improvement in the status of testers who create automation and the testing organization as a whole by demonstrating the understanding and use of sophisticated tools

In general, rarely does a testing group use a single tool; the total ROI the test team will obtain is usually a function of all the tools that are used.   Tools need to share information and work cooperatively together. A long-term, comprehensive test tool strategy is advisable.

6.2.4 Selection Process

Test tools are a long-term investment, likely extending over many iterations of a single project, and/or applicable to many projects. A Test Manager must consider a prospective tool from several different viewpoints.

- To the business, a positive ROI is required. In order to obtain high value in their investments, the organization should ensure that those tools that must interoperate—which may include both test tools and non-test tools—work together. In some cases, the processes and connectivity for tool usage must be improved to achieve this interoperability, and this can take some time to achieve.
- To the project, the tool must be effective (e.g., to avoid mistakes while doing manual testing, such as typing errors during data input). The tool may require an appreciable amount of time in order to start earning positive ROI. In many cases, ROI may occur in the second release or during maintenance rather than during the initial project when the automation was implemented. The Test Manager should consider the total lifecycle of the application.
- To the person who uses the tool, the tool must support the project members in doing their tasks in a more efficient and effective way. The learning curve must be considered to ensure the users will be able to learn the tool quickly with minimal stress. When first introduced, test tools require training and mentoring for the users.

In order to ensure that all viewpoints are taken into account, it is important to create a roadmap for the test tool introduction.

The selection process for a test tool was already discussed in the Foundation Level syllabus as follows:
- Assess organizational maturity
- Identify requirements for tools
- Evaluate tools
- Evaluate the vendor or service support (open-source tools, custom tools)
- Identify internal requirements for coaching and mentoring in the use of the tools
- Evaluate training needs considering the current test team's test automation skills
- Estimate cost-benefits (as discussed in Section 6.2.3 ROI)

For each type of tool, regardless of the testing phase in which it is to be used, a Test Manager should consider the capabilities listed below:
- Analysis
  - Will this tool be able to "understand" the input it is given?
  - Is the tool suited for the purpose?
- Design
  - Will this tool help design testware based on existing information (e.g., test design tools that create test cases from the requirements)?
  - Can the design be generated automatically?
  - Can the actual testware code be generated or partly generated in a maintainable and usable format?
  - Can the necessary test data be generated automatically (e.g., data generated from code analysis)?
- Data and test selection
  - How does the tool select the data it needs (e.g., which test case to execute with which set of data)?
  - Can the tool accept selection criteria entered either manually or automatically?
  - Can the tool determine how to "scrub" production data based on selected inputs?
  - Can the tool determine which tests are needed based on coverage criteria (e.g., given a set of requirements can the tool traverse the traceability to determine which test cases are needed for execution)?
- Execution
  - Will the tool run automatically or will manual intervention be required?
  - How does the tool stop and restart?
  - Should the tool be capable of "listening" for pertinent events (e.g., should a test management tool automatically update test case status if a defect reported against the test case is closed?)
- Evaluation
  - How will the tool determine if it has received a proper result (e.g., the tool will use a test oracle to determine the correctness of a response)?
  - What type of error recovery capabilities does the tool possess?
  - Does the tool provide adequate logging and reporting?

6.3 Tool Lifecycle

There are four different stages in a tool's useful lifecycle that a Test Manager must manage, as follows:
1. Acquisition. The tool must be acquired as discussed above (see Section 6.2 Tool Selection). After a decision to bring in the tool has been made, the Test Manager should assign someone (often a Test Analyst or Technical Test Analyst) to be the administrator of the tool. This

person should make decisions as to how and when the tool will be used, where created artifacts will be stored, naming conventions, etc. Making these decisions up front rather than allowing them to occur in an ad hoc manner can make a significant difference in the eventual ROI of the tool. Training will likely need to be supplied to the users of the tool.

2. Support and maintenance. Ongoing processes for support and maintenance of the tool will be required. The responsibility for maintaining the tool may go to the administrator of the tool, or it might be assigned to a dedicated tools group. If the tool is to work with other tools, then data interchange and processes for cooperation should be considered. Decisions on backup and restore of the tool and its outputs need to be considered.

3. Evolution. Conversion must be considered. As time goes on, the environment, business needs, or vendor issues may require that major changes to the tool or its use be made. For example, the tool vendor may require an update to the tool which causes issues with cooperating tools. A necessary change to the environment for business reasons may cause problems with the tool. The more complex the operating environment for a tool, the more evolutionary change may disrupt its use. At this point, depending on the role the tool plays in testing, a Test Manager may need to ensure that the organization has a way to ensure continuity of service.

4. Retirement. The time will come when the tool has outlasted its useful lifetime. At this point, the tool will need to be retired gracefully. The functionality supplied by the tool will need to be replaced and data will need to be preserved and archived. This can occur because the tool is at the end of its lifecycle, or simply because it has reached a point where the benefits and opportunities of conversion to a new tool exceed the costs and risks.

Throughout the tool lifecycle, it is the responsibility of the Test Manager to ensure the smooth functioning and graceful continuation of the services that the tool provides to the test team.

6.4 Tool Metrics

A Test Manager can design and gather objective metrics from tools used by Technical Test Analysts and Test Analysts. Different tools can capture valuable real-time data and can reduce the effort of data collection. This data can be used by the Test Manager to manage the overall test effort.

Different tools are focused on collecting different types of data. Examples of these include the following:

- Test management tools can supply a variety of different metrics. Traceability from requirements to test cases and automated scripts allows coverage metrics to be obtained. A snapshot of currently available tests, planned tests and current execution status (passed, failed, skipped, blocked, in queue) can be taken at any time.
- Defect management tools can supply a wealth of information about defects including current status, severity and priority, distribution throughout the system, etc. Other illuminating data such as the phase in which defects are introduced and found, escape rates, etc., help the Test Manager drive process improvement.
- Static analysis tools can help detect and report maintainability issues.
- Performance tools can supply valuable information on the scalability of the system.
- Coverage tools can help the Test Manager understand how much of the system has actually been exercised through testing.

The reporting requirements of tools should be defined during the tool selection process. Those requirements must be properly implemented during tool configuration to ensure that the information tracked by the tools can be reported in ways that will be understandable to the stakeholders.

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB

Inte...
Software...
Qualifications...

7. People Skills – Team Composition – 210 mins.

Keywords
Independence of testing

Learning Objectives for People Skills – Team Composition

7.2 Individual Skills
TM-7.2.1   (K4) Using a skills assessment spreadsheet, analyze the strengths and weaknesses of team members related to use of software systems, domain and business knowledge, areas of systems development, software testing and interpersonal skills

TM-7.2.2   (K4) Analyze a given skills assessment for a team in order to define a training and skills development plan

7.3 Test Team Dynamics
TM-7.3.1   (K2) For a given situation, discuss the necessary hard and soft skills required to lead a testing team

7.4 Fitting Testing Within an Organization
TM-7.4.1   (K2) Explain options for independent testing

7.5 Motivation
TM-7.5.1   (K2) Provide examples of motivating and demotivating factors for testers

7.6 Communication
TM-7.6.1   (K2) Explain the factors that influence the effectiveness of communication within a test team, and between a test team and its stakeholders

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB

Inter
Software
Qualifications

7.1 Introduction

Successful Test Managers recruit, hire and maintain teams with the proper mix of skills. Skills requirements may change over time, so in addition to hiring the right people in the first place, providing adequate training and growth opportunities are important in order to retain the test team and maintain it at a level of peak performance. In addition to the skills of the team, the Test Manager must also maintain a set of skills that will allow effective functioning in a high pressure, fast-paced environment.

This chapter looks at how to assess skills, how to fill in the gaps to create a synergistic team that is internally cohesive and is effective in an organization, and also how to motivate that team and how to communicate effectively.

7.2 Individual Skills

An individual's ability to test software can be obtained through experience or by education and training. Each of the following can contribute to the tester's knowledge base:
- Use of software systems
- Knowledge of the domain or business
- Participation in various phases of the software development process activities including analysis, development and technical support
- Participation in software testing activities

End-users of software systems have a good understanding of how the system works, where failures would have the greatest impact, and how the system should react in various situations. Users with domain expertise know which areas are of most importance to the business and how those areas affect the ability of the business to meet its requirements. This knowledge can be used to help prioritize the testing activities, create realistic test data and test cases, and to verify or create use cases.

Knowledge of the software development process (requirements analysis, architecture, design and coding) gives insight into how errors lead to the introduction of defects, where defects can be detected and how to prevent the introduction of defects in the first place. Experience in technical support provides knowledge of the user experience, expectations and usability requirements. Software development experience is important for the use of test tools that require programming and design expertise, and to participate in static code analysis, code reviews, unit testing, and technically focused integration testing.

Specific software testing skills include the capabilities discussed in the Foundation, Advanced Test Analyst, and Advanced Technical Test Analyst syllabi, such as the ability to analyze a specification, participate in risk analysis, design test cases, and the diligence for running tests and recording the results.

Specifically for Test Managers, having knowledge, skills and experience in project management is important since test management includes many project management activities, e.g., making a plan, tracking progress and reporting to stakeholders. In the absence of a project manager, the Test Manager may take on the role of both the Test Manager and the project manager, particularly during the later stages of a project. These skills are in addition to the capabilities discussed in the Foundation syllabus and this syllabus.

In addition to technical skills, interpersonal skills, such as giving and receiving constructive criticism, influencing, and negotiating, are all important in the role of testing. Technically competent testers are likely to fail unless they also possess and employ the necessary soft skills. In addition to working

effectively with others, the successful test professional must also be well-organized, attentive to detail and possess strong written and verbal communication skills.

The ideal test team has a mix of skills and experience levels, and the team members should have a willingness and ability to teach and learn from their peers. In some environments, some skills will be more important or more respected than others. For example, in a technical testing environment where API testing and programming skills are required, technical skills may be valued more than domain knowledge. In a black box testing environment, domain expertise may be the most valued. It is important to remember that environments and projects change.

When creating a skills assessment spreadsheet, the Test Manager should list all the skills that are important to the job as well as being appropriate to the position. Once those skills are itemized, each individual in the team can be assessed against a scoring system (e.g., a rating from 1 to 5 with 5 being the highest skill level expected for that area). Individuals can be assessed to determine their strong and weak areas and, based on that information, individual or group training plans can be created. The Test Manager may set a performance goal for individuals to improve their skills in particular areas and should define specific criteria that are used to assess the individual's skills.

People should be hired for the long term, not for their contribution on a single project. When the Test Manager invests in the testers and creates an environment of continuous learning, the team members will be motivated to increase their skills and knowledge so they will be ready for the next opportunity.

A Test Manager will rarely be able to hire the perfect team members. And, even if the team members are perfect for the current project, they may not be the perfect mix for the next project. The Test Manager should hire people who are intelligent, curious, adaptable, willing to work, able to work effectively as part of a team, and willing and able to learn. Even though that set of perfect individuals is probably not available, a strong team can be built by balancing the strengths and weaknesses of the individuals.

Using the skills assessment spreadsheet, the Test Manager can identify where the team is strong and where it is weak. This information forms the foundation of a training and skills development plan. Starting with the weaknesses that have the greatest impact on the team's effectiveness and efficiency, the Test Manager should decide how to address those areas. One approach is training, e.g., send people to a training course, have training sessions in-house, develop custom training, use e-learning courses. Another approach is self-study, e.g., books, webinars, Internet resources. Yet another approach is cross-training, e.g., assigning someone with an interest in learning a skill to work with someone who already has the skill on a task requiring that skill, having local experts give brief presentations on their area of expertise, etc. (Mentoring is a similar approach, where a person who is new to a role is paired with a senior person who has had that role, and the senior person acts as an ongoing resource to provide advice and assistance.) In addition to addressing weakness, the Test Manager should remember to leverage the strengths identified in the skills assessment as part of the training and skills development plan. For more on such test team development plans, see [McKay07].

7.3 Test Team Dynamics

To build the best team possible, staff selection is one of the most important functions of a management role in the organization. There are many items to consider in addition to the specific individual skills required for the job. When selecting an individual to join the team, the dynamics of the team must be considered. Will this person complement the skills and personality types that already exist within the test team? It is important to consider the advantages of having a variety of personality types on the test team as well as a mix of technical skills. A strong test team is able to deal with multiple projects of varying complexity while also successfully handling the interpersonal interactions with the other project team members.

Testing is often a high pressure activity. Software development schedules are often compressed, even unrealistic. Stakeholders have high expectations of the test team, sometimes unreasonably so. The Test Manager must hire people who will deal well with high pressure situations, who can deflect frustration and who can concentrate on the work even when the schedules seem impossible. It is up to the Test Manager to deal with schedule and expectation issues, but the Test Manager must also understand that these pressures are felt by the team members as well. When the Test Manager is acquiring people for the team, it is important to consider the working environment and to match the personality types to that environment. In an environment where there is more work than time, the Test Manager should be seeking people who finish their tasks and ask what they can do next.

Individuals will work harder and will take more care in what they do if they feel valued and needed. The individuals must understand that they are each an important member of the team and that their contribution is vital to the success of the team as a whole. When this dynamic is created, cross-training will occur informally, work load balancing will be done by the team members themselves and the Test Manager will have more time to deal with external issues.

New team members must be quickly assimilated into the team and provided with adequate supervision and support. Each person should be given a defined role on the team. This can be based on an individual assessment process. The goal is to make each individual successful as an individual while contributing to the overall success of the team. This is done largely by matching personality types to team roles and building on the individual's innate skills as well as increasing their skill portfolio.

When determining whom to hire or add to the test team, an objective assessment of skills can be helpful. This can be done via interviewing, testing the candidate, reviewing work samples and by verifying references. Skills that should be assessed include:

Technical skills (hard skills) can be demonstrated by:
- Deriving test cases from a requirements document
- Reviewing technical documentation (requirements, code, etc.)
- Writing review comments in a clear, understandable and objective way
- Applying various testing techniques appropriately to given scenarios (e.g., using decision tables to test a set of business rules)
- Assessing a failure and documenting it accurately
- Demonstrating an understanding of defect classification information
- Demonstrating an understanding of root causes of defects
- Using a tool to test a given API, including positive and negative tests
- Using SQL to find and alter database information to test a given scenario
- Designing a test automation harness that will execute multiple tests and gather test results
- Executing automated tests and troubleshooting failures
- Writing test plans/specifications
- Writing a test summary reports that includes an assessment of the test results

Interpersonal skills (soft skills) can be demonstrated by:
- Presenting information regarding a test project that is behind schedule
- Explaining a defect report to a developer who thinks there is no defect
- Training a co-worker
- Presenting a problem to management regarding a process that is not effective
- Reviewing a test case created by a co-worker and presenting the comments to that person
- Interviewing a co-worker
- Complimenting a developer

While this is not a complete list and the specific desired skills will vary between environments and organizations, it is a list of skills that are commonly needed. By creating effective interview questions and allowing skills demonstrations, the Test Manager can assess the skills of the candidate and determine areas of strength and weakness. Once a good set of assessments is created, it should be applied to all the existing personnel as well to determine areas for growth and training.

In addition to the skills needed by the individual contributors, the Test Manager must also possess excellent communication and diplomatic skills. The Test Manager must be able to defuse controversial situations, must know the correct media to use for the necessary communication and must be able to focus on building and maintaining relationships within the organization.

7.4 Fitting Testing Within an Organization

Organizations have many ways to fit testing into the organizational structure. While quality is everyone's responsibility throughout the software development lifecycle, an independent test team can contribute significantly to a quality product. Independence of the testing function varies widely in practice, as seen from the following list, ordered from least to most independence:
- No independent testers
  - In this case there is no independence and the code is being tested by the developer who wrote it
  - The developer, if allowed time to do the testing, will determine that the code works as intended, which may or may not match the actual requirements
  - The developer can fix any found defects quickly
- Testing is done by a developer other than the one who wrote the code
  - There is little independence between the developer and the tester
  - A developer testing another developer's code may be reluctant to report defects
  - A developer mind set toward testing is usually focused on positive test cases
- Testing is done by a tester (or test team) that is part of the development team
  - The tester (or test team) may report to project management or may report to the development manager
  - The tester mind set is focused more on verifying adherence to requirements
  - Because the tester is a member of the development team, the tester may have development responsibilities in addition to testing
  - The tester's manager may be more focused on meeting schedule than achieving quality goals
  - Testing may be given a lower status in the team than development
  - The tester may have no authority to influence the quality goals or the adherence to those goals
- Testing is done by test specialists from the business organization, user community, or other non-development technical organization
  - Test results information is reported objectively to the stakeholders
  - Quality is the primary focus of this team
  - Skills development and training are focused on testing
  - Testing is seen as a career path
  - There is specific management devoted to the testing group that is quality-focused
- External test specialists perform testing on specific test types
  - Expertise is applied to specific areas where generalized testing is likely to be insufficient
  - Test types could be usability, security, performance or other areas where specialization is necessary
  - Quality should be the focus of these individuals, but their view is limited to their areas of specialty. A product that performs well may not meet its functional requirements, but that may go undetected by the performance specialist.
- Testing is done by an organization external to the company

- o This model achieves the maximum independence between the tester and the developer
- o Knowledge transfer may not be sufficient for the tester to test competently
- o Clear requirements and a well-defined communication structure will be needed
- o Quality of the external organization must be audited regularly

This list is based on the typical focus of the individuals, but that may not be true in a particular organization. A position and title do not necessarily determine a person's focus. Development managers can be quality-focused and so may be good Test Managers. Independent Test Managers may be reporting to a schedule-focused management chain and so may be more schedule-focused than quality-focused. To determine the best location for a test team within an organization, the Test Manager must understand the goals of the organization.

There are varying degrees of independence between the development and testing organizations. It is important to understand that there may be a tradeoff where more independence results in more isolation and less knowledge transfer. A lower level of independence may increase knowledge, but can also introduce conflicting goals. The level of independence will also be determined by the software development model being used, e.g., within Agile development the testers are most often part of the development team.

Any of the above options can be present in an organization. There may be testing done within the development organization as well as by an independent testing organization, and there may be final certification by an external organization. It is important to understand the responsibilities and expectations for each phase of testing and to set those requirements to maximize the quality of the finished product while staying within schedule and budget constraints.

7.5 Motivation

There are many ways to motivate an individual in a testing position. These include:
- Recognition for the job accomplished
- Approval by management
- Respect within the project team and among peers
- Increased responsibility and autonomy
- Adequate rewards for the work done (including salary, increased responsibility, recognition)

There are project influences that can make these motivational tools difficult to apply. For example, a tester can work very hard on a project that has an impossible deadline. The tester can do everything possible to drive the quality focus of the team, put in extra hours and effort, and yet the product may ship before it should due to external influences. The result may be a poor quality product despite the best efforts of the tester. This can easily be a demotivator if the tester's contribution is not understood and measured, regardless of whether the end product is successful.

The test team must ensure that it is tracking the appropriate metrics to prove that a good job was done to accomplish the testing, mitigate risks and accurately record the results. Unless this data is gathered and published, it is easy for a team to become demotivated when they do not receive the recognition they feel is due for a job well-done.

Recognition is not just determined in the intangibles of respect and approval, it is also apparent in promotion opportunities, increased responsibility and merit increases. If the testing group is not respected, these opportunities may not be available.

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 best achieved by involving the tester at the beginning of the project and with continuing involvement throughout the lifecycle. Over time the testers

will win respect of the other project stakeholders by their positive contribution to the project. This contribution should also be quantified in terms of cost of quality reductions and risk mitigation.

The Test Manager plays an important part in motivating the individuals in a test team as well as acting as a champion of the test team to the larger organization. The Test Manager should recognize the achievements of the individual testers, and must also give fair and honest appraisals of mistakes. A fair and consistent Test Manager will motivate the team members by example.

7.6 Communication

Test team communication primarily takes place by the following means:
- Documentation of test products - test strategy, test plan, test cases, test summary reports, defect reports, etc.
- Feedback provided on reviewed documents - requirements, functional specifications, use cases, component test documentation, etc.
- Information gathering and dissemination - interaction with developers, other test team members, management, etc.

The communication between testers and other stakeholders must be professional, objective and effective in order to build and maintain respect for the test team. Diplomacy and objectivity are required when providing feedback, particularly constructive feedback, on the work products of others. In addition, communication should be focused on achieving test objectives and on improving quality both in products and the processes used to produce the software systems.

Test Managers communicate with a wide audience, including users, project team members, management, external testing groups and customers. Communication must be effective for the target audience. For example a report produced for the development team showing trends and tendencies in the number and severity of defects found might be too detailed to be appropriate for an executive management briefing. With any communication, but particularly during presentations, it is important to understand the message that is being sent, the ways in which the message may be received and the explanation that is needed to create the correct environment for the message to be accepted. Since the Test Manager is often in a position of presenting project status information, it is important that this information be at an appropriate level of detail (e.g., managers usually want to see defect trends rather than individual defects), and be presented in a way that is clearly stated and easily understood (e.g., simple charts and colorful graphs). Communicating efficiently helps to keep the attention of the audience while still conveying the correct message. Each presentation should be viewed by the Test Manager as an opportunity to promote quality and quality processes.

A Test Manager is not only communicating with people external to the department (outward communication). An important part of a Test Manager's job is to communicate effectively within the testing group (inward communication) to pass on news, instructions, changes in priorities and other standard information that is imparted in the normal process of testing. A Test Manager may also communicate to specific individuals both up (upward communication) and down (downward communication) the management chain in the organization. Regardless of the direction of the communication, the same rules apply; the communication should be appropriate for the audience, the message should be sent effectively and understanding should be confirmed.

Test Managers must be masters of the various means of communication. Much information is communicated via e-mail, verbal interactions, formal or informal meetings, formal or informal reports and even through the use of the test management tools such as defect management tools. All communication must be professional and objective. Proofreading, both for quality and content, is a necessary step even in the most urgent communication. Written communication often lives long beyond the actual project. It is important for the Test Manager to produce professional quality documentation that is representative of an organization that promotes quality.

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB

Inter
Software
Qualifications

8. References

8.1 Standards

- [BS7925-2] BS 7925-2, Software Component Testing Standard.
  Chapter 2

- [FDA21] FDA Title 21 CFR Part 820, Food and Drug Administration, USA
  Chapter 2

- [IEEE829] IEEE Standard for Software and System Test Documentation
  Chapter 2 and 4

- [IEEE1028] IEEE Standard for Software Reviews and Audits
  Chapter 2

- [IEEE1044] IEEE Standard Classification for Software Anomalies
  Chapter 4

- [ISO9126] ISO/IEC 9126-1:2001, Software Engineering - Software Product Quality,
  Chapters 2 and 4

- [ISO12207] ISO 12207, Systems and Software Engineering, Software Life Cycle Processes
  Chapter 2

- [ISO15504] ISO/IEC 15504, Information Technology - Process Assessment
  Chapter 2

- [ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product Quality
  Requirements and Evaluation (SQuaRE)
  Chapters 2 and 4

- [RTCA DO-178B/ED-12B]: Software Considerations in Airborne Systems and Equipment
  Certification, RTCA/EUROCAE ED12B.1992.
  Chapter 2

8.2 ISTQB Documents

- [ISTQB_AL_OVIEW]   ISTQB Advanced Level Overview, Version 1.0
- [ISTQB_ALTM_SYL]   ISTQB Advanced Level Test Manager Syllabus, Version 1.0
- [ISTQB_ALTTA_SYL]   ISTQB Advanced Level Technical Test Analyst Syllabus, Version 1.0
- [ISTQB ETM SYL]   ISTQB Expert Test Management Syllabus, Version 1.0
- [ISTQB_FL_SYL]   ISTQB Foundation Level Syllabus, Version 2011
- [ISTQB_GLOSSARY]   Standard glossary of terms used in Software Testing, Version 2.2,
  2012
- [ISTQB ITP SYL]   ISTQB Expert Improving the Test Process Syllabus, Version 1.0

8.3 Trademarks

The following registered trademarks and service marks are used in this document:

- CMMI®, IDEAL^{SM}, ISTQB®, ITIL®, PRINCE2®, TMMi®, TPI Next®
- CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.
- IDEAL is a service mark of Software Engineering Institute (SEI), Carnegie Mellon University
- ISTQB is a registered trademark of the International Software Testing Qualifications Board
- ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office
- PRINCE2 is a registered trademark of the Cabinet Office
- TMMi is a registered trademark of TMMi Foundation
- TPI Next is a registered trademark of Sogeti Nederland B.V.

8.4 Books

[Black03]: Rex Black, "Critical Testing Processes," Addison-Wesley, 2003, ISBN 0-201-74868-1

[Black09]: Rex Black, "Managing the Testing Process, third edition," John Wiley & Sons, 2009, ISBN 0-471-22398-0

[Black11]: Rex Black, Erik van Veenendaal, Dorothy Graham, "Foundations of Software Testing," Thomson Learning, 2011, ISBN 978-1-84480-355-2

[Craig02]: Rick Craig, Stefan Jaskiel, "Systematic Software Testing," Artech House, 2002, ISBN 1-580-53508-9

[Crispin09]: Lisa Crispin, Janet Gregory, "Agile Testing: A Practical Guide for Testers and Agile Teams", Addison-Wesley, 2009, ISBN 0-321-53446-8

[de Vries09]: Gerrit de Vries, et al., "TPI Next", UTN Publishers, 2009, ISBN 90-72194-97-7

[Goucher09]: Adam Goucher, Tim Riley (editors), "Beautiful Testing," O'Reilly, 2009, ISBN 978-0596159818

[IDEAL96]: Bob McFeeley, "IDEAL: A User's Guide for Software Process Improvement," Software Engineering Institute (SEI), 1996, CMU/SEI-96-HB-001

[Jones07]: Capers Jones, "Estimating Software Costs, second edition," McGraw-Hill, 2007, ISBN 978-0071483001

[Jones11]: Capers Jones and Olivier Bonsignour, "Economics of Software Quality," Pearson, 2011, ISBN 978-0132582209

[McKay07]: Judy McKay, "Managing the Test People," Rocky Nook, 2007, ISBN 978-1933952123

[Musa04]: John Musa, "Software Reliability Engineering, second edition," Author House, 2004, ISBN 978-1418493882

[Stamatis03]: D.H. Stamatis, "Failure Mode and Effect Analysis," ASQC Quality Press, 2003, ISBN 0-873-89300

[vanVeenendaal11] Erik van Veenendaal, "The Little TMMi," UTN Publishers, 2011, ISBN 9-490-986038

[vanVeenendaal12] Erik van Veenendaal, "Practical Risk-based Testing," UTN Publishers, 2012, ISBN 978-9490986070

[Whittaker09]: James Whittaker, "Exploratory Testing," Addison-Wesley, 2009, ISBN 978-0321636416

[Wiegers03]: Karl Wiegers, "Software Requirements 2," Microsoft Press, 2003, ISBN 978-0735618794

8.5 Other References

The following references point to information available on the Internet. These references were checked at the time of publication of this Advanced Level syllabus.

http://www.istqb.org
http://www.sei.cmu.edu/cmmi/

http://www.tmmi.org/
http://www.tpinext.com/

Certified Tester
Advanced Level Syllabus - Test Manager

ISTQB®

Inter
Software
Qualifications

9. Index