Certified Tester Advanced Level Test Management Syllabus

Version 3.0

International Software Testing Qualifications Board

Copyright Notice

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 chapter 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
Beta 3.0	31OCT23	Incorporation of all comments from member boards received for all sections (BOIncs) from the Alpha review
POST Beta 3.0	31JAN24	Incorporation of all comments from member boards received for all sections (BOIncs) from the Beta review
POST Beta 3.0	29FEB24	Minor modifications in proofreading

Table of Contents

Acknowledgements

This document was formally released by the General Assembly of the ISTQB® on <date>

It was produced by a team from the International Software Testing Qualifications Board: Horst Pohlmann (Product Owner, Vice Chair AELWG), Tauhida Parveen, Francis Fenner, Laura Albert, Matthias Hamburg, Maud Schlich, Tanja Tremmel, Ralf Bongard, Erik van Veenendaal, Jan Giessen, Bernd Freimut, Andreas Neumeister, Georg Sehl, Rabi Arabi, Therese Kuhfuß, Ecaterina Irina Manole, Veronica Belcher, Kenji Onishi, Pushparajan Balasubramanian, Meile Postuma and Miroslav Renda.

The team thanks Gary Mogyorodi for his technical review (in Beta), Julia Sabatine for her proofreading, the review team and the Member Boards for their suggestions and input.

The following persons participated in the reviewing, commenting, and balloting of this syllabus:

Alpha reviews: Benjamin Timmermans, Mattijs Kemmink, Rik Marselis, Jean-Francois Riverin, Gary Mogyorodi, Ralf Bongard, Ingvar Nordström, Yaron Tsubery, Imre Mészáros, Mattijs Kemmink, Ádám Bíró, Ramit M Kaul, Chinthaka Indikadahena, Darvay Tamás Béla, Beata Karpinska, Young jae Choi, Stuart Reid, Tal Pe'er, Meile Posthuma, Daniel van der Zwan, Klaudia Dussa-Zieger, Jörn Münzel, Ralf Bongard, Petr Neugebauer, Derk-Jan de Grood, Rik Kochuyt, Andreas Hetz, Laura Albert, Eszter Sebestyeni, Tamás Szőke, Henriett Braunné Bokor, Ágota Horváth, Péter Sótér, Ferenc Hamori, Darvay Tamás Béla, Paul Weymouth, Lloyd Roden, Kevin Chen, Huang qin, Pushparajan Balasubramanian, Szilard Szell, Tamas Stöckert, Lucjan Stapp, Adam Roman, Anna Miazek, Márton Siska, Erhardt Wunderlich, László Kvintovics, Murian Song, Mette Bruhn-Pedersen, Petra Schneider, Michael Stahl, Ramit M Kaul, Imre Mészáros, Dilhan Jayakody, Francisca Cano Ortiz, Johan Klintin, Liang Ren, Ole Chr. Hansen, Zsolt Hargitai, Tamás Rakamazi, Kenji Onishi, Arnika Hryzszko, Rabih Arabi, Veronica Belcher, and Vignesh Balasubramanian.

Beta reviews: Maria-Therese Teichmann, Dominik Weber, Thomas Puffler, Peter Kunit, Martin Klonk, Michaël Pilaeten, Wim Decoutere, Arda Ender Torçuk, Piet de Roo, Rik Marselis, Jakub Platek, Ding Guofu, Zheng Dandan, Liang Ren, Yifan Chen, Hallur Helmsdal, Ole Chr. Hansen, Klaus Skafte, Gitte Ottosen, Tanzeela Gulzar, Arne Becher, Klaudia Dussa-Zieger, Jan Giesen, Florian Fieber, Carsten Weise, Arnd Pehl, Matthias Hamburg, Stephanie Ulrich, Jürgen Beniermann, Márton Siska, Sterbinszky Ádám, Ágnes Srancsik, Marton Matyas, Tamas Stöckert, Csilla Varga, Zsolt Hargitai, Bíró Ádám, Horváth Ágota, Sebestyéni Eszter, Szilárd Széll, Péter Sótér, Giancarlo Tomasig, Nicola de Rosa, Kaiwalya Katyarmal, Pradeep Tiwari, Sreeja Padmakumari, Seunghee Choi, Stuart Reid, Dmitrij Nikolajev, Mantas Aniulis, Monika Stoecklein-Olsen, Adam Roman, Mahmoud Khalaili, Ingvar Nordström, Beata Karpinska, Armin Born, Ferdinand Gramsamer, Mergole Kuaté, Thomas Letzkus, Nishan Portoyan, Ainsley Rood, Lloyd Roden, Sarah Ireton.

The Advanced Test Manager Syllabus 2010-2012 was produced by a core team from the International Software Testing Qualifications Board Advanced Level Sub Working Group - Advanced Test Manager: Rex Black (Chair), Judy McKay (Vice Chair), Graham Bath, Debra Friedenberg, Bernard Homès, Paul Jorgensen, Kenji Onishi, Mike Smith, Geoff Thompson, Erik van Veenendaal, Tsuyoshi Yumoto.

The core team thanks the review team and the National Boards for their suggestions and input.

At the time the Advanced Level Syllabus was completed the Advanced Level Working Group had the following membership (alphabetical order):

Graham Bath, Rex Black, Maria Clara Choucair, Debra Friedenberg, Bernard Homès (Vice Chair), Paul Jorgensen, Judy McKay, Jamie Mitchell, Thomas Mueller, Klaus Olsen, Kenji Onishi, Meile Posthuma,

Eric Riou du Cosquer, Jan Sabak, Hans Schaefer, Mike Smith (Chair), Geoff Thompson, Erik van Veenendaal, Tsuyoshi Yumoto.

The following persons participated in the reviewing, commenting, and balloting of this syllabus:

Chris van Bael, Graham Bath, Kimmo Hakala, Rob Hendriks, Marcel Kwakernaak, Rik Marselis, Don Mills, Gary Mogyorodi, Thomas Mueller, Ingvar Nordstrom, Katja Piroué, Miele Posthuma, Nathalie Rooseboom de Vries, Geoff Thompson, Jamil Wahbeh, Hans Weiberg.

0 Introduction

0.1 Purpose of this Syllabus

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

1. To member boards, to translate into their local language and to accredit training providers. Member boards may adapt the syllabus to their particular language needs and modify the references to adapt to their local publications.
2. To certification bodies, to derive examination questions in their local language adapted to the learning objectives for this syllabus.
3. To ISTQB® accredited training providers, to produce courseware and determine appropriate teaching methods.
4. To certification candidates, to prepare for the certification exam (ISTQB® recommends taking an ISTQB® accredited training prior to attend an ISTQB® Advanced Level exam.)
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.

0.2 The Certified Tester Advanced Level Test Management in Testing

This Advanced Level qualification is aimed at anyone involved in the management of software testing. This includes people in roles such as testers, test consultants, test managers, user acceptance testers, scrum masters, project managers or product owners. This Advanced Level Test Management qualification is also appropriate for anyone who wants an advanced understanding of software testing such as project managers, quality managers, software development managers, business analysts, IT directors and management consultants. Holders of the Advanced Level Test Management Certificate will be able to go on to the ISTQB® Expert Level software testing qualifications. The ISTQB® Certified Tester Advanced Level – Test Manager or Test Management certificate is valid for life and does not require renewal. The certificate is recognized internationally and demonstrates the candidates' professional competence and credibility in test management.

0.3 Career Path for Testers

The ISTQB® scheme provides support for testing professionals at all stages of their careers. Individuals who achieve the ISTQB® Certified Tester Advanced Level Test Management Certification may also be interested in the other Core Advanced Level certifications (i.e., Test Analyst and Technical Test Analyst) and thereafter the ISTQB® Expert Level certifications (i.e., Test Management or Improving the Test Process). Anyone seeking to develop skills in testing within Agile software development can consider the Agile Technical Tester or Agile Test Leadership at Scale certifications. The Specialist stream offers a deep dive into areas that have specific test approaches and test activities (e.g., in Test Automation, AI Testing, or Mobile App Testing), or cluster testing know-how for certain industry domains (e.g., Automotive or Gaming). Please visit www.istqb.org for the latest information of ISTQB®´s Certified Tester Scheme.

0.4 Business Outcomes

This section lists the eleven Business Outcomes expected of a candidate who has achieved the Advanced Level Test Management certification.

An Advanced Level Test Management Certified Tester can…

TM_01	Manage testing in various software development projects by applying test management processes established for the project team or test organization
TM_02	Identify test stakeholders and software development lifecycle models that are relevant in a given context
TM_03	Organize risk identification and risk assessment sessions within any software development lifecycle and use the results to guide testing to meet the test objectives
TM_04	Define a project test strategy consistent with the organizational test strategy and project context
TM_05	Continuously monitor and control testing to achieve project goals
TM_06	Assess and report test progress to project stakeholders
TM_07	Identify necessary skills and develop those skills within your team
TM_08	Prepare and present a business case for testing in different contexts that outlines the costs and expected benefits
TM_09	Lead test process improvement activities in projects or software development product streams and contribute to organizational test process improvement initiatives
TM_10	Plan the test activities including the required test infrastructure and estimate the effort required to test
TM_11	Create defect reports and a defect workflow suitable for a software development lifecycle

0.5 Examinable Learning Objectives and Cognitive Level of Knowledge

Learning objectives support business outcomes and are used to create the Certified Tester Advanced Test Management exams.

In general, all contents of this syllabus are examinable at a K1 level, except for the Introduction, References, Epilogue and Appendices. That is, the candidate may be asked to recognize, remember, or recall a keyword or concept mentioned in any of the three chapters in this syllabus. The specific learning objectives and corresponding levels are shown at the beginning of each chapter, and classified as follows:

- K2: Understand
- K3: Apply
- K4: Analyze

Further details and examples of learning objectives are given in Appendix A.

All terms listed as keywords right under chapter headings shall be remembered (K1), even if not explicitly mentioned in the learning objectives.

0.6 The Advanced Level Test Management Certificate Exam

The Advanced Level Test Management Certificate exam is based on this syllabus. Answers to exam questions may require the use of material based on more than one section of this syllabus. All sections of the syllabus are examinable, except for the Introduction, Appendices and References. Standards and books are included as references (Chapter 5), but their content, beyond what is summarized in the syllabus itself from such standards and books, is not examinable

Refer to the ´Exam Structures and Rules 1.1 Compatible with Syllabus Foundation and Advanced Levels and Specialists Modules` document for further details.

- The entry criterions for taking the Advanced Level Test Management Certificate exam are:
    - Candidates hold the ISTQB® Foundation Level Certificate before taking the Advanced Level Test Management certification exam. However, it is strongly recommended that the candidate has at least a minimal background in either software development or software testing, such as six months experience as a tester or as a software developer.
    - Completion of a course that has been accredited to ISTQB® standards (by one of the ISTQB®-recognized member boards).

0.7 Accreditation

An ISTQB® Member Board may accredit training providers whose course material follow this syllabus. Training providers should obtain accreditation guidelines from the Member Board or body that performs the accreditation. An accredited course is recognized as conforming to this syllabus and is allowed to have an ISTQB® exam as part of the course.

The accreditation guidelines for this syllabus are the generic Accreditation Guidelines published by the Processes Management and Compliance Working Group of the ISTQB®.

0.8 Handling of Standards

There are standards referenced in the Advanced Level Test Management Syllabus (e.g., ISO, IEC, and IEEE). The purpose of these references is to provide a framework or to provide a source of additional information if desired by the reader. Please note that the syllabus uses these standards as references. The standards are not intended for examination. Refer to Chapter 5 References for more information regarding standards.

0.9 Keeping It Current

The software industry changes rapidly. To deal with these changes and to provide the stakeholders with access to relevant and current information, the ISTQB® working groups have created links on the www.istqb.org website, which refer to supporting documents and changes to standards. This information is not examinable under any ISTQB® syllabus.

0.10 Level of Detail

The level of detail in this syllabus allows internationally consistent courses and exams. To achieve this goal, the syllabus consists of:

- General instructional objectives describing the intention of the Advanced Level Test Management Syllabus
- A list of keywords that students must be able to recall
- Learning objectives for each knowledge area, describing the cognitive learning outcome to be achieved
- A description of the key concepts, including references to sources such as accepted literature or standards

The syllabus content is not a description of the entire knowledge area of software testing; it reflects the level of detail to be covered in Advanced Level Test Management training courses. It focuses on test concepts and test techniques that can be applied to all software projects.

0.11 How this Syllabus is Organized

There are three chapters with examinable content. The top-level heading for each chapter specifies the minimum time required for accredited training courses to cover the chapter contents; timing is not provided below chapter level. For accredited training courses, the syllabus requires a minimum of 22.75 hours of instruction, distributed over the three chapters as follows:

- Chapter 1: Managing the Test Activities (750 minutes)
  - The participant learns to explain the test management activities (i.e., test planning, test monitoring, test control, and test completion)
  - The participant learns to define a project test strategy including the test objectives and selecting the proper test approach consistent with the organizational test strategy and the project context
  - The participant learns to manage projects in various contexts
  - The participant learns to apply risk-based testing to focus testing on the identified risks
  - The participant learns to lead the test process improvement activities by conducting a project or iteration retrospective
  - The participant learns how to improve tool support for testing by taking into consideration the risks, costs, and benefits of tool support
- Chapter 2: Managing the Product (390 minutes)
  - The participant learns how to monitor and control testing to meet test objectives using test metrics and to assess and report the test progress
  - The participant learns to select proper test estimate techniques at different subsidiaries following different software development models

- The participant learns how to define a defect workflow within defect management to fit sequential, Agile, and hybrid software development models

- Chapter 3: Managing the Team (225 minutes)
  - The participant learns how to analyze a given project context to identify the skills required by the test team
  - The participant learns to manage a team according to the whole team approach
  - The participant learns how to define a business case for the testing activities in the project

0.12 What are the Fundamental Assumptions of this Syllabus?

This syllabus is intended for anyone who wants to achieve an advanced level of competence in test management, such as test managers, test analysts, test engineers, test consultants, test coordinators, test leaders, and project managers. The syllabus is aligned with the ISTQB® Foundation Level Syllabus V.4, which provides the basic knowledge and understanding of software testing.

This syllabus covers two main roles in testing: the test management role and the testing role. The test management role is also known as the test manager in the context of sequential development model context, where the test manager is typically a separate role from the project manager or the product owner. The test management role is responsible for the overall test process, the test team, and test management. This includes defining the test strategy, planning the test activities, monitoring and controlling the test progress, reporting the test results, and managing the test risks and issues. The test management role also ensures that test objectives are aligned with the business and stakeholder needs, and that test activities are coordinated with other project stakeholders.

The testing role also performs test evaluation, defect management, and test closure activities. The testing role uses various testing techniques to ensure the quality and reliability of the test artifacts and the system under test. The testing role also uses test tools and automation to support the test process and improve the test efficiency and effectiveness. The activities and tasks assigned to these roles may vary depending on the context, such as the project, product, skills, and organization. (see the ISTQB® Foundation Level Syllabus V.4).

The term *test team member* is used in this syllabus to refer to any person in a test management or testing role who performs testing, regardless of the organizational context and other roles. Test teams are composed of individuals with different skills and competencies. Team members may also have varying levels of experience and certification, such as foundation level, advanced level, or expert level. Test team members may also have different roles and responsibilities depending on the test approach and the test process model used, such as Agile testing, model-based testing, risk-based testing, etc.

An important point about the perspective that the syllabus provides is the fact that it focuses on test management at the project level and not test management at the organizational level. Therefore, this syllabus conforms to and contains information that can be used for project-level test management activities, but less so for test management at organizational level.

Hybrid software development is used in this syllabus to refer to a software development approach that combines elements of different software lifecycle models, such as the V-model, Iterative, Incremental, and Agile. Hybrid software development aims to leverage the strengths and mitigate the weaknesses of each model, depending on the context and needs of the project. For example, a hybrid software development approach may use a V-model model for the initial planning and requirements analysis phases, followed by an Agile model for the design, development, and testing phases. Alternatively, a hybrid software

development approach may use an iterative model for the overall project management, while applying an incremental model for each iteration, and an Agile model for each increment. Hybrid software development requires an elevated degree of flexibility, communication, and collaboration among the stakeholders, as well as a clear understanding of the goals, risks, and constraints of each phase and model.

According to this syllabus and the Glossary, a test strategy is a description of how testing will be performed in order to achieve test objectives under given circumstances. A test strategy defines the overall scope, approach, and resources for testing a system or a product. It is typically documented in a test plan or as part of other documents, depending on the context of the testing. A test strategy is influenced by the organizational test strategy, which is a high-level test strategy that describes how testing is done in an organization. A test strategy may also exist for a single test level or a test type, which are specific aspects of testing that focus on different objectives, targets, and criteria. The generic term “test strategy” can be used in any context (project, organization, product). A test approach is the manner in which testing tasks are implemented, especially the selection and combination of test levels, test types, and test techniques for static and dynamic testing, as well as other test practices such as scripted testing, manual testing, back-to-back testing, etc. The test approach chosen by the test management role is a key decision in formulating an appropriate test strategy for a given context.

1 Managing the Test Activities – 750 minutes

Keywords

experience-based testing, functional testing, hybrid software development, incremental development model, iterative development model, non-functional testing, product risk, quality risk, retrospective, risk analysis, risk assessment, risk identification, risk impact, risk level, risk likelihood, risk management, risk mitigation, risk monitoring, risk-based testing, sequential development model, S.M.A.R.T goal methodology, software development lifecycle, test completion, test control, test level, Test Maturity Model integration, test monitoring, test objective, test plan, test planning, test process improvement, test strategy, test type, TPI NEXT

Domain Specific Keywords

goal question metric (GQM), IDEAL, indicator, measure, metric

Learning Objectives for Chapter 1:

1.1 The Test Process

TM-1.1.1 (K2) Summarize test planning

TM-1.1.2 (K2) Summarize test monitoring and test control

TM-1.1.3 (K2) Summarize test completion

1.2 The Context of Testing

TM-1.2.1 (K2) Compare why different stakeholders are interested in testing

TM-1.2.2 (K2) Explain why stakeholders' knowledge is important in test management

TM-1.2.3 (K2) Explain testing in a hybrid software development model

TM-1.2.4 (K2) Summarize test management activities for various software development lifecycles

TM-1.2.5 (K2) Compare test management activities at various test levels

TM-1.2.6 (K2) Compare test management activities for various test types

TM-1.2.7 (K4) Analyze a given project and determine test management activities that emphasize test planning, test monitoring, and test control

1.3 Risk-Based Testing

TM-1.3.1 (K2) Explain the various measures that risk-based testing takes in order to respond to risks

TM-1.3.2 (K2) Give examples of different techniques a test manager can use for identifying risks related to product quality

TM-1.3.3 (K2) Summarize the factors that determine the risk levels related to product quality

TM-1.3.4 (K4) Select appropriate test activities to mitigate risks according to their risk level in a given context

TM-1.3.5 (K2) Differentiate between heavyweight and lightweight examples of risk-based testing techniques

TM-1.3.6 (K2) Give examples of success metrics and difficulties associated with risk-based testing

1.4 The Project Test Strategy

TM-1.4.1 (K2) Explain typical choices for a test approach

TM-1.4.2 (K4) Analyze an organizational test strategy and the project context to select the appropriate test approach

TM-1.4.3 (K3) Use the S.M.A.R.T. goal methodology to define measurable test objectives and exit criteria

1.5 Improving the Test Process

TM-1.5.1 (K2) Explain how to use the IDEAL model for test process improvement on a given project

TM-1.5.2 (K2) Summarize the model-based improvement approach to test process improvement and understand how to apply it on a project context

TM-1.5.3 (K2) Summarize the analytical-based improvement approach to test process improvement and understand how to apply it on a project context

TM-1.5.4 (K3) Implement a project or iteration retrospective to evaluate test processes and discover testing areas to improve

1.6 Test Tools

TM-1.6.1 (K2) Summarize the best practices for tool introduction

TM-1.6.2 (K2) Explain the impact of different technical and business aspects when deciding on a tool type

TM-1.6.3 (K4) Analyze a given situation to create a plan for tool selection, covering risks, costs, and benefits

TM-1.6.4 (K2) Differentiate among the stages of the tool lifecycle

TM-1.6.5 (K2) Give examples for metric collection and evaluation by using tools

1.1 The Test Process

Introduction

The ISTQB® Certified Tester Foundation Level Syllabus V.4 describes a test process that includes the following activities: test planning, test monitoring and control, test analysis, test design, test implementation, test execution, and test completion.

The ISTQB® Foundation Level Syllabus V.4 states that these activities in the test process are often implemented iteratively or in parallel, depending on the Software Development Lifecycle (SDLC) model and the project context. Tailoring these activities within the context of the product and the project is usually required.

In this syllabus, the focus is on the following key test management activities:

- **Test planning:** defining the test objectives, test approach, test scope, test resources, test schedule, test deliverables and test participants (test stakeholders).
- **Test monitoring and control:** tracking the test progress, test results and test deviations; taking corrective actions when necessary; reporting the test status and outcomes to relevant stakeholders.
- **Test completion:** finalizing and archiving the test artifacts, evaluating the test process and the test product, identifying the test process improvement actions, and communicating the test closure to relevant stakeholders.

The ISO/IEC/IEEE 29119-2 standard defines the test management processes that cover these test management activities. These test management processes can be applied at different levels of testing such as project, program, or portfolio. Each level of testing can have its own test plan that aligns with the higher level test plan.

1.1.1 Test Planning Activities

This section focuses on the activities for planning tests for different scopes such as the entire project, a test level, a test type, or a release/iteration/sprint in Agile. Depending on the scope, test planning may begin and end at different points in the development process. Test planning is an activity that involves identifying the activities and resources required to achieve the test objectives identified in the test policy. Test planning should be initiated as early as possible in the development process, preferably before requirements are identified, and should be updated as the project progresses. Test planning is often an iterative process that requires re-planning during the project to accommodate changes and feedback.

The following tasks are part of test planning (similar to those found in the ISO/IEC/IEEE 29119-2):

- **Understand the context and organize test planning**
  Understanding the context of the organization (e.g., the test policy and the organizational test strategy), the scope of the test, and the test item (i.e., the work product being tested) is critical to test planning. (see Section 1.2., The Context of Testing). This also involves all activities needed to organize the test plan development and to obtain approval of those activities and the schedule by the stakeholders (e.g., the product owner, the project manager, or the development team manager).
- **Identify and analyze product risks**
  Risk analysis involves identifying and assessing the potential impact and likelihood of product

risks as part of test planning. See Section 1.3 of this syllabus, Risk-Based Testing, for further details about product risks.

- **Identify risk treatment approaches**
  Based on the risk analysis, the appropriate risk treatment approaches are selected and documented in the test plan. These may include preventive, corrective, or mitigating actions to address the identified risks.
- **Define test approach and estimate and allocate test resources**
  Based on the organizational test strategy, regulatory standards, any constraints given by the project, and the risk treatment approaches, the test approach is defined for the current scope of testing (see Section 1.4, The Project Test Strategy). When a test approach is defined, it is important to estimate the required test resources such as test staff, test tools, test environments, and test data, and to allocate those resources to the test activities.
- **Establish the test plan**
  The test plan must be accepted by all stakeholders, and therefore, disagreements between them should be settled.

1.1.2 Test Monitoring and Control Activities

In order for Test Management to provide efficient test control, a testing schedule and monitoring framework must be established to enable tracking of the status and progress of testing. This framework should include the detailed measures and targets that are required to relate the status of test work products and resources 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 must be defined to achieve this. This can include defining measures and targets that are required 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 in a manner that is understandable and relevant to the project and business stakeholders.

Test monitoring and control are ongoing activities.

Test control compares actual progress against the test plan and implements corrective actions as needed. It guides the testing to meet the test strategies and objectives (see Section 1.4, The Project Test Strategy), and revisits the test planning activities when necessary. The control data requires detailed test planning information for appropriate reactions. This activity involves:

- Implementing the test plan and control directives
- Managing deviations from planned testing
- Treating newly identified and changed risks
- Establishing readiness to begin testing
- Granting and obtaining approval for test completion based on the exit criteria

Test monitoring involves collecting and recording test results, identifying deviations from planned testing, identifying and analyzing new risks that require testing, and monitoring changes for identified risks.

1.1.3 Test Completion Activities

Test completion usually occurs at project milestones (e.g., a release, the end of an iteration, or at test level completion). For any unresolved defects, change requests or product backlog items are created. See the ISTQB® Foundation Level Syllabus V.4. Once exit criteria are met, the key outputs should be captured, archived and provided to the relevant stakeholders. Test completion requires the following tasks:

- **Create and approve the test completion report**
  This task ensures that all testing has been accomplished and all test objectives have been met. This task involves collecting relevant information from various testware such as test plans, test results, test progress reports, test completion reports, and defect reports. The collected information is evaluated and summarized in the test completion report. The test completion report is approved and communicated to relevant stakeholders.
- **Archive testware**
  This task identifies testware that can be useful in the future or expected to be reused, which are typically test cases. It makes them accessible and easy to understand for future reuse. In addition, test results, test logs, test reports, and other testware should temporarily be archived in the configuration management system.
- **Handover testware**
  This task delivers valuable work products to those who need them. For example, known defects deferred or accepted should be communicated to those who will use, or support the use of testware.
- **Perform all necessary tasks to clean the test environment and to restore it to a pre-defined state**
  This task ensures that the test environment is ready for the next testing cycle or project. It involves removing any test data, test tools, test drivers, test stubs, and test scripts from the test environment. It also involves restoring the test environment to its original or desired state.
- **Perform/collect/document lessons learned**
  This task is performed in retrospectives where important lessons learned during the test process are discussed and documented. This may include findings for the entire software development lifecycle (SDLC). The lessons learned can be used for test process improvement, as described in Section 1.5.of this syllabus, Improving the Test Process.

1.2 The Context of Testing

Introduction

The context of testing encompasses the unique conditions and constraints that influence the test process, shaping the decisions and strategies for planning, designing, and executing tests. It is vital for test managers to grasp this context in order to align testing with the specific needs and objectives of the software development project. This context may differ based on the product type, industry, regulatory requirements, and crucially, the software development life cycle (SDLC) being employed.

Test managers are tasked with applying established test strategies and choosing test techniques rather than developing them. They play a key role in formulating test plans that are tailored to the project's context. By understanding and considering these various factors, test managers can ensure that the testing is pertinent, effective, and efficient in meeting the test objectives.

1.2.1 Test Stakeholders

Test stakeholders are individuals or groups with a direct or indirect interest in the product's quality. Below is a typical list of potential stakeholders, amended to reflect their diverse interests in testing:

- **Developers, Development Leads, and Development Managers:** In addition to implementing the system under test and acting on test results, these stakeholders are also involved in unit testing and contributing to the testing process.
- **Testers, Test Leads, and Test Managers:** These individuals prepare testware, which includes developing test plans and contributing to the testing process through activities such as requirements analysis, test design, test execution, defect tracking and reporting, test automation, and test progress reporting.
- **Project Managers, Product Owners, and Business Users:** They specify requirements, define the requested level of quality, and recommend required coverage based on perceived risks. They also review work products, participate in User Acceptance Testing (UAT), and make decisions based on test results.
- **Operations Team:** Engaged in operational acceptance testing, they ensure the system's readiness for production and contribute to defining non-functional requirements.
- **Customers and Users:** Customers purchase the product, while users directly utilize it. Both are key in defining requirements and should be involved in User Acceptance Testing (UAT) to validate that the product meets their needs.

This list does not include every potential stakeholders. Test managers must conduct a stakeholder analysis as part of creating the test strategy and test plan, considering the test scope in identifying specific stakeholders for their project.

1.2.2 Importance of Stakeholders' Knowledge in Test Management

In test management, it is crucial to consider the perspectives and influence of various stakeholders. The stakeholder matrix, often referred to as the power-interest matrix, guides test managers in prioritizing stakeholder engagement and managing expectations efficiently. The stakeholder matrix is a strategic tool in test management that:

- Utilizes stakeholders' expertise, with end-users and technical teams providing feedback and insights on performance and security.
- Supports risk management by highlighting stakeholder interests and influence, encouraging proactive mitigation efforts.
- Values diverse perspectives with valuable feedback.

The stakeholder matrix is composed of four quadrants:

- **Promoters (High Influence, High Interest):** key collaborators with high influence and interest, vital for shaping the test strategy and plan.
- **Latents (High Influence, Low Interest):** While they may not have a strong interest in the day-to-day tasks, their decisions are critical for resource allocation and high-level project direction.
- **Defenders (Low Influence, High Interest):** They often provide qualitative feedback and can be kept engaged through regular updates and involvement in specific discussions.
- **Apathetics (Low Influence, Low Interest):** Though not closely involved, updating them on significant milestones and seeking their input on particular issues can yield unique insights.

The test manager's role includes compiling a detailed stakeholder list and understanding each one's connection to the testing activities, using the stakeholder matrix to enhance the effectiveness of test management practices.

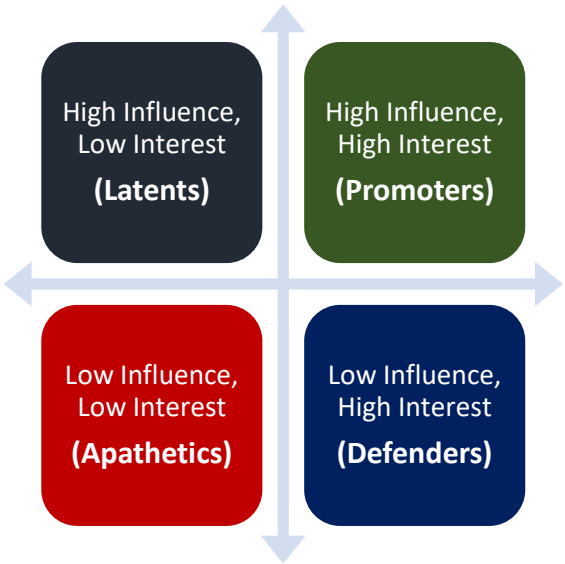

Figure 1. Different types of stakeholders

1.2.3 Test Management in a Hybrid Software Development Model

Hybrid software development models integrate elements from both traditional sequential approaches and Agile practices to suit specific project needs or organizational transitions. The following are common reasons to use a hybrid software development model, although depending on the organization and the project, there can be other reasons as well:

- **Hybrid as a transition to Agile:** Transitioning from traditional to Agile methodologies can be challenging due to the fundamental changes in workflow, culture, and team dynamics. Hybrid models provide a balanced approach that eases this transition by combining the structure of traditional methods with the flexibility of Agile practices.
- **Hybrid as fit for purpose:** Some organizations or projects may not be able to move to Agile. Projects that are high-risk may require sequential tasks for some things and Agile practices for others. They can use a hybrid model as it fits their purpose.

In a hybrid setting, test management activities may include:

- Evaluating team's understanding and capability to seamlessly transition between traditional and Agile methodologies.
- Identifying strengths and weaknesses in adapting to a hybrid approach
- Ensuring the team is adept at combining structured processes with Agile flexibility
- Enhancing collaboration between the test team and stakeholders to better manage testing within sprints and traditional test phases
- Participating in coordinated efforts, such as scrum-of-scrums for testers, to maintain focus on testing while contributing to the overall development objectives.
- Tracking and reviewing test efforts and cases within sprints to ensure they align with Agile practices.

1.2.4 Test Management Activities for Various Software Development Lifecycle Models

To properly align testing within the SDLC model, a test manager must understand the various SDLC models used in their organization and utilize that knowledge to properly align testing with the development activities.

The table below shows a comparison between various test management activities based on different SDLC models:

Aspect	Sequential Development Model e.g., V-Model	Iterative Development Model e.g., SCRUM
Estimation	Early detailed estimation for each test level.	Iterative estimation, part of story planning per iteration.
Testware	Includes strategy, plan, cases, schedule, and reports.	Focuses on acceptance criteria and definition of done; minimal documentation.
Roles	Test manager oversees decisions and team management.	Roles are integrated; facilitator or coach replaces traditional test manager.
Tools	Predominantly test management tools suited for phase-based testing.	Tools for CI/CD and automation are central, supporting continuous testing.
Testing Approach	Scheduled in advance, corresponding to project phases	Embedded within iterations, with a focus on adaptability and feedback.
Test automation	Implemented strategically, can occur at	Built-in from inception, with emphasis on

Aspect	Sequential Development Model e.g., V-Model	Iterative Development Model e.g., SCRUM
	various stages	automated regression in CI/CD
Monitoring and reporting	Milestone-based reporting, with optional automated dashboards	Continuous reporting with real-time dashboards and daily status updates
Metrics	Focuses on traditional test metrics and defect management. (e.g., test execution, defect rates).	Includes Agile metrics for iteration tracking in addition to traditional metrics. (e.g., team velocity, burndown charts)

Table 1: Test Management Activities for Various SDLC Models

1.2.5 Test Management Activities at Various Test Levels

Component testing:

- Define the scope, objectives, and completion criteria for component testing (unit testing).
- Involve testers in activities beyond traditional testing roles such as code reviews, where their analytical skills add value.
- Coordinate with the development team for issue resolution and unit test contribution.

Component integration testing:

- Determine integration sequences and test combinations in collaboration with the development team, taking into account the SDLC model, tools, and processes.
- Oversee progress to ensure it aligns with system and acceptance testing strategies.
- Manage this phase cooperatively with developers, considering component (unit) integration testing as well.

System integration testing:

- Ensure the scope and objectives of system integration testing are clear and attuned to risk assessment and quality targets.
- Maintain oversight of progress, outcomes, and issue management during system integration testing.

System testing:

- Tailor planning to the SDLC model, with careful allocation of resources, tool selection, and scheduling.
    - For Agile projects, integrate system testing with iterative story testing, avoiding distinct test phases and ensure testing is continuous and integrated. While in sequential models, testing may follow planned stages.

Acceptance testing:

- Collaborate with stakeholders to review and confirm fulfillment of acceptance criteria and plan testing activities, including managing user testing in UAT.
- Coordinate acceptance testing logistics, facilitating tests at the customer's site, to ensure the product meets the business needs and quality standards outside of the development environment.
- Facilitate the resolution of any issues with UAT, and guide stakeholders through the process of product sign-off upon meeting acceptance criteria.

1.2.6 Test Management Activities for Different Test Types

Effective test management requires an integrated approach that considers the unique demands of functional, non-functional, black-box, and white-box testing. For managers carrying out functional testing, the focus is on ensuring that all functionalities are thoroughly tested and meet the defined requirements. Non-functional testing management revolves around verifying system attributes like performance and security. Black-box testing management involves ensuring that tests are user-focused and that all possible external interactions are covered. White-box testing management emphasizes understanding the code structure and ensuring that tests thoroughly cover the internal logic.

Functional Testing Management:

- Strategic Planning and Progress Tracking: crafting a detailed test strategy that aligns with functional requirements and project objectives, as well as monitoring progress.
- Resource Coordination: Allocating human and technical resources efficiently to cover all functional aspects of the system.

Non-Functional Testing Management:

- Performance Benchmarking: Establishing performance benchmarks and managing the testing activities that assess the system against these criteria.
- Compliance Verification: Overseeing tests that ensure the system meets non-functional standards such as security, usability, and reliability.

Black-Box Testing Management:

- Test Coverage Analysis: Ensuring that black-box tests cover all user scenarios and business requirements.
- Feedback Incorporation: Managing the process of gathering feedback from stakeholders to refine black-box testing approaches and the fixing of defects.

White-Box Testing Management:

- Code Coverage Optimization: Overseeing the use of code coverage tools to identify gaps in white-box testing and directing resources to address these areas.
- Technical Insight Integration: Managing the incorporation of technical insights into the test planning process, ensuring that tests are designed with an understanding of the internal workings of the application.

1.2.7 Test Management Activities to Plan, Monitor, and Control

Effective test management is the cornerstone of any successful testing effort, encompassing a wide range of activities that necessitate careful planning, vigilant monitoring, and strategic control. Test managers play a pivotal role in ensuring that the test process is not only effective and efficient but also tailored to the unique demands of the project at hand.

Test Planning:

- **Comprehensive Scope Definition:** A test plan must be meticulously crafted, incorporating a thorough definition of scope. This includes identifying all functional and non-functional requirements to ensure complete test coverage. It also involves considering the implications of both black-box and white-box testing methodologies, ensuring that the test cases developed are capable of validating the system under test from all angles.
- **Risk Assessment and Mitigation Plan:** Integral to the test plan is a robust risk management framework. Test managers must undertake a detailed risk analysis, pinpointing potential vulnerabilities and challenges that could impact both the project workflow and the end product. The development of mitigation strategies is crucial, involving preemptive planning to circumvent or minimize these risks effectively.
- **Resource Allocation Strategy:** Resource planning is another critical element. This extends beyond mere allocation to defining the structure of the team, delineating roles, and establishing communication protocols. In environments where teams are distributed, as with onsite/offsite models, this becomes especially significant to maintain synchrony and to ensure seamless collaboration.

Test Monitoring:

- **Execution Oversight:** Monitoring plays a central role in the test management process. It involves a continual review of test execution against the established plan, tracking the progress of test cases, and managing any defects that arise. Adjusting test priorities based on risk assessments and real-time developments ensures that testing remains focused and aligned with the most critical areas.
- **Tool and Environment Optimization:** The judicious selection and usage of test tools and environments are crucial for supporting the testing strategy. Continuous monitoring ensures that they are effectively integrated within the CI/CD pipeline, facilitating continuous testing and immediate feedback loops that are vital for the Agile development process.
- **Development Collaboration:** Maintaining a close working relationship with the development team is essential for successful test outcomes. This collaboration should support a comprehensive approach to testing, leveraging insights from both white-box and black-box perspectives to preemptively address potential issues.

Test Control:

- **Adaptive Process Management:** Test control is about dynamically adjusting the testing process in response to new insights, challenges, and evolving project dynamics. It requires a test manager to be responsive and flexible, capable of implementing changes to the testing approach that reflect the current state of the project.

- **Quality Gate Management:** A structured approach to quality gate management is fundamental. This includes defining what constitutes a quality gate within the testing lifecycle and making informed decisions about the progression of the testing phase, which is instrumental in maintaining product integrity.

By focusing on these specific activities within test planning, test monitoring, and test control, test managers can ensure that the test process is well-defined, adaptable to project changes, and results in a product that meets both the project requirements and stakeholder expectations.

1.3 Risk-Based Testing

Introduction

Risk-based testing involves the identification, assessment, monitoring and mitigation of risks to drive testing. These risks are identified by diverse stakeholders and can be used to select and prioritize tests. The higher the risk level, the earlier the testing should begin, and the more intense and prolonged the test effort should be.

1.3.1 Testing as a Risk Mitigation Activity

A product risk is a potential situation where quality problems may exist in a product. When tests reveal defects, testing has helped mitigate product risk by providing the awareness of defects and opportunities to deal with them before release. When tests do not find defects, testing indicates that the product risk level is lower than expected.

Among other things, the test manager is responsible for delivering a correct and reliable evaluation of the product quality. This requires an active involvement in project risk management with a focus on project risks related to quality assurance (e.g., ambiguous requirements that lead to major issues in late validation, insufficient test environments obstructing test execution).

Risk-based testing focuses testing on the quality risks. It follows the generic risk management process, which consists of the following main activities:

- Risk analysis, consisting of risk identification and risk assessment
- Risk control, consisting of risk monitoring and risk mitigation

These main activities are logically organized in sequential order, but they can overlap.

To focus testing on quality risks, the quality risks must be identified and assessed. To be most effective, risk analysis should include diverse stakeholders. Being a principal stakeholder of the quality risk analysis, the test manager should understand and monitor these activities and be able to moderate them.

Test monitoring should include risk monitoring. In addition to monitoring the evolution of known quality risks, it should include analyzing any new quality risks and adjusting the risk register.

The test manager is one of several people who drive the mitigation of quality risks. Risk mitigation is distributed over several test activities. For example, the results of quality risk analysis are used in test planning to focus the testing on the correct areas using the correct techniques. In test analysis, risk levels guide the selection of test conditions to be covered. In test execution, risks-based prioritization governs the sequence of test execution.

1.3.2 Identification of Quality Risks

The task of the test manager is to gather the risks from the stakeholders. Stakeholders can identify quality risks through one or more of the following techniques:

- Expert interviews
- Independent assessments
- Retrospectives

- Risk workshops
- Brainstorming
- Checklists
- Referring to past experience

By involving the broadest possible sample of stakeholders, the risk identification process often identifies most of the significant product risks. Identification of which stakeholders to participate at this stage is very important. It is essential to ensure that the list of participating stakeholders is comprehensive and agreed upon with the project manager. Risk identification that misses key stakeholders can be very problematic. The key is to ensure that all the relevant stakeholders have a chance to participate. If they cannot participate, they should at least have a chance to delegate the task. When key stakeholders may not be represented, a kick-off meeting can be used to determine if they are missing.

In risk-based testing, it is important to understand that risk is not uniformly distributed within the test objects. For example, customer-facing components of a self-service application may have very different usability risks from the administration components. Identifying the individual risks of the various test items is an important task in test planning.

Risk identification often produces by-products, (i.e., identification of issues which are not product 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 focus of risk-based testing. The test manager can often play a significant role in highlighting these by-products and making it clear that quality is everyone's concern. Poor or missing requirements are often an indication of a more fundamental problem in planning and preparation, as quality assurance is involved throughout the SDLC.

1.3.3 Quality Risk Assessment

Once risks have been identified, they can be assessed. Quality risk assessment includes the categorization of risks by type (product risk or project risk) and by quality characteristics impacted.

Determining the risk level typically involves assessing, for each risk item, the risk likelihood of occurrence and the risk impact upon occurrence. Factors influencing risk likelihood for quality risks include:

- Complexity of technology, tools, or system architecture
- Maturity of the organization
- Personnel issues with skills, availability, motivation or autonomous working, including knowledge of the SDLC in use
- Conflict within the team
- Contractual problems with suppliers
- Geographically distributed teams
- Weak managerial or technical leadership
- Time, resource, budget, and management pressure
- Lack of early quality assurance activities

- High change rates of the test basis, product, or personnel

Factors influencing the risk impact include:

- Frequency of use of the affected feature
- Criticality of the affected feature
- Criticality of the affected business goal
- Damage to reputation
- Loss of business income
- Potential financial, ecological or social losses, or liability
- Civil or criminal legal sanctions
- Interfacing and integration issues
- Lack of reasonable workarounds
- Safety needs

The test manager combines risk likelihood and risk impact to determine the risk level.

If risk analysis is based upon extensive and statistically valid risk data, a quantitative assessment is appropriate. For example, risk likelihood can be expressed as a percentage and risk impact as an amount. In such a case, the risk level can be calculated as the product of these two factors. Typically, though, risk likelihood and impact can only be ascertained qualitatively on ordinal scales, e.g. as very high, high, medium, low, or very low. Risk likelihood and risk impact values are then combined through a risk matrix to create an aggregate risk level. This aggregate risk level should be interpreted as a qualitative, relative rating upon an ordinal scale.

Unless the risk analysis is based upon extensive and statistically valid risk data, the risk analysis will be qualitative, based on the stakeholders' subjective perceptions of risk likelihood and risk impact.

1.3.4 Quality Risk Mitigation Through Appropriate Testing

In software development, testing is the most important quality risk mitigation activity, and makes it possible to reduce the likelihood of failures. Other possible risk mitigation measures include a contingency plan (e.g., by providing workarounds), risk transfer to a third party (e.g., the vendor of a component), or risk acceptance.

In **test planning**, the time and effort associated with developing and executing a test should be proportional to the risk level: Testing for higher risk levels should start early and use more rigorous test techniques, while testing for lower risk levels may start later and should use less rigorous test techniques. To best mitigate the overall risk through testing, the test manager should analyze the following contextual factors and select an appropriate test approach:

- **The test items:** Different test items within a test object may have different levels of the same risk type, so a test object does not need to be tested with uniform rigor
- **The quality characteristics:** Risks affecting specific quality characteristics should be mitigated by associated test types which need specific test effort, test environments, and testing skills

- **The test levels and test types:** Certain risks may only be tested dynamically on particular test levels; others by static testing, (e.g., static analysis and code reviews for maintainability), or by a combination of both (e.g., by a review of the architecture, and dynamic testing of the integrated system for security vulnerabilities). Testing every test item as early as possible mitigates the risk of finding critical defects late in the lifecycle which would cause higher internal failure costs and delays.
- **The SDLC:** Test activities have their own specific entry criteria. Various SDLCs fulfill them at different times.
- **The test team:** The most qualified people should test the test items with the highest risk levels.
- **The regulatory requirements:** Some safety related standards (e.g., IEC 61508 standard), prescribe the test techniques and the required coverage based on the integrity level. The test manager has to ensure that these standards are followed.

In addition, the risk level should influence quality control decisions such as the use of reviews of work products like test cases, the level of independence of testing from development, and the extent of regression testing performed.

During **test monitoring and test control**, risk-based testing allows reporting on the test progress in terms of the residual risk level at any point in time. This supports the development team and stakeholders in monitoring and controlling software development, including making release decisions, based on the residual risk level. This requires reporting test results in terms of risks in a way stakeholders can understand.

During **test implementation**, test prioritization is based on the risk levels. During test execution, this ensures early coverage of the most critical areas and mitigation of the highest level risks.

- In some cases, tests are prioritized for execution in strict descending order of the levels of risk they cover, starting with the highest. This approach is called depth-first and is appropriate when it is important to mitigate the highest level risks as early as possible.
- Alternatively, at least one test for each risk is assigned highest priority. All other tests are prioritized based on their levels of risk covered. This approach is called breadth-first and is appropriate when stakeholders want an overall view of the product quality as early as possible. In practice, testing often starts with the depth-first approach, but as time becomes more limited, it switches to the breadth-first approach, testing all remaining risk items at least once.

Whether risk-based testing proceeds depth-first, breadth-first, or combined, the time allocated for testing may be consumed without all planned tests being run. At this point, risk-based testing facilitates the provision of a justified recommendation to management whether to extend testing or to accept the remaining risk.

1.3.5 Techniques for Risk-Based Testing

There are a number of specific techniques with various degrees of formality to implement risk-based testing. The suitability of a technique depends on project, process, and product considerations. There are two basic types of techniques: heavyweight or lightweight. In safety-critical systems, heavyweight techniques are very often used. In non-safety-critical applications, lightweight techniques are usually employed.

Heavyweight techniques are formal, using defined procedures and detailed documentation. They involve broad groups of stakeholders. Risk assessment within heavyweight techniques uses detailed factors of

risk likelihood and risk impact, and mathematical formulas to calculate the risk likelihood and risk impact from those factors. Examples of heavyweight techniques are:

- **Hazard analysis:** Extends the analytical process upstream by attempting to identify the hazards that underlie each risk
- **Cost of exposure:** Determines for each quality risk item the likelihood of a failure, the cost of a loss associated with a typical failure, and the cost of testing for such failures
- **Failure mode and effect analysis (FMEA) and its variants:** Identifies quality risks, their potential causes and their likely effects and then assigns severity, priority, and detection rates
- **Fault tree analysis:** Relates potential failures to defects that can cause the failure, then with errors that can cause those defects, continuing on until the various root causes are identified

In contrast, lightweight techniques are less thorough and require less effort from the test team and the stakeholders. Like heavyweight techniques, they are also based on stakeholder involvement, using the results of risk analysis as the basis for test planning and the test conditions. However, the group of stakeholders may not be that broad, and the risk factors are usually reduced to risk impact and risk likelihood on an ordinal scale. Some of these techniques, like Systematic Software Testing (SST) (Craig, 2002), can only be used when requirements specifications are provided. Other techniques, including Pragmatic Risk Analysis and Management (PRAM) (Black, 2009), Product Risk Management (PRISMA) (van Veenendaal, 2012), use the requirements and/or other specifications as an input to the risk analysis but can function entirely based on stakeholder input.

1.3.6 Success Metrics and Difficulties Associated with Risk-Based Testing

In a retrospective, the test team should measure the extent to which they realized the benefits of risk-based testing. In many cases, this involves answering some or all of the following questions through the use of metrics and consultation:

- Were the relevant stakeholders involved or represented in the risk analysis?
- Was the involvement of the stakeholders in risk analysis appropriate?
- If there have been critical incidents in production that indicate critical defects have escaped, have they been resolved?
- Were most of the high priority defects found early in test execution?
- Was the test team able to explain the test results to stakeholders in terms of risk?
- Did the skipped tests have a lower level of associated risk than those executed?

In most cases, successful risk-based testing results in an affirmative answer for all these questions. In the long term, process improvement goals for success metrics should be set, along with striving to improve the efficiency of the quality risk analysis process.

Managing risk often encounters unexpected difficulties due to complexities that are often overlooked.

- **Difficulty in assessing the risk level:** Estimating the risk impact and risk likelihood can be very difficult. Solution: use historical data and ask key project stakeholders for their assessment.
- **Keen beginnings:** Setting up and maintaining a proper risk-based test approach is often neglected in the face of high short-term pressure to succeed. Solution: regular monitoring and reporting of risk to the stakeholders.

- **Déjà vu:** The same set of risks are being raised for each project which leads to complacency towards risk. Solution: involve the right people in risk identification and mitigate only those risks that are deemed important.
- **Key risks are being missed:** The root cause of this issue is usually due to the involvement of inexperienced or inappropriate people in the process. Solution: involve appropriate people and train them.
- **Stakeholder churn:** Stakeholders may change over time and also new risks may pop up, so risk analysis is an ongoing, iterative activity and should not only be performed once at the beginning.

1.4 The Project Test Strategy

Introduction

Throughout this syllabus, the organizational test strategy is taken as given. The development and maintenance of an organizational test strategy is considered in the context of ISO/IEC/IEEE 29119-3 (where it is referred to as an “organizational test practice”) and the syllabi for the ISTQB® Expert Level - Test Management, and the ISTQB® Certified Tester Agile Test Leadership at Scale.

If an organizational test strategy does not exist or does not cover the required aspects, test management must seek to clarify the missing details with the relevant stakeholders.

In the context of this section the definition of a project test strategy is an example of any type of a detailed test strategy for a project, a release, a product, or any other type of system development or acquisition initiative. A project test strategy (referred to as a "test strategy" in the ISO/IEC/IEEE 29119-3) describes the approach to testing in a specific context so that the organization's objectives can be met, particularly those relating to product quality and test activities. A test strategy may also exist for a single test level or a test type.

The project test strategy is the main outcome of the test planning for a project and is typically documented in a test plan or as part of other documents. Documentation of the test strategy is recommended, but not necessarily in the form of a formal test plan. The need for documentation depends on the context of testing (see Section 1.2, The Context of Testing). When a project follows a sequential development model the project test strategy is usually documented, preferably in the test plan (see ISO/IEC/IEEE 29119-3). Documentation is also often required by contracts, agreements, regulatory bodies, or laws.

1.4.1 Choosing a Test Approach

The project test strategy guides all testing activities within a project and details objectives, resources, schedules, and responsibilities. This strategy must be tailored to the unique requirements of the project. Key decisions include the selection of test levels, test types, and test techniques for static and dynamic testing and other test practices (e.g., scripted testing, manual testing, back-to-back testing).

In theory, all test types can be performed at any test level, and any test technique can be applied to any test type at any test level. In practice, the appropriate selection and combination of these choices have a significant impact on the effectiveness and efficiency of testing. For example, code maintainability can often be evaluated more effectively and efficiently using static code analysis or code review. On the other hand, performance efficiency may be better evaluated through scripted system tests due to the interaction of internal components, or the usefulness of functionality may be better validated with users through collaboratively developed manual acceptance tests. Choosing the best approach for a test strategy can be a complex process that can be influenced by organizational test strategy, project context, and other aspects.

Selecting and combining test levels, test types and test techniques is therefore critical to an effective project testing strategy as it significantly influences the efficiency and effectiveness of testing.

1.4.2 Analyzing the Organizational Test Strategy, Project Context and Other Aspects

The organizational test strategy, the project context, and additional factors or constraints related to testing must be fully understood to enable the development of a project test strategy.

To choose the appropriate test approach, the following factors typically must be analyzed:

- **Domain:** The domain for which the product will be created or modified. Any domain-specific regulations, standards and practices may change the rigor of testing, the documentation required, as well as its level of detail. For example, in pharmaceuticals and medicine, the test approach often emphasizes intensive user acceptance testing focused on risks to patient health, using test cases based on functional user requirements, whereas user acceptance testing for web-based insurance applications might focus on usability and increasing the likelihood of new insurance contracts through A/B-testing.
- **Organizational goals and overarching quality characteristics:** Organizational goals can include the need to demonstrate the value of testing and to increase the degree of test automation or quality characteristics of the test process such as the maturity level of the testing or the efficiency of defect detection. These may determine the test levels and test types that need to be adhered to.
- **The project goals and type of project:** The project goals (e.g., concerning budget, time, and quality) and the type of project (i.e., customer specific versus market-oriented product development) typically contain constraints and risks, as well as opportunities that affect testing. For example, tight budget and time constraints may require the rigorous use of risk-based testing to prioritize test cases for test execution, while developing of a product specifically for a customer may require tests that cover pre-defined contractual acceptance criteria.
- **Test resources:** Any constraints regarding the availability of test resources including the test tools, test infrastructure, technology and development environment used in the project as well as available testing staff and their skills must be considered. (see Section 3.1, The Test Team). For example, experience-based testing requires testers with a good domain knowledge; mobile applications typically need to be tested on a typically limited number of different devices; the use of testing tools may be limited by the number of licenses available.
- **The software development lifecycle model used for the project:** To determine appropriate test levels, test effort, appropriate entry criteria and exit criteria, see the ISTQB® Foundation Level Syllabus V.4, Section 2.2 and 5.1. A software lifecycle with continuous integration requires more automated testing than one-off development using a waterfall model and therefore, different test types and test techniques may be used.
- **Interfaces with other systems:** In a system of systems, aligning the testing with other teams or projects and selecting appropriate test levels especially for system integration testing is essential. For example, risk-based testing helps to prioritize and scale system integration testing.
- **Availability of test data:** Constraints on the availability of test data, such as the need for anonymized test data from production, or the creation of specific test data that may be difficult to provide and needs to be validated, such as data for AI testing, must be considered. For example, model-based testing can support test data creation and management of test data.

The test manager should determine which combination of test techniques, test levels and test types should be used as the best approach to satisfy organizational test strategy, the project context, and additional factors or constraints related to testing.

1.4.3 Definition of Test Objectives

A test plan should be defined for each test project and should contain, among other aspects, the test scope, the test objectives, and the exit criteria. The test plan can be set up at the release level, as a project test plan (also known as a master test plan) and, if required, as a level test plan for the different test levels. Additionally, test plans for the different quality characteristics, such as a security test plan or a performance test plan, can be defined. In Agile software development and hybrid software development projects, an iteration test plan can be agreed upon. For each release and iteration, the scope of functional features and their non-functional characteristics to be delivered is defined in the test plan and agreed upon by the stakeholders.

Associated with the features delivered for testing in a project, the project test objectives and exit criteria must be defined. This can be done by using the S.M.A.R.T. goal methodology:

- S = specific. A project test objective and exit criterion should be clear and unambiguous.
- M = measurable. It should be quantifiable and have specific criteria for measuring progress to determine whether it has been reached.
- A = achievable. It should be feasible considering the available resources, timeframe and capabilities.
- R = relevant. It should be aligned with the overall project objectives.
- T = timely. It should have a specific timeframe and a defined deadline for completion.

The project test objectives should address all targeted aspects of quality and quantity, provided they are measurable or evaluable. Examples of project test objectives are:

- Reaching the specified exit criteria within the defined timeframe
- Meeting the quality goals of the organization (e.g., measured as a key performance indicator for the number of claims from customers for a product)
- Complying with rules and regulations of the specific industry
- Ensuring the availability of data to authorized users only (e.g., by access rights)
- Checking the functional completeness, functional correctness, performance, efficiency, portability and security of data migration
- Enhancing the level of test automation,(e.g. for regression or performance tests by a defined percentage)
- Refactoring code successfully and showing it has not introduced new defects (e.g., to remove poorly structured source code or technical debt while maintaining the existing functionality, proven by a regression test)
- Proving the security of interfaces (e.g., by validating Extensible Markup Language (XML) messages against their XML Schema Definition to ensure the rejection of malicious data)
- Checking the usability of a user interface and achieving some degree of sub-characteristic (e.g., by measuring the time it takes to complete a specific task in an online shop)

Apart from the counting and measuring of the project test objectives, the assessment of the quality level by domain level experts and stakeholders should be considered.

Depending on the project context and test objectives, sometimes multiple test environments with the available resources and/or test tools may be required. The test environments may not all be available at the same time. This needs to be considered when formulating achievable test objectives and exit criteria.

Depending on the project context, additional factors would have to be considered in defining the test objectives and test scope of the project, as described in Section 1.2 of this syllabus, The Context of Testing.

1.5 Improving the Test Process

Introduction

Testing is an important part of software development and often accounts for at least 30-40% of the total project costs. Next to the many (technical) challenges that software projects are facing (e.g., increasing complexity and size, new technologies, wide variety of devices and operating systems, and security vulnerabilities), there is a need to optimize the effectiveness and efficiency of testing as well as a need to improve test processes accordingly. Learning from existing best practices and one's own errors makes it possible to improve the test process and to make projects more successful.

An improvement process at an organizational level is typically more useful than an improvement process at a project or team level. However, it is nevertheless also possible and beneficial to apply process improvement at a project or team level but it should be tailored to the needs of the project or team. A test improvement can be initiated, for example, by dissatisfaction with the results of current tests, unexpected defects, changing circumstances, a benchmark result, or lack of communication. Different techniques are available to improve testing (Bath, 2014). Some of these techniques are described below. The techniques described in this syllabus can be applied to both sequential development models and Agile software development/incremental development models. The ISTQB® Expert Level Improving the Test Process Syllabus offers a deeper insight.

1.5.1 The Test Improvement Process (IDEAL)

Once it has been agreed upon that test processes should be improved, the process improvement implementation activities to be adopted for this activity can be defined as in the IDEAL model, which is based on similar ideas as the well-known plan-do-check-act (PDCA) cycle. IDEAL is an acronym that stands for Initiating, Diagnosing, Establishing, Acting and Learning.

Although IDEAL was originally defined to support improvement activities at an organizational level, it can also be applied at a project or an Agile software development team level. In a project context, the objectives of the activities (see hereafter) are yet to be achieved. The main difference is probably the initiating phase, which is much smaller at a project or team level than an organizational level. Diagnosing through a retrospective and establishing a plan would most likely be much smaller than for an organizational level. Acting and learning will all also be relevant at a project or team level.

Initiating the Improvement Process

At the start of the improvement process, the objectives and scope of the process improvements are agreed upon by the stakeholders.

Diagnosing the Current Situation

The current test process is assessed to identify possible improvements. The assessment is typically made against a standard framework in case of model-based test process improvement (see Section 1.5.2, Model-Based Test Process Improvement) or can be based on an analysis of specific metrics in case of analytical-based test process improvement (see Section 1.5.3, Analytical-Based Test Process Improvement Approach).

Establishing a Test Process Improvement Plan

A test process improvement plan can be a formal document that lists all of the detailed actions that must be performed to achieve improvements. Depending on the context, the plan can be highly informal and

very lightweight. The list of possible process improvements should be prioritized. The prioritization can be based on return on investment (ROI), risks, alignment with project or team strategies, and/or measurable quantitative or qualitative benefits that are to be accomplished.

Acting to Implement Test Process Improvement

The test process improvement plan for the delivery of the improvements is implemented. This typically includes training and piloting of changed processes and their full deployment in the project or team.

Learning from the Improvement Program

Having fully deployed the process improvements, it is essential to verify which benefits, either planned or unexpected, were received. Having learned what worked and did not work, we must act on this information and only thereafter the next improvement cycle can start.

1.5.2 Model-Based Test Process Improvement

One premise for both model-based test process improvement and analytical-based improvement is the assumption that the product quality is highly influenced by the quality of the processes being used and applied. When applying model-based test process improvement, one uses a test improvement model. Test improvement models are based on best practices in testing and organize test improvement in a stepwise manner.

Several recommended process models have emerged that support test process improvement. These include Test Maturity Model integration (TMMi®) and TPI NEXT®.

Model-based improvement can also be applied on a project level. In such cases the assessment and improvement process are specifically focused on the test processes or key areas defined in the model that relate to the activities at the project level (e.g., test planning and test design) and often largely omit those that are at the organization level (e.g., test policy and test organization). Alternatively, one can also appropriately tailor the practices that address the organizational level to the project's context.

For more information on model-based test process improvement see ISTQB® Expert Level Improving the Test Process Syllabus.

Test Maturity Model integration

The TMMi® (van Veenendaal, 2011 and 2020) is composed of five maturity levels. Each maturity level, except for TMMi® level 1, contains test process areas and improvement goals. In addition, to facilitate and support its implementation TMMi® contains practices, sub-practices, and examples. TMMi® was initially developed to complement the Capability Maturity Model Integration (CMMI®), but today is widely used independent of CMMI®.

To facilitate and support the update of TMMi® in Agile software development, a specific guideline has been developed that explains how TMMi® can be used and applied beneficially in Agile software development.

For more information on TMMi® see www.tmmi.org.

TPI NEXT®

The TPI NEXT® model (van Ewijk, 2013) defines 16 key areas, each of which covers a specific aspect of the test process (e.g., test strategy, test metrics, test tools and test environment). Four maturity levels are defined in the model for each of the 16 key areas.

Specific checkpoints are defined to assess each key area at each of the maturity levels. Assessment results are summarized and visualized by means of a maturity matrix which covers all key areas.

For more information on TPI NEXT® see www.tmap.net.

1.5.3 Analytical-Based Test Process Improvement Approach

Using a model-based improvement approach, as described in the previous section, improvements are introduced by comparing the test approach of a project or team to external best practices. Analytical approaches identify problems based on data from the project or team itself. Appropriate improvements can be derived from an analysis of these problems. Analytical approaches can be used together with a model-based approach to verify results and provide diversity.

Problems can be identified by using quantitative and qualitative data. Section 1.5.3 of this syllabus, Analytical-Based Test Process Improvement Approach, introduces analytical approaches that mainly use quantitative data from the test process and data from defects to assess the current approach. Section 1.5.4 of this syllabus, Retrospectives, introduces retrospectives, in which qualitative data regarding what works well and what does not work well, is collected from development and test team members.

Data analysis is important for objective test process improvement and a valuable support to purely qualitative assessments, which may otherwise result in imprecise recommendations that are not supported by data. Applying an analytical approach to improvement most often involves a quantitative analysis of the test process to identify problem areas and set project-specific goals. The definition and measurement of key parameters is required to assess the test process and evaluate whether improvements are successful.

Examples of analytical approaches are:

- Root cause analysis
- Analysis using measures, metrics, and indicators
- The GQM (Goal-Question-Metric) approach

Root cause analysis is the study of problems to identify their root causes. This allows the identification of solutions that remove the causes of problems rather than merely addressing the immediate obvious symptoms. A possible analysis procedure would involve selecting an appropriate set of defects, identifying clusters in this data, and using cause-effect diagrams (also called Ishikawa or fishbone diagrams) to identify the root causes of important defect clusters. Improvements are then derived to prevent similar defects from occurring.

Measures, metrics, and indicators are used in a quantitative manner to assess how well the test process in the project or team is performed. Key attributes of the test process to be considered are effectiveness, efficiency, and predictability. For each of these attributes, one or several metrics can be selected. By collecting and analyzing corresponding data, the key areas requiring improvement can be identified.

The GQM approach (Basili, 2014, van Solingen, 1999) provides a framework to define and analyze metrics that are tailored to the information needs of relevant project stakeholders. Measurement goals define a quality aspect of an object that needs to be measured for a particular purpose, perspective, and context. These goals are refined into questions that define the quality aspect from the stakeholders' viewpoint. Metrics are then selected that provide the necessary information to answer the question. Data collected for the metrics answer the questions, to assess the measurement goal and satisfy stakeholders' information needs.

More information on these analytical-based test process improvement approaches can be found in ISTQB® Expert Level Improving the Test Process Syllabus.

1.5.4 Retrospectives

Retrospectives are meetings in which a team reviews its methods and collaboration, captures lessons learned (good and bad), and decides on changes and actions to achieve improvements (both for testing and non-testing issues). Retrospectives address topics such as the process, people, organization, collaboration, and tools.

Retrospectives are used in both sequential development models and Agile software development. In sequential development models they are a part of test completion. In this context, retrospectives aim at generating lessons learned in order to better manage future projects. In Agile software development retrospectives are generally held at the end of each iteration to discuss what was successful and what needs to be improved, and how those improvements can be incorporated in the next iteration. Retrospectives are performed by the entire team and thus support the whole team approach and foster continuous improvement. Note that dedicated retrospectives are sometimes required to address testing issues.

A typical retrospective consists of the following steps:

Introduction: The goal and agenda of the retrospective are reviewed, and an atmosphere of mutual trust is created so that problems can be discussed without placing blame or judgment.

Collect data: Data is collected regarding what happened during the iteration or project. It is possible to collect qualitative data, such as a timeline of key events that identifies issues and lists how each team member feels about those issues. In addition, quantitative data from metrics can be presented, for example, data for test progress, defect detection, test effectiveness, test efficiency, and predictability can provide an objective insight into the testing of the project or iteration.

Derive improvements: The collected data is analyzed to understand the current situation and to generate improvement ideas. For example, root cause analysis can be applied to identify root causes of identified problems and a brainstorming session can be held in order to generate ideas on how to resolve the root causes.

Decide on improvement actions: Actions to implement the improvement ideas are derived and prioritized. An improvement plan and responsibilities are defined. Goals and associated metrics can be defined to evaluate the impact of the actions on the identified problems. Implementing too many improvements at once is difficult to manage with verifiable steps.

Close retrospective: In this last step, the retrospective itself is reviewed to identify strengths and improvements in the retrospective process. A retrospective is performed regularly, especially in Agile software development. Continuous improvement is also applied to the retrospective itself.

It is important to appropriately document the results of a retrospective. In a sequential development model, findings, conclusions, and recommendations need to be distributed and communicated in an understandable way to members of the organization. In Agile software development, problems and actions should also be documented to allow the review of actions and their potential impact on the problems in the next iteration.

Testers, being a part of the (project) team, bring in their unique perspective. They can raise testing-related problems (and others) and stimulate the team to think about possible improvements.

Further information can be found in (Derby, 2006).

1.6 Test Tools

Introduction

There are three types of business tools:

- Commercial tools
- Open-source tools
- Custom tools

When selecting a business tool, all organizational and stakeholder requirements and regulations must be considered.

There also exist technical tools such as test automation tools, test management tools and many more.

Examples for the use of test tools can be found in the ISTQB® Foundation Level Syllabus V.4.

1.6.1 Good Practices for Tool Introduction

This section contains necessary steps for the evaluation and introduction of a test tool.

A test manager may be involved in the introduction of a tool or may foster or facilitate the introduction process. Test managers are typically responsible for a dedicated test tool, or any tool related to testing such as a requirements management, defect management, or monitoring tool.

There are generic good practices and considerations when evaluating and selecting a test tool. These practices and considerations include the following:

- Identify opportunities for process improvement, with the support of appropriate tools
- Understand the technologies used in an organization and select a tool that is compatible with these technologies
- Understand how a tool is technically and organizationally integrated into the SDLC
- Evaluate the tool against clear requirements and objective criteria
- Evaluate the vendor if you are considering using a commercial tool. Evaluate support for non-commercial (e.g., open source) tools
- Identify internal requirements for coaching, mentoring, or training in the use of the tool
- Consider pros and cons of various licensing models
- As a final step, perform a proof-of-concept evaluation

Generic good practices in the adoption and roll out of a tool include:

- Run a pilot project to validate the selection criteria and requirements and to evaluate how the tool fits with existing processes and practices
- Adapt and improve processes to fit with the use of the tool, also adapt the tool to existing processes, if necessary
- Define guidelines for the use of the tool

- Provide training, coaching, and mentoring for tool users
- Roll out the tool to the organization in increments
- Implement a way to gather information from the actual use of the tool for further improvements
- Define the ownership of the tool

1.6.2 Technical and Business Aspects for Tool Decisions

Multiple factors impact the decision regarding the implementation and usage of a tool. For a test manager it is important to know and address them.

- **Regulations and security**: Organizations that develop safety-critical or mission-critical software, or are subject to regulatory compliance, may prefer commercial tools as they more often meet the required standards and often possess appropriate certification.
- **Financial aspects:** Open-source tools usually come at lower initial cost because of community support and development. Commercial tools may have a one-time purchase price as well as recurring license costs. The initial cost of a custom tool is difficult to determine because it depends on the requirements and the stage of development of the tool. Besides the initial costs, the cost for training and maintenance over the lifetime of a tool must be calculated and considered. All tools may have high maintenance and support costs.
- **Stakeholder requirements:** It is important to gather the requirements from all stakeholders to evaluate and identify the most appropriate tool. Commercial tools and open-source tools do not necessarily fulfill all requirements in detail. Custom tools can be the best choice to meet all individual requirements and in instances where no other tool provides the required functionality.
- **Existing software landscape and tool strategy:** The existing composition of tools (software landscape) and the associated tool strategy must be evaluated as there may be preferred or locked vendors, integrated systems that have dependencies with other products, or a special full-service support model for the entire software landscape with specific regulations.

1.6.3 Selection Process Considerations and Return on Investment Evaluation

Test tools can be a long-term investment, perhaps extending over many iterations of a single project, and/or applicable to many projects. A prospective tool must therefore be considered from different viewpoints.

- For the senior management, a positive ROI is required.
- For the support and operations team, a limited, but necessary number of tools used across the organization is preferred. Maintaining a larger number of tools, keeping track of their licenses, and managing the tool stack should not be cost or time-consuming.
- For the project leads, the tool must add measurable value to the project or organization.
- For the people who use the tool, usability is very important. Usability includes, for example, support for given tasks, learnability, and operability.
- For the operational staff members, maintainability is important.

Features must be analyzed for each business and technical type of tool. Different perspectives and interests have an influence on that analysis: test management, (technical) test analysis, test automation

or development. The person in the organization that is responsible for the tool (the tool owner) must make sure the analysis is accomplished, and the above-mentioned bullet points are considered.

All tools introduced into the test process should also ensure a positive ROI to the organization. It is the responsibility of the test manager to take care of the calculation and further evaluation of the ROI. In Agile software development it may be the responsibility of the entire development team.

A cost-benefit analysis should be performed before acquiring or building a tool to ensure it benefits the organization. This analysis should take both recurring and non-recurring costs into account.

Non-recurring activities and costs include the following:

- Defining and determining tool requirements to meet objectives
- Evaluating and selecting the correct tool and vendor, proof of concept
- Purchasing, adapting or developing the tool for initial usage
- Defining guidelines for the usage of the tool
- Initial training for the tool
- Integrating the tool into the existing tool landscape
- Procuring hardware/software needed to support the tool

Recurring activities and costs include the following:

- Recurring licensing and support fees
- Maintenance costs
- Ongoing training costs
- Porting the tool to different environments

Opportunity costs must also be considered. This means that the time spent on evaluating, administering, training, and using the tool could have instead been spent on actual test tasks. Therefore, more test resources may be needed before the tool can be used for intended activities.

The following risks regarding ROI should be considered when selecting tools:

- Immaturity of the organization can lead to inefficient use of the tool
- Changes in the maintenance policy of the vendor can increase workload
- Higher costs than expected
- Lower benefit than expected

The following benefits may apply to test tools:

- Reduction of manual repetitive work (e.g., regression testing)
- Speed-up of test cycle-time through automation
- Saving test execution costs by a decrease in manual activities
- Increased coverage for certain test types supported by the tool
- Reduction in human error due to fewer manual activities

- Quicker access to information about tests

Additional benefits and risks, especially for test automation tools, can be found in the ISTQB® Foundation Level Syllabus version 4 and ISTQB® Test Automation Engineer Syllabus.

In general, a test organization rarely uses a single tool. The total ROI for an organization is usually a mix of the ROI of all tools that are used for testing. Tools need to share information and work cooperatively. A long-term, comprehensive strategy for test tools that includes risks, costs, and benefits is advisable.

1.6.4 Tool Lifecycle

There are four different stages in the lifecycle of a tool. A tool administrator must be appointed to ensure that the activities of these stages are defined, carried out and managed.

- **Acquisition:** First of all, a decision has been made to select a tool. In the second step, a tool owner needs to be assigned. This person makes decisions on the use of the tool (e.g., naming conventions of work products and where these work products will be stored). Making these decisions up front can make a significant difference in the eventual ROI of the tool.
- **Support and maintenance:** The tool owner is accountable for maintaining the tool. Responsibility for maintenance activities should be on the administrator of the tool or a dedicated tools group. In the case of interoperability, data interchange and processes for cooperation and communication must be considered. Also, decisions on backup and restoration of artefacts related to the tool are required.
- **Evolution:** As time goes on, the environment, business needs, or vendor decisions can require changes to the tool. The more complex an operating environment becomes for a tool, the easier a change can disrupt its use.
- **Retirement:** At the end of its lifetime, the tool should be retired. In most cases the functionality supplied by the tool will be replaced and data must be preserved and/or archived. This may be based on a vendor decision or because it has reached a point where the benefits and opportunities of moving to a new tool exceed its costs and risks.

1.6.5 Tool Metrics

Objective metrics from tools are designed and collected based on the needs of the test team and other stakeholders. Test tools mostly capture valuable real-time data and reduce data collection efforts. This data is used to manage the overall test effort and identify areas for optimization.

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

- Test management tools can supply a variety of different metrics related to available test items, tests, planned tests as well as current and passed test execution status (e.g., passed, failed, skipped, blocked or planned)
- Requirements management tools deliver traceability regarding requirements coverage by passed and failed test cases
- Defect management tools can provide defect information such as status, severity, priority and defect density of test items. Other valuable data, such as the defect detection percentage, the test levels at which defects are introduced and detected defect lead time help to drive process improvement, but may not all be provided solely by the defect management tool.

- Static analysis tools, among others, supply metrics related to code complexity
- Performance testing tools can supply valuable information such as response times and failure rates under peak loads
- Code coverage tools help to understand which parts of the test object have been exercised by testing
- Although test tools can be used to collect metrics, they should also monitor themselves. In this context the quality of the test process can be measured (e.g., the number of defects found with and without tools and requirements coverage)
- Test efficiency (e.g., duration of test execution and number of executed tests)

More details on the collection and usage of metrics can be found in Section 2.1 of this syllabus, Test Metrics.

2 Managing the Product – 390 minutes

Keywords

anomaly, defect, defect report, defect workflow, failure, metric, test estimation, test objective, test progress

Domain Specific Keywords

planning poker, three-point estimation, Wideband Delphi

Learning Objectives for Chapter 2:

2.1 Test Metrics

TM-2.1.1 (K2) Give examples of metrics to achieve the test objectives

TM-2.1.2 (K2) Explain how to control test progress using test metrics

TM-2.1.3 (K4) Analyze test results to create test reports that empower stakeholders to make decisions

2.2 Test Estimation

TM-2.2.1 (K2) Explain the factors that need to be considered in test estimation

TM-2.2.2 (K2) Give examples of factors which may influence test estimates

TM-2.2.3 (K4) Select an appropriate technique or approach for test estimation for a given context

2.3 Defect Management

TM-2.3.1 (K3) Implement a defect management process, including the defect workflow, that can be used to monitor and control defects

TM-2.3.2 (K2) Explain the process and participants required for effective defect management

TM-2.3.3 (K2) Explain the specifics of defect management in Agile software development

TM-2.3.4 (K2) Explain the challenges of defect management in hybrid software development

TM-2.3.5 (K3) Use the data and classification information that should be gathered during defect management

TM-2.3.6 (K2) Explain how defect report statistics can be used to devise process improvement

2.1 Test Metrics

Introduction – Why Have Test Metrics?

There´s a saying in management “What gets measured, gets done.” Likewise, what is not measured is not likely to be done because it is easy to ignore. Therefore, it is important to establish a proper set of metrics for any endeavor, including testing.

Test objectives are the answer to why we test (see Section 1.4, The Project Test Strategy). To determine whether the test objectives have been met, one must define a way to measure them. Test metrics are the indicators that help us answer this question.

Test metrics can be categorized as follows:

- **Project metrics** measure progress against existing project exit criteria, such as the percentage of tests executed, passed, and failed
- **Product metrics** measure product attributes such as the degree to which the product meets the quality expectations of the intended users
- **Process metrics** measure the capability of the testing process and the effectiveness of testing. Process metrics are therefore used to report process-related effectiveness and efficiency.

More information on product and process metrics management is found in the ISTQB® Expert Test Management Syllabus.

More information on the use of process metrics can be found in the ISTQB® Expert Improving the Test Process Syllabus.

The following sections will discuss the metrics for test planning, test monitoring, test control, and test completion. These are the four main management activities related to metrics.

2.1.1 Metrics for Test Management Activities

The Advanced Level Test Management Syllabus focuses on the following generic test management activities:

- Test planning
- Test monitoring and test control
- Test completion (see Section 1.1, The Test Process).

Test management must be able to define a set of test metrics for test monitoring, test control and test completion as part of the test planning activities. Each metric needs to be defined, measured, monitored and reported.

During test planning, appropriate test metrics are defined that match the test objectives from the project test strategy.

The metrics used during test monitoring and test control may be different from those used at test completion. During test monitoring and test control, the metrics are about the progress of the test activities. In test completion, the test objectives should be achieved. One or more metrics could be combined to measure test exit criteria assigned to given test objectives.

The following table provides examples of metrics (there are many more) used in test management activities:

Metric (planned/monitored for defined milestones)	Test Monitoring and test control	Test Completion
Requirements coverage	X	X
Product risk coverage	X	X
Code coverage	X	
Actual vs. planned estimation (in hours) for testing activities	X	
Percentage of executed test cases per status (e.g. failed, blocked) vs. planned test cases	X	X
Accumulated number of resolved defects vs. the accumulated number of defects	X	
Actual automated test cases vs. planned automated test cases		X
Actual vs. planned cost of testing	X	

Table 2: Examples of Metrics Used in Test Management Activities

The metric used in a specific test activity is shown in the table. Metrics with an X in test monitoring and test control are primarily used to measure progress and are reported in test progress reports (CTFL). Metrics with an X in test completion are primarily used to measure the achievement of test objectives and are reported in the test completion report (CTFL). Metrics with an X in both columns can be used for either.

There are also metrics for monitoring test effectiveness (e.g., defect detection percentage (DDP)).

DDP is covered in the ISTQB® Expert Level syllabus, Improving the Test Process module, specifically in the Implementing Test Process Improvement section.

2.1.2 Monitoring, Control and Completion

Test metrics are indicators that show how far the test has progressed and whether the exit criteria or the related test tasks have been achieved.

Test monitoring is the activity of collecting data regarding the test and the associated evaluation and assessment. It is used to evaluate the testing progress and to check the fulfillment of the exit criteria or the associated test activities (see Section 1.1.2, Test Monitoring and Control). Exit criteria are derived from the test objectives.

Test control uses the information from test monitoring to provide guidance and corrective actions to achieve effective and efficient testing. Examples of test control directives include re-prioritizing tests when an identified risk becomes an issue, re-evaluating whether a test item meets the entry criteria or exit criteria due to rework, adjusting the test schedule to account for a delay in the delivery of the test environment, and adding new resources when and where needed.

Test completion collects data from completed test activities to consolidate lessons learned, testware, and other relevant information. Test completion occurs at project milestones such as the completion of a test

level, the completion of an iteration, the completion (or cancellation) of a test project, the release of a product, or the completion of a maintenance release.

Common test metrics used in test management activities include project progress metrics and metrics that show progress against the planned schedule and budget, the current test item quality, and the effectiveness of testing against the test objectives or iteration objectives.

2.1.3 Test Reporting

Test management should understand how to interpret and use metrics to understand and report test status. For higher levels of testing such as system testing, system integration testing, acceptance testing, and security testing, the primary test basis is typically work products such as requirement specifications, use cases, user stories and product risks. Metrics of structural coverage are more applicable to lower levels of testing such as component testing (e.g. statement coverage) and component integration testing (e.g., interface coverage). While test management may use metrics of code coverage to measure the extent to which their tests exercise the structure of the system under test, reporting of higher-level test results should be tailored to the specific context and needs of the project. For example, in frequently changing environments, code coverage metrics may be useful to monitor the impact of code changes on the test suite and identify potential gaps or risks. In addition, test management should understand that even if component tests and component integration tests achieve 100% of their structural coverage, defects and quality risks remain to be addressed at higher test levels.

The objective of reporting metrics is to provide an immediate understanding of the information for management purposes. Metrics can be reported as a snapshot of a metric at a point in time or as the evolution of a metric over time to evaluate trends.

Product risks, defects, test progress, coverage and related cost and test effort are measured and reported in specific ways at the end of the project.

The following are examples of metrics that can be used for different purposes:

Metrics related to product risks include:

- Percentage of risks of which all tests passed
- Percentage of risks of which some or all tests failed
- Percentage of risks not yet completely tested

These metrics can be used to assess the quality of the test basis and the effectiveness of the test cases in covering the product risks.

Metrics related to defects include:

- Accumulated number of resolved defects versus the accumulated number of defects
- Breakdown of the number or percentage of defects categorized by:
    - Test items or components
    - Source of the defect (e.g., requirement specification, new feature, or regression)
    - Test releases
    - Test level or iteration introduced, detected, and removed
    - Priority/severity

- Root cause
- Status (e.g., rejected, duplicated, open, closed)

These metrics can be used to monitor the defect detection and resolution process, identify the areas of high defect density or defect severity, and evaluate the test efficiency and effectiveness.

Metrics related to test progress include:

- Test execution status: Total number of tests planned, implemented, executed, passed, failed, blocked, and skipped
- Test effort: Number of actual versus planned resource hours devoted to testing

Metrics related to coverage include:

- Requirements coverage: The percentage of requirements that are covered by test cases
- Product risk coverage: The percentage of identified product risks that are mitigated by test cases
- Code coverage: The percentage of code statements, branches, paths, or conditions that are executed by test cases

Metrics related to costs and test effort include:

- Residual risks for untested components: The potential impact and likelihood of defects in the components that are not tested
- Test cost: The actual versus planned cost of testing

Additionally, it is useful to combine metrics from different categories (e.g., a metric that shows the correlation between the trends of open defects versus the trends of executed tests, or a metric that shows the quality of the test basis based on the number of defects found in the requirements). When test execution continues and fewer and fewer defects are identified, a decision can be made to terminate the tests. This decision should be based on the reporting of the metrics and the agreed exit criteria.

2.2 Test Estimation

Introduction

Project management best practices for estimation of system and software engineering exist, regarding all types of resources (e.g., the cost, people, or time). Test estimation is the application of these best practices to testing associated with a project or operation.

2.2.1 Estimating What Activities Testing Will Involve

Test estimation is a test management activity that estimates how much time, effort, and cost a task will take to complete. Test estimation is one of the major and important tasks in test management.

The main characteristics of estimation in test management are:

- **Effort** is usually calculated in person hours or story points required to finish project test tasks. Often, test effort and test duration (elapsed time) can be different, and the test management may need to estimate the total duration of the activity. How many person hours will it take?
- **Time** required to finish the project. Time is a critical resource in a project. Test planning needs to estimate test effort in calendar days and in working days. Every project has milestones and a deadline for delivery. How long will it take to finish the test project?
- **Cost** is the budget of the project. It includes the expenses for the test resources, tools, and infrastructure. What will the test project cost?

Testing is often a subproject within a (large) project, sometimes distributed over several test locations (e.g., test centers). To perform test estimation, the first step is to identify the test levels, test activities and test tasks. Next, divide the testing project into its main test activities (e.g., test planning and test execution) within the test process (see the ISTQB® Foundation Level Syllabus V.4). In Agile projects testing activities are estimated often within the development work, and not as separate values. The next step is to estimate the test effort required to finish the tasks or the work products and what the expected costs are from this.

Because testing is a sub-activity of a project, there are always some natural project constraints that influence it and require compromise, and we cannot manipulate these values arbitrarily. That can be found in the quality management as the time-cost-quality triangle. In project management, the time-cost-quality triangle comprises three values that are interdependent, meaning they are closely connected and influence each other. This relationship is commonly observed in project scenarios.

2.2.2 Factors Which May Influence Test Effort

Test effort estimation involves predicting the amount of test activities-related work that will be needed to meet the test objectives for a particular project, release, or iteration. Factors influencing the test effort may include characteristics of the:

Product:

- The quality of the test basis
- The size of the product to be tested (i.e., the test object)

- The complexity of the product domain (e.g., environment, infrastructure, and history)
- The requirements for testing quality characteristics (e.g., security and reliability)

These product-related factors can influence test estimates because they create a context specific for the system under test.

Development process:

- The stability and maturity of the organization's development processes
- The development model (e.g., Agile software development/iterative, or hybrid software development models) in use
- The material factors (e.g., availability of test automation, tools and test environments)

These development process-related factors can influence test estimates because testing is directly related to development.

People:

- People satisfaction (e.g., provided by public holidays, vacation times, other expected benefits)
- The skills and experience of the people involved, especially regarding similar projects and products (e.g., domain knowledge)

People are the most necessary resources, thus any instability should be taken into account. Therefore, people are an important factor in test effort estimation. See also Section 3.1 of this syllabus, The Test Team.

Test results:

- The number and severity of defects found during test execution
- The amount of rework required

Historical statistics support test estimation. Thus, knowing these factors will support an estimation with more accurate values.

Test context:

- The distribution of testing across several subsidiaries, the composition and location of the teams, complexity of the project (e.g., multiple sub-systems)
- The type of work (e.g., virtual or on-site)

Context-related factors are the ones that affect the entire test estimation. See also Section 1.2. of this syllabus, The Context of Testing.

2.2.3 Selection of Test Estimation Techniques

Test estimation should cover all activities involved in the test process. The estimated cost, effort, and, especially, duration of test execution is often most important to test management because these values will impact the project. However, test execution estimates tend to be difficult to generate when the overall software quality is low or unknown. In addition, familiarity and experience with the product will likely affect the quality of the estimates. A common practice is to estimate the number of test cases derived from the test basis (e.g., requirements or user stories). Assumptions made during test estimation should always be documented as part of the estimate.

Test estimation techniques or approaches can be categorized as either metric-based or expert-based.

Further details about test estimation techniques are explained in the ISTQB® Foundation Level Syllabus V.4.

In most cases, the estimate, once prepared, must be delivered to project management, along with a justification. Frequently, some input parameters are changed (e.g., test scope) often resulting in adjusting 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 keep in mind that any estimate is based on the information available at the time it is prepared. Early in a project, the information may be quite limited. In addition, this information may change over time. To remain accurate, estimates should be updated to reflect new and changing information.

The selection of the estimation technique depends on various factors, such as:

- **Estimation error:** Some techniques provide a way to calculate the standard deviation, which is a measure of the uncertainty or variability of the estimate. For example, the three-point estimation technique uses the optimistic, pessimistic, and most likely estimates to calculate the expected value and the standard deviation of the estimate (see ISTQB® Foundation Level Syllabus V.4, Section 5.1.4 for more details).
- **Data availability:** Some techniques require historical data from previous or similar projects which may not be available or reliable. For example, estimation based on ratios and extrapolation rely on historical data to derive the ratios or the trends for the current project.
- **Expert availability:** Some techniques require the involvement of experts who have the knowledge and experience to provide accurate and realistic estimates. For example, the Delphi method and planning poker rely on the opinions and judgments of experts or team members.
- **Knowledge in modeling:** Some techniques require the use of mathematical models or formulas to calculate the estimates, which may require some skills and knowledge in modeling. For example, extrapolation and three-point estimation use formulas to derive the expected value and the standard deviation of the estimate.
- **Time constraints:** Some techniques require more time and effort to perform than others, which may affect the feasibility and suitability of the technique. For example, planning poker is easy to be done, while extrapolation may be more difficult.

This shows that the selection criteria of the proper test estimation techniques are highly dependent on the context of testing (e.g., SDLC, stakeholders, test levels and test types used in the project) (see Section 1.2 of this syllabus, The Context of Testing). The test manager must be able to coordinate and to apply the test estimation techniques, (e.g. with different SDLC models in one project over different subsidiaries).

For example, to select the proper estimation technique, first determine the complexity of the topic. If the complexity is low, then metric-based techniques could be used. If the complexity is high, then expert-based techniques could be used. If a sequential development model is used, then the Wideband Delphi estimation technique could be used. If an Agile software development model is used, then planning poker could be used.

2.3 Defect Management

Introduction

The ISTQB® Foundation Level Syllabus V.4 describes the activities which begin after observing actual results that differ from expected results. The syllabus refers to these activities as defect management. Other standards use the term “incident management” ISO/IEC/IEEE 29119-3 Standard or “anomaly management” (TMAP) to emphasize the fact that at the beginning of the process we may not know if the discrepancy is caused by a defect in a work product, or due to something else (e.g., test automation failure or a misunderstanding of the requirements by the tester). Defect management and the tool used to manage defects are of critical importance to the testers and to other team members involved in software development. Information from an effective defect management process allows the test team and other project stakeholders to gain insight into the state of a project throughout its SDLC. Defect management is also crucial for deciding which defects will be fixed. This ensures that effort is spent on working with the correct defects. Collecting and analyzing defect-related data over time can help to locate areas of potential improvement both for testing and for other processes within the SDLC (e.g., better defect prevention by improved architecture and technical design).

In addition to understanding the overall defect lifecycle and how it is used to monitor and control both the software development and testing processes, the test manager, and the testers (or whole Agile team in Agile software development) must also be familiar with which data is critical to capture. The test manager must be an advocate of the proper use of both defect management process and the selected defect management tool.

2.3.1 Defect Lifecycle

Each phase of the SDLC 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 into subsequent activities. The earlier each defect is detected and removed, the lower the overall cost of quality for the product. Cost of quality is minimized when each defect is removed within the same phase in which it was introduced (i.e., when the software process achieves perfect phase containment).

During static testing we search for defects. During dynamic 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. When an anomaly is observed, further investigation should be performed. This investigation usually starts with completing a defect report in accordance with the defined test and defect management process. A failed test does not always result in the creation of a defect report. (For example, in test-driven development, where component tests, usually automated, are used as a form of executable design specification). Until the development of the component is complete, some or all of the tests must initially fail. Until the development of the component is complete, some or all of the tests must initially fail. Therefore, the result of such a test is not necessarily caused by a defect and is typically not tracked via a defect report.

A defect report progresses along a workflow (for simplicity and consistency with most defect management tools we will further use the term “defect workflow”) and moves through a sequence of defect states. In most of these states, one person owns the defect report and is responsible for carrying out a task (e.g., analysis, defect removal, or confirmation test). The following diagram represents a simple defect workflow:

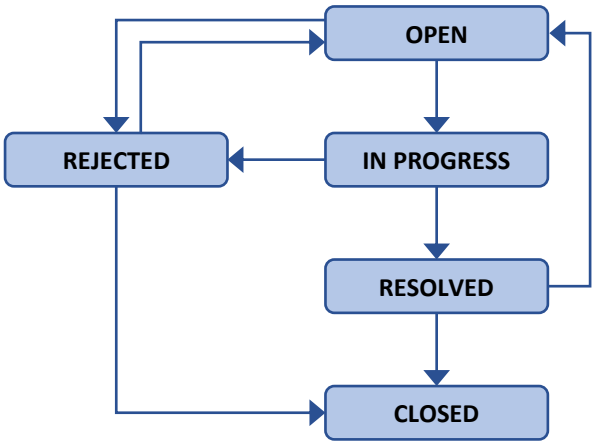

Figure 2: A Simple Defect Workflow

A simple defect workflow may cover the following defect states:

- OPEN (may be called NEW): The initial state when the defect report is created.
- IN PROGRESS: The team is working on the defect report analysis and/or fix.
- REJECTED: A defect report is rejected by the person who processed it (usually a developer or an analyst). There may be many reasons for rejection (e.g., invalid information, incorrect test, duplicate defect report) and this information is added to the defect report.
- RESOLVED (may be called FIXED, READY FOR RETEST): A tester runs a confirmation test often following the steps to reproduce the failure from the defect report itself to determine whether the fix has indeed resolved the defect.
- CLOSED: The defect report has reached its terminal state, and no further work is intended to be done. The tester transitions the defect report to this state either after a successful confirmation test or to acknowledge rejection of the defect report.

A simple defect workflow is used in many organizations and is extended by the use of other defect states relevant for a given context (e.g., RE-OPENED, ACCEPTED, CLARIFICATION, or DEFERRED).

The defect workflow may vary in different organizations in terms of different names of the defect states, rules for transitions among defect states and roles responsible for tasks in given defect states. Often the defect workflow is more simple in Agile software development than in sequential development models. The defect workflow should be adapted to a given context. When designing the defect workflow, it is advisable to respect several good practices:

- If possible, the defect workflow should be defined organization-wide to provide unified defect management across all projects
- Duplicate and false-positive defects should be represented by a separate state or a combination of the REJECTED status with choosing the reason for rejection. They may be helpful in further defect analyses with the aim of improving the test process.

- It is recommended to use only one terminal state (e.g., CLOSED). The transition to this state often requires choosing a reason for closure, useful for process assessment and process improvement activities.
- The names of states in the defect workflow should be the same as for analogous states used for other entities (e.g., user stories and test tasks) to simplify working with them.
- Consecutive defect states should belong to different responsible roles. If two or more consecutive states belong to the same responsible role, there should be a good reason (e.g., to measure the time spent in a defect state).
- Each defect state, except for the terminal state, should have more than one outgoing transition to allow the responsible role a decision regarding the next step. Exceptions of this rule should be justified (e.g., to monitor time spent on a given activity).
- The set of attributes required to be entered when performing a state transition should be limited to those which give substantial value to defect management.

2.3.2 Cross-functional Defect Management

Although the test organization and the test manager often own the overall defect management process and the defect management tool, a cross-functional team is generally responsible for managing the defects for a given project. This team, sometimes called the defect management committee, may include the test manager, representatives of development, suppliers, project management, product management or product owner and other stakeholders who have an interest in the software under test.

As anomalies are discovered and entered into the defect management tool, the defect management committee should determine whether each defect report represents a valid defect and whether it should be fixed (and by which party in case several development teams are participating in delivery), rejected, or deferred. This decision requires the defect management committee to consider the benefits, risks and costs associated with fixing the defect. It is beneficial to discuss the consideration in a meeting (often called the triage meeting). If the defect is to be fixed, the team should establish the priority of fixing the defect relative to other tasks. The test manager and test team may be consulted regarding the relative importance of a defect and should provide the available objective information.

On very large projects the appointment of a full-time defect manager may be justified by the effort needed to prepare for and to follow-up on the decisions made by defect management committee meetings, at least during those SDLC phases when testing is at its most intensive. In other situations, several large projects may share a defect manager.

A defect management tool should not be used as a substitute for good communication, nor should a defect management committee be used as a substitute for effective use of a good defect management tool. Communication, adequate tool support, a well-defined defect workflow (incl. defect report properties) and an engaged defect management team are all necessary for effective and efficient defect management.

2.3.3 Specifics of Defect Management in Agile Teams

Defect management in organizations using Agile software development is often lightweight and/or less formal than in sequential development models. If Agile teams are co-located or have well established communication means available, information about a defect or failure is often exchanged among testers,

customer representatives and developers without a formal defect report. Defect reports should, however, be created for:

- Defects that block other current sprint activities (i.e., development, testing, or other) and could not be fixed immediately within the Agile team
- Defects that cannot be resolved within the same iteration. Some Agile teams have a rule of creating a defect report if the defect cannot be resolved during the day the failure is found.
- Defects which must be resolved by or in cooperation with other teams in multi-team organizations
- Defects that must be solved by a supplier
- Defects where a defect report is explicitly requested (e.g., when a developer cannot immediately work on a fix)

Common practice is to add defects that cannot be resolved within the same iteration to the product backlog so that they can be prioritized among other defects and user stories for a later iteration.

Although the foundations of defect management should be set in an organization's test strategy, many aspects including the level of formality, triggers for creation of a defect report, and defect attributes to be captured may be left to agreement among Agile team members. In general, the level of formality of defect management and the approach to creating defect reports should reflect the following:

- Co-location of team members
- Distribution of team members across time zones
- The number of teams that cooperate on product development
- Maturity of the team(s)
- Size of the team(s)
- Risks associated with the product
- Regulatory, contractual, or other requirements (if and where applicable)

The final decision of the Agile team regarding the details of defect management should always be documented (e.g., with guidelines in a knowledge management tool).

2.3.4 Defect Management Challenges in Hybrid Software Development

In practice, multiple teams often collaborate on the delivery of the system or system of systems. Examples include hybrid software development when a customer uses Agile software development and one of their suppliers uses a sequential development model, or when an organization using a sequential development model requires delivery of a subsystem from a team using Agile software development. Such a multi-team environment poses various challenges:

- **Alignment on defect attributes and tools to be used for defect management:** In an ideal scenario all teams use one defect management tool. In practice it is common for each team to use a different defect management tool, especially when several supplier teams contribute to project delivery. In such cases it is good to establish synchronization between the defect management tools (preferably automatically).

- **Prioritization of defects:** The product owner(s) should be included in defect management meetings and actively seek information about consequences and risks associated with defects. Defect management meetings should be held more often with Agile software development than in sequential development models to keep pace with the Agile team's faster rate of product increment delivery. These meetings may, however, be shorter with Agile teams. Sometimes it is beneficial for a smaller group of defect management stakeholders to have the final word about prioritization of defects.
- **Alignment and transparency of the test plan for new development and defect fixes**: All teams' work should align to the same project plan irrespective of whether they are using Agile software development or sequential development models. All deliverables, including defect fixes, should be aligned with this project plan. Better alignment can be achieved by active participation of the members of all teams in the planning process (e.g., participation of sequential development model teams in Agile software development meetings where defects are discussed and prioritized). Transparency of development plans can be improved by sharing them between teams (e.g., via dashboards, or via the Product Backlog).

2.3.5 Defect Report Information

The information on a defect report should suffice for the following purposes:

- Management of the defect report through the defect lifecycle
- Assessment of overall project status, especially in terms of product quality, and test progress
- Assessment of the status of a product increment in terms of product quality
- Assessment of process capability

The information needed for defect management and project status can vary depending on when the defect is detected in the SDLC. In addition, defect reports related to non-functional quality characteristics may need more information (e.g., load conditions for performance issues). However, the core information gathered should be consistent across the SDLC and ideally across all projects in an organization to allow for meaningful comparison of defect data throughout the project and across all projects.

Many data items can be collected in a defect report. The test manager should decide which information is appropriate for effective defect management for a given project context. Due to the fact that each additional attribute increases the time spent on defect reporting and may increase confusion by the person who is entering the defect report, it is advisable to only collect data that is needed for defect management in the given context and/or will be used for process improvement.

To manage the defect report in most environments, the following are mandatory:

- A defect title with a short summary of the anomaly
- A detailed description of the anomaly preferably including steps to reproduce the failure
- Severity of the impact on the system under test and/or the product stakeholders
- Priority to fix the anomaly

Additional important data items are often created by the defect management tool:

- Unique identifier for the defect report
- Date/time of creation of the defect report

- Name of the person who discovered and/or reported the anomaly
- Project and SDLC phase in which the anomaly was discovered
- Current state of the defect report
- Current owner (i.e., the person currently assigned to work on the defect)
- Change history such as the sequence of actions, including date/time information, taken by project team members to isolate, repair and confirm the defect as fixed
- References (e.g. to test case, to connected defects).

Depending on context, further information (e.g. traceability) may also be collected in a defect report (see the ISO/IEC/IEEE 29119-3 for more information). The following bullet points group the information according to the intended purpose:

- **To help defect resolution**: The subsystem or component in which the defect lies, the specific test item and its release number in which the anomaly was observed or the test environment in which the defect was observed
- **To assess the overall project status:** Information to monitor progress, (e.g., risks, costs, opportunities, and benefits associated with fixing or not fixing the defect, a description of any available workaround, or requirements affected by the defects)
- **To assess the status of a product increment in terms of product quality:** The type of defect (usually corresponding to a defect taxonomy), the work product in which the defect was introduced, or the quality characteristic/sub-characteristic affected by the defect
- **To assess the process capability**: Information to monitor the effectiveness and efficiency of the development processes (e.g., the SDLC phase of introduction, detection, and removal for the defect or defect root cause)

2.3.6 Defining Process Improvement Actions Using Defect Report Information

As discussed in Section 2.3.5 of this syllabus, Defect Report Information, defect reports can be useful for project status monitoring and reporting. While the implications of metrics on the test process are primarily addressed in the Expert Test Management Syllabus, at the Advanced Test Management Level, test managers should be aware of what defect reports mean to assessing the capability of the software development and testing processes.

In addition to the test progress monitoring information mentioned in this syllabus, in section 2.1.2, Monitoring, Control and Completion, and in section 2.1.3, Test Reporting, defect information should support process improvement initiatives as discussed during retrospectives. Examples include:

- Using information about the phases of introduction, detection, and removal of defects to assess phase containment and/or perform cost of quality analysis with the aim of suggesting ways to improve defect detection effectiveness in each phase and minimize the cost associated with defects
- Using information about the phase of introduction for analysis of the phases in which the largest number of defects are introduced, to enable targeted improvements for defect prevention
- Using defect root cause information to determine the underlying reasons for defect introduction, to enable process improvements that reduce the total number of defects

- Using defect location information to perform defect cluster analysis, to better understand technical risks (for risk-based testing) and to enable re-factoring of troublesome components
- Using information about re-opened defects to assess the quality of debugging implementations
- Using information about duplicate and rejected defects to assess the quality of the defect report creation
- Enable process improvements that reduce the total number of defects by introducing pro-active measures to avoid errors upfront

The use of metrics to assess the test process effectiveness and efficiency is discussed in the Expert Test Management Syllabus.

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

3 Managing the Team – 225 minutes

Keywords

appraisal, cost of quality, defect, defect prevention, external failure, failure, internal failure

Learning Objectives for Chapter 3:

3.1 The Test Team

TM-3.1.1 (K2) Give examples of typical skills needed by test team members within four areas of competence

TM-3.1.2 (K4) Analyze a given project context to determine required skills for test team members

TM-3.1.3 (K2) Explain typical techniques for skill assessments for test team members

TM-3.1.4 (K2) Differentiate between the typical approaches for developing skills of test team members

TM-3.1.5 (K2) Explain skills required to manage a test team

TM-3.1.6 (K2) Give examples of motivating and hygiene factors for test team members

3.2 Stakeholder Relationship

TM-3.2.1 (K2) Give examples for each of the four categories determining the cost of quality

TM-3.2.2 (K3) Apply a cost-benefit calculation to estimate the added value of testing for stakeholders

3.1 The Test Team

Introduction

Any team that performs test tasks is made up of individuals with different competencies. While in some organizations teams are self-organized, in others test managers recruit and develop these teams. The right mix of skills[1] is a critical factor for all teams to successfully complete testing tasks.

The skills required by a test team member may change over time. It is important to select the correct people for the test team and to provide adequate training and growth opportunities. In addition, people outside the test team may provide additional specific skills.

This section looks at the fundamental process of analyzing and developing required skills of test team members, as well as the skills that are required to lead or to coach a test team. This also includes knowledge of factors that motivate or demotivate test team members and other factors to ensure successful teamwork.

Each individual already has skills and can develop these skills further through various ways such as work experience, education, and training. The ideal test team has all of the necessary skills for given test tasks or it is only responsible for tasks of which it has the required skills. To be successful, a test team needs various skills at different levels. Depending on the project context, some skills will be more important or necessary than others. It may make sense to bring in external experts for specific testing tasks that are beyond the capabilities of the test team.

3.1.1 Typical Skills within Four Areas of Competence

The skills of a person can be classified into four areas of competence (Sonntag, et al., 2005; Erpenbeck, et al., 2017)[2]:

- **Professional competence:** Consists of skills to perform specialized tasks. Examples include skills in test techniques, technological, and business expertise in the application domain, as well as project management skills.
- **Methodological competence:** Includes general skills that a person can use independently in a domain and that enable the independent performance of complex or novel tasks. Examples include analytical, conceptual, and judgmental skills.
- **Social competence:** Includes skills related to communication, cooperation, and conflict management in intra and intercultural contexts. They enable one to relate to others in order to act appropriately in a given situation and to achieve individual and shared goals. Examples include communication skills, conflict resolution skills, ability to work in a team, adaptability and assertiveness.

[1] The term "skill" is used as an umbrella term for skills themselves, for having knowledge of something, and for having the ability to do something.

[2] The four areas of competence used here are based on the model described in these references, which is widely used. There are other models described in the literature that group skills differently. These are not part of this syllabus.

- **Personal competence:** Includes the ability and willingness to develop oneself and to develop one's own talent, motivation, and willingness to perform as well as the development of specific attitudes and an individual personality. Examples include self-management, personal responsibility, ability to receive criticism, reliability, resilience, ability to act with confidence, discipline, openness to changes, willingness to help and to learn, and the ability to delegate.

All areas of competence are important for the success of any testing team. As the methodological, social, and personal competence is not specific to testing, the ISTQB® focuses on the development of professional competence. This includes skills in managing test tasks, analyzing the test basis, designing tests, identifying and analyzing risks and developing, configuring and maintaining test data, test environments and test scripts.

3.1.2 Analyze Required Test Team Member Skills

Staffing is an activity within test planning. This includes the task of identifying the roles and skills of the staff required to implement testing in the test strategy. A detailed context analysis is required to determine the required skills for a project.

Professional and methodological competence

For testing, the focus is on the test skills required for the test tasks. Below are some examples:

- Test planning requires conceptual knowledge for developing a test strategy.
- Test monitoring and test control require project management skills for managing all test tasks.
- Test analysis requires analytical skills in analyzing the test basis and the product risks.
- Test design requires skills in test techniques to design test cases and conceptual knowledge for designing the test environments.
- Test implementation requires judgment skills for the selection of tests, and technical expertise for test script programming and setting up test environments.
- Test execution requires technical expertise to execute automated tests, performing exploratory testing, and evaluating test results.
- Test completion requires the ability to communicate project outcomes and personal responsibility for decisions made.

Different test types and test levels require different skills (e.g., business expertise in the application domain to assess the functional suitability of a system, or technical expertise to assess the maintainability of the code).

In addition, the project context provides valuable information about the required professional competence:

- The system domain requires business expertise in the application area such as information technology, the automotive industry, or the gambling industry.
- The software and system architecture and technologies used in the project require, for example, technical expertise in programming languages, interface technology or security vulnerabilities.
- The SDLC requires, for example, knowledge about test levels, testing roles, and specific test techniques.

Social competence

In the context of testing, social competence enables test team members to behave appropriately in relationships with other team members and to achieve the test objectives. In particular, it includes communication, cooperation, and conflict resolution skills (e.g., dealing constructively with sub-optimal test conditions or reporting defects to developers).

Software development and software testing are typically done by different members of (different) groups who coordinate their tasks through communication. Communication skills, the ability to work in teams and the ability to resolve conflicts are required for project success. However, the required level of social skills may differ depending on the project context. For example, Agile software development may require higher demands on social skills than document-centric sequential development models as well as offsite testing.

Personal competence

The effectiveness and efficiency of test team members also depends on their ability and willingness to develop themselves, their skills, and their attitudes. For example, working in a self-organized Agile team may require a higher level of self-management and discipline from all team members, while a test manager of a hierarchical test team, for example, needs to be able to delegate work. A high degree of reliability and resilience is often required, particularly in time-critical projects. In addition, willingness to help, to learn and openness to change are important in a change process in all SDLC models.

3.1.3 Assessing Test Team Member Skills

In many cases, test teams are formed with existing staff. To understand the capabilities of the team members and the need for personal development, test management needs to assess the existing test team skills and compare these with the required skills, which may be documented in a skills matrix.

There are models to help teams and team members work more effectively (e.g., Meredith Belbin's "Team Roles", DISG® or PCM®). According to Belbin (Belbin, 2010) teams work effectively when they are made up of different personality and role types. These models help teams identify what skills they have, and what skills they may be lacking.

The professional and methodological competence of test team members can be assessed by demonstrating typical test tasks:

- Outlining a test strategy and discussing feedback with colleagues
- Reviewing the test basis and communicating the findings, which may also reveal communication skills
- Determining test techniques to achieve specific test objectives for a given project context.
- Applying various test techniques appropriately
- Writing a test completion report that includes an assessment of the test results

In addition, skills can be assessed through external credentials, certifications, work experience, and degrees.

Especially in Agile software development, teams identify skills that they required by regularly participating in retrospectives and receiving feedback. Experienced coaches or mentors help them to develop their skills and to identify and resolve knowledge gaps.

3.1.4 Developing Test Team Member Skills

A test team may not have all of the skills required at the start of a project. While a perfect set of individuals may not be available, a strong team can balance the strengths and weaknesses of the individuals.

The test management or the test team can identify necessary development needs by comparing required with available skills in a skills matrix. On this basis, they can determine the approaches to competence development:

- Training and education teach predefined knowledge and practices, usually in a (virtual) classroom, (e.g., sending people to a training course, having training sessions in-house, developing custom training, or using live e-learning courses).
- Self-study is a way of learning about a subject that involves studying alone, rather than in a (virtual) classroom, (e.g., reading books, watching recorded videos, or researching internet resources).
- Peer learning in which colleagues share knowledge, ideas, and experiences and learn with and from each other.
- Mentoring or coaching are approaches where a team member who is new to a role gets individual guidance by a coach, or knowledge, skills and/or experience from an experienced mentor. The experienced person acts as an ongoing resource to provide advice and assistance.
- Training on the job is also well known and a mixture of self-study, peer learning and mentoring or coaching.

Not all approaches to competence development are equally effective and efficient. Self-study and training, for example, are well suited for developing professional and methodological competence. Because of this, basic knowledge in testing can be developed by participating at ISTQB® training sessions or by self-study based on the ISTQB® syllabi. However, for developing social and personal competence, it is recommended to use approaches such as training and coaching, which are often more promising than self-study. Social exchange, feedback and reflection are among the key success factors to develop social and personal competence.

3.1.5 Management Skills Required to Manage a Test Team

Anybody who wants to successfully lead a test team must have management skills. These include professional and methodological competence in fundamental management activities, (e.g., planning, monitoring progress, controlling, and reporting). Specific test management knowledge and skills are required for testing (e.g., knowledge of different test approaches, the development of test strategies, or the use of test techniques or the applied SDLC).

Leading or coaching a test team means acting appropriately in relationships with other test team members and having the ability and willingness to develop in changing circumstances. For this reason, social and personal competence are essential success factors for leading a test team. This includes resilience, the ability to delegate, and the capacity to be accepted by the test team. Further, it includes the ability to assert test interests in the project, to promote benefits of testing, and to communicate and resolve conflicts with all stakeholders.

To acquire people for the test team, skills in analyzing social, team and working conditions is required. These skills help to ensure that the test team fits the working conditions or, if possible, that the working

conditions are adapted to the test team. In addition, test teams are subject to dynamic development processes and therefore require situational skills. (e.g., according to the phases of the Tuckman model of small group development) (Tuckman, 1965; Bonebright, 2010):

- Willingness to help test team members join into the test team (Forming)
- The ability to resolve conflicts within the test team (Storming)
- Discipline and target-oriented leadership to ensure agreed upon values and rules (Norming)
- The ability to delegate in order to give the test team personal responsibility (Performing)
- The ability to act with appreciation and confidence with departing test team members (Adjourning)

3.1.6 Motivating or Demotivating Factors for a Test Team in Certain Situations

Satisfied and motivated test team members increase productivity and performance and therefore have a significant impact on the success of projects. When this is achieved, cross-training takes place informally, the test team members can manage their own workload, and test management has more time to deal with external issues. The motivation-hygiene theory (Herzberg, et al., 1993) distinguishes between motivators and hygiene factors:

Motivators are perceived consciously and can lead to growth and satisfaction. This can include:

- Recognition and appreciation for the work done (e.g., incentives and any other individual approach that test team members find appreciative)
- Increased responsibility and autonomy (e.g., to define the test processes in a test team)
- Interesting, meaningful, and challenging tasks that test team members perceive as attainable and at the same time worth striving for (e.g., the selection and introduction of a new tool for test automation)
- Professional advancement and development (e.g., the development of the experienced tester to a test manager or test process owner)

Hygiene factors are usually taken for granted. Fulfilling them does not automatically lead to greater satisfaction. If missing, they can have a demotivating effect on the test team members:

- Appropriate remuneration (e.g., market salary, paid overtime, good social benefits)
- Appreciative personnel policy and management style (e.g., lean management, realistic targets, protection against external access and overload)
- Pleasant working conditions (e.g., unambiguous specifications, mature test objects, adequately fixed defects, appropriate workplace, a stable test environment)
- Safety as an existential need (e.g., safe workplace and adhered to agreements)
- Good interpersonal relationships (e.g., with co-workers, supervisors)

Consequently, test management should continuously eliminate demotivating factors and at the same time create and strengthen motivating factors.

Further information can be found in (Belbin, 2010, Marston, 1999, Kahler, 2008).

3.2 Stakeholder Relationships

Introduction

In test management, it is important to optimize testing to deliver good business value. Excessive testing can result in unreasonable delays and costs that outweigh the benefits, while insufficient testing can lead to delivering a product of low quality to users. The optimal approach lies between these two extremes. It is the test manager's responsibility to help stakeholders understand this balance and the added value of testing in achieving this balance, while also keeping in mind, for example, the typical time constraints of a project.

3.2.1 Cost of Quality

The benefits of testing are offset by quality costs. A means of quantifying the total cost of quality-related efforts and defects is called cost of quality. Cost of quality involves classifying project and operational costs into four categories related to product defect costs:

- **Defect prevention costs:** The cost of all activities that are planned and proactive to prevent poor quality (e.g., qualification of the developers for their tasks such as training in the creation of maintainable or secure code, reviewing the test basis as early as possible, and appropriate communication within the team)
- **Appraisal costs:** The cost of all activities aimed at defect detection (e.g., performing static testing and dynamic testing and reviewing work products)
- **Internal failure costs:** The cost of all reactive activities (e.g., fixing defects found during testing, providing workarounds)
- **External failure costs:** The cost of all non-value added and reactive activities (e.g., loss of either revenue, assets, human health, human life, or the environment, legal costs related to defect fixing, testing, deployment, and support because of defective product being delivered to the customer ("post release"), fixing field defects (flagged by customers)).

The total appraisal costs and internal failure costs are usually significantly less than the external failure costs. This therefore makes testing extremely valuable. By determining the costs in these four categories, test managers can create a convincing business case for testing.

There are more approaches that can be considered for defining the cost of quality. The ISTQB® syllabus supports two of them. This syllabus is based on the Feigenbaum's approach, and the ISTQB® Foundation Level Syllabus V.4 presents Boehm's approach (see the ISTQB® Foundation Level Syllabus V.4, Section 1.3, Testing Principles). These two approaches have been selected to reach a broader understanding of the cost of quality. Feigenbaum's approach (Feigenbaum, 1956) considers quality as a customer-oriented and company-wide process, while Boehm's approach (Boehm, 1979) focuses on the trade-off between the cost of prevention and the cost of failure in software development (Hadjicostas, 2004).

3.2.2 Cost-benefit Relationship of 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 operational details of testing (i.e., aspects specific to the test task or test level), while ignoring the larger tactical and strategic (higher level) issues related to testing, that other stakeholders, especially managers, care about.

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

- **Qualitative benefits** include improved reputation for quality, smoother and more predictable releases, increased confidence, protection from legal liability, and a reduction of risk of loss of entire missions or even lives
- **Quantitative benefits** include defects found or prevented or fixed prior to release, defects that were found to be known prior to release, (i.e., not fixed but documented, perhaps with workarounds), cost benefits (Bohm 1981, Böhler 2008), reducing risk level by running tests, and delivering information on project, process, and product status

An additional benefit of testing is that all stakeholders get adequate information to make informed decisions as to whether the quality of the product is sufficient to go live, with or without defects. Sometimes going live with known defects is much better than waiting to go live until the defects are resolved. In these cases in which a defect can be tolerated, it is heavily dependent on the probability of occurrence and severity of the defect.

The cost of quality per defect of testing is calculated as follows:

Average Savings per Defect = Average of External Failure Costs - (Average Appraisal Costs + Average of Internal Failure Costs)

Total cost of quality = (Defect prevention costs + (Appraisal costs * Number of defects found before release)) + ((Internal failure cost * Number of defects found before release) + (External failure costs * Number of defects found after release))

As an example, assume that you have calculated the following cost of quality per defect for a product:

- Defect prevention costs: $180
- Average appraisal costs per defect: $500
- Average of internal failure costs per defect: $200
- Average of external failure costs per defect: $4,000

The average defect prevention costs, appraisal costs and internal failure costs are calculated using the number of defects found prior to release, while the average external failure costs are calculated using the number of defects found after release. With these values, we can calculate the average savings per defect as follows:

Average Savings per defect = $4,000 – ($500+ $200) = $3,300.

Boehm´s curve is a graphical representation of the costs of fixing defects over time in the SDLC. This concludes that testing should be done earlier in the SDLC to reduce the cost of fixing defects. Boehm's curve shows that internal failure costs, or the cost of fixing a defect, increases the later a defect is discovered in the SDLC. Using this information the test manager should strive to find the optimal relationship between defect prevention costs vs. internal and external costs.

The test effort must be based on the specific risk of the project and product and the risk a business is willing to take. Too much testing can cause higher costs than the benefit of risk level reduction. If too little is tested, missed defects can pose a high risk to generate higher costs than the omitted tests would have costed. Risk-based testing (see Section 1.3 of this syllabus, Risk-Based Testing) supports the cost-benefit relationship of testing by investing test effort levels proportional to the risk levels, and by prioritizing tests based on their risk levels.

Test managers should understand which of these benefits and costs apply for their organization, project, and/or operation, and be able to communicate the added value of testing in terms of these benefits and the cost of quality per defect.

4 References

Standards

IEC 61508 (2010) Functional safety of electrical/electronic/programmable electronic safety-related systems - Parts 1 to 7

ISO/IEC/IEEE 29119-2 (2021) ISO/IEC/IEEE 29119-2 Software and systems Engineering-Software testing-Part 2 Test processes

ISO/IEC/IEEE 29119-3 (2021) ISO/IEC/IEEE 29119-3 Software and systems Engineering-Software testing-Part 3 Test documentation

ISTQB® Documents

ISTQB® Certified Tester Agile Test Leadership at Scale Syllabus v2.0 (2023)

ISTQB® Certified Tester Foundation Level Syllabus V.4 (2023)

ISTQB® Certified Tester Expert Level Test Management Syllabus v1.0 (2011)

ISTQB® Certified Tester Expert Level Improving the Test Process Syllabus v1.0.1 (2011)

Books

Basili, Trendowicz, Kowalczyk, Heidrich, Seaman, Münch, Rombach (2014) Aligning Organizations Through Measurement – The GQM+ Strategies Approach, Springer International

Bath, Graham, van Veenendaal, Erik (2014) Improving the Test Process - chapter 6: Process for Improvement, Rocky Nook

Belbin, R. Meredith (2010) Management Teams: Why They Succeed or Fail, Routledge: London

Black, Rex (2009) Managing the Testing Process, 3rd Edition, John Wiley & Sons

Boehm, B.W. (1979) Software Engineering Economics, Prentice-Hall.

Bonebright, Denise A. (2010) 40 years of storming: a historical review of Tuckman's model of small group development. Human Resource Development International. 1, 2010, Vol. 13.

Craig, Rick, Jaskiel, Stefan P. (2002) Systematic Software Testing, Artech House

Derby, E., Larsen, D. (2006) Agile Retrospectives – Making Good Teams Great, The Pragmatic Bookshelf

Erpenbeck, John und von Rosenstiel, Lutz (2017) Handbuch Kompetenzmessung. Stuttgart: Schäffer-Poeschel

Fowler, M. (2010) Hybrid development processes. IEEE Software, 27(2), 57-63.

Herzberg, Frederick, Mausner, Bernard und Bloch Snyderman, Barbara (1993) Motivation to Work, Routledge: London

Kahler, Taibi (2008) The Process Therapy Model: The Six Personality Types with Adaptations, Taibi Kahler Associates, Inc.

Marston, William Moulton (1999) Emotions Of Normal People, Routledge: London

Sonntag, Karlheinz und Schmidt-Rathjens, Claudia (2005) Anforderungsanalyse und Kompetenzmodelle. Wiesbaden, VS Verlag für Sozialwissenschaften

Tuckman, Bruce W (1965) Developmental sequence in small groups, Psychological Bulletin

van Ewijk, Alexander (2013) TPI NEXT – Business Driven Test Process Improvement, Sogeti Nederland B.V.

van Solingen, Berghout (1999) The Goal Question Metric Method – A Practical Guide for Quality Improvement of Software Development, McGraw-Hill

van Veenendaal, Erik (2012) The PRISMA Approach: Practical Risk-Based Testing, UTN Publishers

van Veenendaal, Erik (ed.) (2020) TMMi in the Agile world, version 1.4, TMMi Foundation

van Veenendaal, Erik, Cannegieter, Jan Jaap (2011) The Little TMMi – Objective-Driven Test Process Improvement, UTN Publishers

Articles

Feigenbaum, Armand V. (Nov/Dec 1956) Harvard Business Review, Vol. 34 Issue 6, p93-101

Hadjicostas, Evsevios (2004) Total Quality Management and Cost of Quality, Springer https://link.springer.com/chapter/10.1007/978-3-662-09621-5_7

Websites and Web Pages

www.tmmi.org Test Maturity Model integration (TMMi®); last visit January 31st, 2024

www.tmap.net Test Process Improvement (TPI); last visit January 31st, 2024

www.wikipedia.org/wiki/PDCA Plan-Do-Check-Act; last visit January 31st, 2024

5 Appendix A – Learning Objectives/Cognitive Level of Knowledge

The specific learning objectives that apply to this syllabus are shown at the beginning of each chapter. Each topic in the syllabus will be examined according to its learning objective.

The learning objectives begin with an action verb corresponding to its cognitive level of knowledge as listed below.

Level 1: Remember (K1)

The candidate will remember, recognize, and recall a term or concept.

Action verbs: Recall, recognize.

Examples
Recall the concepts of the test pyramid.
Recognize the typical objectives of testing.

Level 2: Understand (K2)

The candidate can select the reasons or explanations for statements related to the topic, and can summarize, compare, classify, and give examples for the testing concept.

Action verbs: Classify, compare, differentiate, distinguish, explain, give examples, interpret, summarize

Examples	**Notes**
Classify test tools according to their purpose and the test activities they support.	
Compare the different test levels.	Can be used to look for similarities, differences, or both.
Differentiate testing from debugging.	Looks for differences between concepts.
Distinguish between project and product risks.	Allows two (or more) concepts to be separately classified.
Explain the impact of context on the test process.	
Give examples of why testing is necessary.	
Infer the root cause of defects from a given profile of failures.	
Summarize the activities of the work product review process.	

Level 3: Apply (K3)

The candidate can carry out a procedure when confronted with a familiar task or select the correct procedure and apply it to a given context.

Action verbs: Apply, implement, prepare, use

Examples	Notes
Apply boundary value analysis to derive test cases from given requirements.	Should refer to a procedure/technique/process etc.
Implement metrics collection methods to support technical and management requirements.	
Prepare installability tests for mobile apps.	
Use traceability to monitor test progress for completeness and consistency with the test objectives, test strategy, and test plan.	Could be used in a LO that wants the candidate to be able to use a technique or procedure. Similar to 'apply'.

Level 4: Analyze (K4)

The candidate can separate information related to a procedure or technique into its constituent parts for better understanding and can distinguish between facts and inferences. A typical application is to analyze a document, software or project situation and propose appropriate actions to solve a problem or complete a task.

Action verbs: Analyze, deconstruct, outline, prioritize, select

Examples	Notes
Analyze a given project situation to determine which black-box or experience-based test techniques should be applied to achieve specific goals.	Examinable only in combination with a measurable goal of the analysis. Should be of form 'Analyze xxxx to xxxx' (or similar).
Prioritize test cases in a given test suite for execution based on the related product risks.	
Select the appropriate test levels and test types to verify a given set of requirements.	Needed where the selection requires analysis.

Reference

(For the cognitive levels of learning objectives)

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon

The specific learning objectives that apply to this syllabus are shown at the beginning of each chapter.

6 Appendix B – Business Outcomes Traceability Matrix with Learning Objectives

This section lists the traceability between the Business Outcomes and the Learning Objectives of The Advanced Level Test Management Syllabus.

Business Outcomes: Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
TM_01	Manage testing in various software development project by applying test management processes established for the project team or test organization		12										
TM_02	Identify test stakeholders and software development lifecycle models that are relevant in a given context			4									
TM_03	Organize risk identification and risk assessment sessions within any software development lifecycle and use these results to guide testing to reach the test objectives				6								
TM_04	Define a project test strategy consistent with the organizational test strategy and project context					11							
TM_05	Continuously monitor and control testing to achieve project objectives						4						
TM_06	Assess and report test progress to project stakeholders							3					
TM_07	Identify necessary skills and develop those skills within your team								6				
TM_08	Prepare and present a business case for testing in different contexts that outlines the costs and expected benefits									5			

Business Outcomes: Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
TM_09	Lead test process improvement activities in projects or software development product streams and contribute to organizational test process improvement initiatives										5		
TM_10	Planning the test activities and estimating the test effort											9	
TM_11	Create defect reports and a defect workflow suitable for a software development lifecycle												6
Unique LO	Learning Objective	K-Level											
1	Managing the Test Activities												
1.1	The Test Process												
TM-1.1.1	Summarize test planning	K2	X			X							
TM-1.1.2	Summarize test monitoring and test control	K2	X				X						
TM-1.1.3	Summarize test completion	K2	X					X					
1.2	The Context of Testing												
TM-1.2.1	Compare why different stakeholders are interested in testing	K2		X		X							
TM-1.2.2	Explain why stakeholders' knowledge is important in test management	K2		X		X							
TM-1.2.3	Explain testing in a hybrid software development model	K2		X		X							

Business Outcomes: Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
TM-1.2.4	Summarize test management activities for various software development lifecycles	K2	X	X		X							
TM-1.2.5	Compare test management activities at various test levels	K2	X			X							
TM-1.2.6	Compare test management activities for various test types	K2	X			X							
TM-1.2.7	Analyze a given project and determine test management activities, emphasizing test planning, test monitoring, and test control	K4	X			X							
1.3	Risk-based Testing												
TM-1.3.1	Explain the various measures that risk-based testing takes to respond to risks	K2			X								
TM-1.3.2	Give examples of different techniques a test manager can use for identifying risks related to product quality	K2			X								
TM-1.3.3	Summarize the factors that determine the risk levels related to product quality	K2			X								
TM-1.3.4	Select appropriate test activities to mitigate risks according to their risk level in a given context	K4			X								
TM-1.3.5	Differentiate between heavyweight and lightweight examples of risk-based testing techniques	K2			X								

Business Outcomes: Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
TM-1.3.6	Give examples of success metrics and difficulties associated with risk-based testing	K2			X								
1.4	The Project Test Strategy												
TM-1.4.1	Explain typical choices for a test approach	K2				X							
TM-1.4.2	Analyze an organizational test strategy and the project context to select the appropriate test approach	K4				X							
TM-1.4.3	Use the S.M.A.R.T. goal methodology to define measurable test objectives and exit criteria	K3				X							
1.5	Improving the Test Process												
TM-1.5.1	Explain how to use the IDEAL model for test process improvement on a given project	K2									X		
TM-1.5.2	Summarize the model-based improvement approach to test process improvement and understand how to apply it on a project context	K2									X		
TM-1.5.3	Summarize the analytical-based improvement approach to test process improvement and understand how to apply it on a project context	K2									X		
TM-1.5.4	Implement a project or iteration retrospective to evaluate test processes and discover testing areas to improve	K3									X		

Business Outcomes: Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
1.6	Test Tools												
TM-1.6.1	Summarize the best practices for tool introduction	K2										X	
TM-1.6.2	Explain the impact of different technical and business aspects when deciding on a tool type	K2										X	
TM-1.6.3	Analyze a given situation to create a plan for tool selection, covering risks, costs, and benefits	K4										X	
TM-1.6.4	Differentiate among the stages of the tool lifecycle	K2										X	
TM-1.6.5	Give examples for metric collection and evaluation by using tools	K2									X	X	
2	Managing the Product												
2.1	Test Metrics												
TM-2.1.1	Give examples of metrics to achieve the test objectives	K2					X						
TM-2.1.2	Explain how to control test progress using test metrics	K2					X						
TM-2.1.3	Analyze test results to create test reports that empower stakeholders to make decisions	K4					X	X					
2.2	Test Estimation												
TM-2.2.1	Explain the factors that need to be considered in test estimation	K2	X							X		X	

Business Outcomes: Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
TM-2.2.2	Give examples of factors which may influence test estimates	K2	X							X		X	
TM-2.2.3	Select an appropriate technique or approach for test estimation for a given context	K4	X							X		X	
2.3	Defect Management												
TM-2.3.1	Implement a defect management process, including the defect workflow, that can be used to monitor and control defects	K3											X
TM-2.3.2	Explain the process and participants required for effective defect management	K2											X
TM-2.3.3	Explain the specifics of defect management in Agile software development	K2	X										X
TM-2.3.4	Explain the challenges of defect management in hybrid software development	K2	X										X
TM-2.3.5	Use the data and classification information that should be gathered during defect management	K3											X
TM-2.3.6	Explain how defect report statistics can be used to devise process improvement	K2										X	X
3	Managing the Team												
3.1	The Test Team												
TM-3.1.1	Give examples of typical skills needed by test team members within four areas of competence	K2							X				

Business Outcomes: Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
TM-3.1.2	Analyze a given project context to determine required skills for test team members	K4							X				
TM-3.1.3	Explain typical techniques for skill assessments for test team members	K2							X				
TM-3.1.4	Differentiate between the typical approaches for developing skills of test team members	K2							X				
TM-3.1.5	Explain skills required to manage a test team	K2							X				
TM-3.1.6	Give examples of motivating and hygiene factors for test team members	K2							X				
3.2	Stakeholder Relationships												
TM-3.2.1	Give examples of each of the four categories determining the cost of quality	K2								X			
TM-3.2.2	Apply a cost-benefit calculation to estimate the added value of testing for stakeholders	K3						X		X			

7 Appendix C – Release Notes

The ISTQB® Advanced Level Test Management Syllabus v3.0 is a major update based on the Advanced Level Syllabus Test Manager 2012. For this reason, there are no detailed release notes per chapter and section. However, a summary of principal changes is provided below.

In this version all Learning Objectives (LOs) have been edited to make them atomic, and to create one-to-one traceability between LOs and syllabus sections, thus not having content without also having a LO. The goal is to make this version easier to read, understand, learn, and translate, focusing on increasing practical usefulness and the balance between knowledge and skills.

This major release has made the following changes:

- Size reduction of the overall syllabus. A syllabus is not a textbook, but a document that serves to outline the basic elements of an advanced course on software testing, including what topics should be covered and at what level. Therefore, in particular:
  - In most cases examples are excluded from the text. It is a task of a training provider to provide the examples, as well as the exercises, during the training
  - The “Syllabus writing checklist” was followed, which suggests the maximum text size for LOs at each K-level (K2 = maximum of 1,500 nonblank characters, K3 = maximum of 2,500 nonblank characters, K4 = maximum of 3,000 nonblank characters, +/- 20%)
- Reduction of the number of LOs compared to the CTAL TM 2012 Syllabus
  - 36 K2 LOs compared with 39 LOs in CTAL TM 2012
  - 5 K3 LOs compared with 12 LOs in CTAL TM 2012
  - 7 K4 LOs compared with 10 LOs in CTAL TM 2012
- Complete structure of the syllabus is revised
- Alignment with the ISTQB® Foundation Level Syllabus V.4 is complete
- Major changes in 2012 former Chapter 1 (Test Process)
  - Restricted to Managing the Test Activities (Test Planning, Test Monitoring, Test Control, and Test Completion)
  - Integrated as a section in the new chapter “Managing the Test Activities”
- New chapter **Managing the Test Activities**
  - Section 1.1 – The Test Process: see above
  - Section 1.2 – The Context of Testing: Expanded to cover non-sequential software development models
  - Section 1.3 – Risk-Based Testing: Completely rewritten to make it more applicable on a project level
  - Section 1.4 – The Project Test Strategy: Because the test plan is already defined in the ISTQB® Foundation Level Syllabus V.4, the focus is on selecting the adequate test approach and how to define measurable test objectives

    - Section 1.5 – Improving the Test Process: Integrating this into Managing the Test Activities, showing how to apply it in a project context, and implementing it using retrospectives within an iteration or project
    - Section 1.6 – Test Tools: Introducing tools was moved from the ISTQB® Foundation Level Syllabus V.3.1 (is not present in ISTQB® Foundation Level Syllabus V.4)
- New chapter **Managing the Product**
    - Section 2.1 – Test Metrics: Former sections defining metrics and use of metrics, Test Metrics
    - Section 2.2 – Test Estimation: The ISTQB® Foundation Level Syllabus V.4 already covers the calculation of test estimation. Expanded to select at a K4 level adequate test estimation techniques and the use of test estimates across the SDLC models
    - Section 2.3 – Defect Management: aligned with the latest editions of standards and expanded to use in Agile and hybrid software development
- New chapter **Managing the Team**
    - Section 3.1 – The Test Team: The main topics are the same as in the TM 2012 Syllabus. Identify the individual skills and compose the test teams
    - Section 3.2 – Stakeholder Relationships: It is the former ATM 2012 Syllabus section entitled “Business Value of Testing”
- Major changes and deleted sections/chapters in CTAL TM Syllabus 2012
    - Section on Distributed, Outsources and Insources Testing removed
    - Section on Managing the Application of Industry Standards removed
    - Chapter on Reviews removed
    - Subsections on Improving the Test Process CTP and STEP removed
    - Subsections on Test Analysis, Test Design, Test Implementation and Test Execution removed

8 Appendix D – Domain-Specific Keywords

Term Name	Definition
goal question metric (GQM)	An approach to software measurement using a three-level model consisting of a conceptual level (goal), an operational level (question) and a quantitative level (metric).
IDEAL	An organizational improvement model that serves as a roadmap for initiating, planning, and implementing improvement actions.
indicator	A measure that provides an estimate or evaluation of specified attributes derived from a model with respect to defined information needs.
measure	The number or category assigned to an attribute of an entity by making a measurement.
metric	A measurement scale and the method used for measurement.
planning poker	A consensus-based estimation technique, mostly used to estimate effort or relative size of user stories in Agile software development. It is a variation of the Wideband Delphi method using a deck of cards with values representing the units in which the team estimates.
Three-point estimation	An expert-based technique, three estimations are made by the experts: the most optimistic estimation (a), the most likely estimation (m) and the most pessimistic estimation (b). The final estimate (E) is their weighted arithmetic mean.
Wideband Delphi	An expert-based test estimation technique that aims at making an accurate estimation using the collective wisdom of the team members.

9 Appendix E – Trademarks

CMMI® is a registered trademark in the U.S. Patent and Trademark Office by Carnegie Mellon University.

ISTQB® is a registered trademark of International Software Testing Qualifications Board.

TMMi® is a registered trademark of TMMi Foundation.

TPI-Next® is a registered trademark of Sogeti, The Netherlands.

10 Index

All terms are defined in the ISTQB® Glossary (http://glossary.istqb.org/).