Certified Tester

Expert Level Test Management (CTEL-TM) v2.0

International Software Testing Qualifications Board



© International Software Testing Qualifications Board

Copyright Notice

Copyright Notice © International Software Testing Qualifications Board (hereinafter called ISTQB®)

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

Copyright © 2026, new reworked Version 2.0 released on November 10, 2026.

The authors Ina Schieferdecker, Arnika Hryszko, Adam Roman, Armin Born, Anja Kribernegg, Erhardt Wunderlich, Jaroslaw Hryszko, Renzo Cerquozzi.

Working Group team members: Benjamin Timmermans, Anoop Rayudu, Erik van Veenendaal, Reifa Tangon, Samuel Ouko, Webstar Ongaki Okerio.

hereby transfer the copyright to the ISTQB®.

Copyright © 2022, the author for the errata updates 2022 Gary Mogyorodi (Glossary Working Group).

Copyright © 2011, the authors Graham Bath (ISTQB® Chair, Expert Level), Rex Black (Chair for the Expert Level Test Management syllabus), Debra Friedenberg, Kenji Onishi, Marcel Kwakernaak, Martin Klonk, Judy McKay (Vice-Chair for the Expert Level Test Management syllabus), Gary Mogyorodi, Erik van Veenendaal (ISTQB® Vice-Chair, Expert Level)

All rights reserved. The authors hereby transfer the copyright to the ISTQB®. The authors (as current copyright holders) and ISTQB® (as the future copyright holder) have agreed to the following conditions of use:

Extracts, for non-commercial use, from this document may be copied if the source is acknowledged. Any Accredited Training Provider may use this syllabus as the basis for a training course if the authors and the ISTQB® are acknowledged as the source and copyright owners of the syllabus and provided that any advertisement of such a training course may mention the syllabus only after official Accreditation of the training materials has been received from an ISTQB®-recognized Member Board.

Any individual or group of individuals may use this syllabus as the basis for articles and books, if the authors and the ISTQB® are acknowledged as the source and copyright owners of the syllabus.

Any other use of this syllabus is prohibited without first obtaining the approval in writing of the ISTQB®.

Any ISTQB®-recognized Member Board may translate this syllabus provided they reproduce the abovementioned Copyright Notice in the translated version of the syllabus.

Revision History

Version	Date	Remarks
V2.0	2026/09/29	Ready for Release
v2.0	2025/10/30	Complete rework of Syllabus The main changes for the CTEL-TM Syllabus V2.0 • BOs, LOs reengineering and adaptation for the new Expert TM “understanding” • Shorter Syllabus (duplicated text contained in other Syllabi removed) • New chapters’ structure • One certificate only (instead of the current three) • Training duration will be 5 days • Exam question will be MCQ • Certificate validity period duration removed (the EL-TM 2.0 certificate does not need renewal)
1.0.1	2022/mm/dd	Errata update of Syllabus. Document format updated to the Syllabus template version 3.0.1, and the terms in the Syllabus updated to agree with Glossary version 3.6.1
2011 Release	2011/11/1	Released version 1.0
2011 Beta	2011/4/14	Beta version

© International Software Testing Qualifications Board

Table of Contents

Copyright Notice 2
Revision History 3
Table of Contents 4
Acknowledgements 8
0 Introduction 9
0.1 Purpose of this Syllabus 9
0.2 The Test Management Expert in Software Testing 9
0.3 Career Path for Testers 10
0.4 Business Outcomes 10
0.5 Learning Objectives and Cognitive Level of Knowledge 11
0.6 The CTEL-TM Certificate Exam 11
0.7 Accreditation 11
0.8 Handling of Standards 12
0.9 Level of Detail 12
0.10 How this Syllabus is Organized 13
1 Test Policies, Test Strategies and Test Objectives – 495 minutes 15
Introduction to Test Policies, Test Strategies, and Test Objectives 16
1.1 Organizational Quality Goals and Test Policy 16
1.1.1 Organizational Quality Goals and Test Policy Definition 16
1.1.2 Test Objectives and Long-term Quality Considerations 17
1.2 Organizational-level Test Strategies and Practices 18
1.2.1 Test Strategy Selection and Analysis 18
1.2.2 Test Strategy Failure Analysis 19
1.2.3 Creating Test Strategy Documents 20
1.3 Aligning the Test Policy with the Test Strategy within the Organization 21
1.3.1 Aligning the Test Policy with the Test Strategy 21
2 Managing the Test Team – 600 minutes 24
Introduction to Managing the Test Team 25
2.1 Building the Test Team 25
2.1.1 Resume Evaluation, Interview Design, and Candidate Selection 25
2.1.2 New Employee Training Program 27

© International Software Testing Qualifications Board

2.1.3 Employment and Contractor Termination Considerations 28
2.2 Developing the Test Team 28
2.2.1 Test Team Development Planning and S.M.A.R.T. Goal Methodology (SMART) 28
2.2.2 Role Clarity and Test Team Responsibilities 29
2.2.3 Skills Development and Learning Options 30
2.2.4 Performance Review Management 31
2.3 Leading the Test Team 32
2.3.1 Adaptive Leadership and Communication Strategy 32
2.3.2 Test Team Building and Distributed Test Team Management 33
2.2.3 Test Team Motivation Challenges 34
3 Managing Across the Organization – 510 minutes 36
Introduction to Managing Across the Organization 37
3.1 Advocating for the Test Team 37
3.1.1 Test Team Advocacy Strategies and Systemic Leadership 37
3.1.2 Value Communication and Stakeholder Engagement 39
3.2 Placement of the Test Team 43
3.2.1 Organizational Placement Selection 43
3.2.2 Placement Impact Analysis and Assessment 46
3.3 Building Strategic Relationships 47
3.4 Stakeholder Communication 49
3.5 Integrating Tools Across the Organization 50
3.5.1 Factors to Consider When Dealing with Multi-Use Tools 50
3.5.2 Impact Analysis of Proposed Changes to Multi-Use Tools 51
4 Managing External Relationships – 330 minutes 54
Introduction to Managing External Relationships 55
4.1 Types of External Relationships 55
4.2 Contractual Issues 56
4.3 Communication Strategies for Third Parties 56
4.4 Integrating from External Sources 58
4.5 Merging Third-Party Test Strategies 59
4.6 Verifying Third-Party Quality 60
5 Project Management Essentials – 420 minutes 62
Introduction to Project Management Essentials 63
5.1 Test Estimation and Defect Prediction 63

© International Software Testing Qualifications Board

5.1.1 Uncertainty in Test Estimation 63
5.1.2 Defect Prediction Models and Techniques 64
5.2 Project Risk Management 66
5.2.1 Trade-off Decision Making 66
5.2.2 Project Risk Analysis 67
5.3 Change Management 68
5.3.1 TME Role in Change Management Process 68
5.3.2 Impact of Project-wide Changes on Testing 69
5.4 Test Management Considerations for Project Types 71
5.4.1 Managing in Various Project Types 71
6 Test Monitoring and Test Control – 495 minutes 72
Introduction to Test Monitoring and Test Control 73
6.1 Metrics and Measurement 73
6.1.1 Selecting Test Metrics 73
6.1.2 Evaluating the Test Measurement and Test Reporting Process 74
6.2 Test Reporting 76
6.2.1 Internal and External Test Reporting 76
6.2.2 Data Visualization in Test Progress Dashboard 77
6.2.3 Preparing Test Progress Reports 78
6.3 Interpreting the Test Results 79
6.3.1 Quality Control Tools 79
6.3.2 Data-Based Test Process Analysis 80
6.3.3 Performing Test Control 81
7 Evaluating Effectiveness, Efficiency and Satisfaction and Improving Test Policy – 315 minutes 84
Introduction to Evaluating Effectiveness, Efficiency and Satisfaction and Improving Test Policy 85
7.1 Effectiveness, Efficiency and Satisfaction Metrics for the Test Process 85
7.1.1 Tracking the Test Process Metrics 85
7.1.2 Using Metrics to Identify Opportunities for Test Process Improvement 86
7.2 Effectiveness, Efficiency and Satisfaction Metrics for the Test Policy 88
7.3 Test Policy Improvement 92
7.3.1 Coordinating a Managerial Review for Test Policy Assessment 92
7.3.2 Formulating Improvement Plans 93
8 Ethical Considerations - 105 minutes 95

© International Software Testing Qualifications Board

Introduction to Ethical Considerations 96
8.1 Ethical Principles in Test Management 96
8.2 Identifying and Addressing Ethical Issues 97
9 List of Abbreviations 99
10 References 101
10.1 Standards 101
10.2 ISTQB® Documentation 101
10.3 Glossary References 102
10.4 Books, Articles and Web Pages 102
10.4.1 Books, Articles and Web Pages (not directly referenced in this Syllabus) 103
11 Trademarks 105
12 Appendix A – Learning Objectives/Cognitive Level of Knowledge 106
Level 1: Remember (K1) 106
Level 2: Understand (K2) 106
Level 3: Apply (K3) 107
Level 4: Analyze (K4) 107
Level 5: Evaluate (K5) 108
13 Appendix B – Business Outcomes traceability matrix with Learning Objectives 109
14 Appendix C – Release Notes 118
15 Appendix D – Information for training providers 119
15.1 Training Time 119
15.2 Guideline for Practical Exercises in the Workplace 120
16 Index 121

© International Software Testing Qualifications Board

Acknowledgements

This document was formally released by the General Assembly of the ISTQB® on 2026/10/30.

It was produced by a team from the International Software Testing Qualifications Board:

Ina Schieferdecker, Arnika Hryszko, Adam Roman, Armin Born, Anja Kribernegg, Erhardt Wunderlich, Jaroslaw Hryszko, Benjamin Timmermans, Anoop Rayudu, Erik van Veenendaal, Reifa Tangon, Samuel Ouko, Webstar Ongaki Okerio, Renzo Cerquozzi.

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

The team thanks Gary Mogyorodi for performing the technical edit.

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

(in alphabetical order): Arda Ender Torçuk, Bíró Ádám, Bob van de Burgt, Dandan Zheng , Daniel van der Zwan, Eduard Hartog, Frank Neirynck , Gary Mogyorodi, Imre Meszaros, Ine Lutterman, József Kreisz, Judy McKay, Klaus Skafte, Laura Albert, Liang Ren, Lucjan Stapp, Maria Omoise, Martijn de Ruijter, Meile Posthuma, Menno van den Berg, Michaël Pilaeten, M. Marton, Márton Siska, Matthias Hamburg, Micheel Stahl, Myriam Christener, Nishan Portoyan, Ole Chr. Hansen, Péter Sótér, Petr Neugebauer, Ray Gillespie, Stephanie Ulrich, Tal Pe'er, Thomas Letzkus, Walter Eder, Zsolt Hargitai.

The v1.0 (2011) of this document was produced by a core team of authors from the International Software Testing Qualifications Board Expert Level Working Group for the module – Test Management: Graham Bath (ISTQB® Chair, Expert Level), Rex Black (Chair for the Expert Level Test Management syllabus), Debra Friedenberg, Kenji Onishi, Marcel Kwakernaak, Martin Klonk, Judy McKay (Vice-Chair for the Expert Level Test Management syllabus), Gary Mogyorodi, Erik van Veenendaal (ISTQB® Vice-Chair, Expert Level).

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

At the time the Expert Level Syllabus for this module was completed, the Expert Level Working Group (Test Management) had the following membership (alphabetical order): Graham Bath, Rex Black, Monika Boegge, Debra Friedenberg, Simon Frankish, Leanne Howard, Kari Kakkonen, Marcel Kwakernaak, Martin Klonk, Vipul Kocher, Steen Lerche-Jensen, Judy McKay, Gary Mogyorodi, Joanna Nowakowska, Thomas Rossner, Andrea Rott, Dakar Shalom, Erik van Veenendaal

The following persons participated in the reviewing, commenting and balloting of the v1.0 version of this syllabus (alphabetical order): Graham Bath, Gualtiero Bazzana, Rex Black, Monika Boegge, Piet de Roo, Debra Friedenberg, Bernard Homés, Leanne Howard, Kari Kakkonen, Beata Karpinska, Marcel Kwakernaak, Martin Klonk, Alon Linetzki, Judy McKay, Gary Mogyorodi, Ingvar Nordstrom, Thomas Rossner, Andrea Rott, Andy Soundranagam, Jurian van de Laar, Erik van Veenendaal.

The v1.0 version of this document was formally released by the General Assembly of ISTQB® on November 1, 2011.

© International Software Testing Qualifications Board

0 Introduction

0.1 Purpose of this Syllabus

This syllabus forms the basis for the International Software Testing Qualification for the Expert Level 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 training providers, to produce training materials and determine appropriate teaching methods.
4. To certification candidates, to prepare for the certification exam (either as part of a training course or independently).
5. To the international software and systems engineering community, to advance software and systems testing profession, and as a basis for books and articles.
6. To the international software and systems engineering research community, with the aim of comprehending the state of the art to advance the body of knowledge on software testing through follow-up research.

0.2 The Test Management Expert in Software Testing

The Expert Level Test Management qualification (CTEL-TM) provides the skills needed to perform test management at the expert level for organizations and test projects. It details the **Test Management Expert** (TME) role and responsibilities in testing at the strategic, organizational, and technical level.

The TME is an individual with a broad knowledge of testing in general, and an in-depth experience of test management. An in-depth experience is defined as having a proven record of accomplishment of outstanding knowledge of testing theory and practice to be able to support, define, and actively influence the direction that an organization and/or project takes when creating, implementing, and executing test processes. A person in the TME role may hold a title such as test director, director of testing/quality, and senior test manager.

In this syllabus, the TME is understood as a role which:

- is responsible for the test management of an organization and for test projects that are typically complex and large in scale and business impact
- supervises and coaches test managers and testers
- supports test activities by providing guidance to test managers and stakeholders in general, instructing them and facilitating any necessary adjustments
- reports directly to the executive and board level of an organization

The Test Management Expert Level certification is aimed at people who have already achieved an advanced point in their careers in software testing, hold the ISTQB® Advanced Level Test Management (CTAL-TM) certificate, and wish to further develop their expertise in test management.

© International Software Testing Qualifications Board

0.3 Career Path for Testers

The ISTQB® scheme supports testers at all stages of their careers. Individuals who achieve the ISTQB® Expert Level Test Management certification may also be interested in the Expert Level Improving the Test Process certification (ISTQB-ITP). Anyone seeking to develop skills in test practices in an agile environment area could consider the Agile Technical Tester (ISTQB-ATT) or Agile Test Leadership at Scale (ISTQB-ATLaS) certifications. In addition, specialist streams offer certification products focusing on specific test technologies and test strategies, specific quality characteristics and test levels, or testing within particular industry domains. Please visit www.istqb.org for the latest information on ISTQB®´s Certified Tester Scheme.

0.4 Business Outcomes

This section lists the Business Outcomes expected of a candidate who has achieved the Expert Level Test management certification.
A CTEL-TM Certified Tester can…

ELTM-BO1	On organizational/strategic level, promote and position testing to produce quality software.
ELTM-BO2	Define, implement, evaluate, and improve the test policy and test strategy suitable for the business objectives and support commitments within the organization.
ELTM-BO3	Align test management with other roles or management areas across the organization and outside the organization.
ELTM-BO4	Lead the test management within an organization, project or program to identify and manage critical success factors and risks with management commitment at strategic level.
ELTM-BO5	Understand and manage the human themes associated with test project management and implement necessary changes.
ELTM-BO6	Set up a metrics tracking and test reporting for test monitoring and test control, interpreting the test results, making business decisions, and communicating to internal and external stakeholders.
ELTM-BO7	Assess the current test maturity, implement improvements, and manage the introduction of change.
ELTM-BO8	Adapt test management to address different software development lifecycles (SDLCs) models or project types.
ELTM-BO9	Maximize productivity by selecting the appropriate mix of personnel considering skill levels, staffing options, location of resources and management structure.
ELTM-BO10	Introduce innovations to test management within organization, which among others include roles, skills, test techniques, technologies, test tools and organizational structure.

© International Software Testing Qualifications Board

0.5 Learning Objectives and Cognitive Level of Knowledge

In general, all the content of this syllabus is examinable, except for the Introduction and Appendices. The exam questions will confirm knowledge of learning objectives at the respective level of knowledge.

The specific learning objectives and their levels of knowledge are shown at the beginning of each chapter, and classified as follows:

- K1: Remember
- K2: Understand
- K3: Apply
- K4: Analyze
- K5: Evaluate

Further details and examples of learning objectives are given in Appendix A – Learning Objectives/Cognitive Level of Knowledge.

For all terms listed as keywords just below chapter headings, the correct name and definition from the ISTQB® Glossary shall be understood (K2), even if not explicitly mentioned in the learning objective.

0.6 The CTEL-TM Certificate Exam

The CTEL-TM Certificate exam will be based on this syllabus. Answers to exam questions may require the use of material based on more than one section of this syllabus. Standards and books are included as references, but their content is not examinable, beyond what is summarized in the syllabus itself from such standards and books.

Refer to Exam Structures and Rules document for the Expert Level for further details.

The entry criterion for taking the Expert Level Test Management exam is that the candidate

- has the ISTQB® CTFL certificate
- has the ISTQB® CTAL-TM certificate

People that formally comply with the criteria defined above and pass the exam will receive the formal ISTQB® Test Management Expert Level certificate. Those that possess an ISTQB® Expert Level certificate are allowed to use the Certified Tester Expert Level – Test Management (CTEL-TM) abbreviation.

0.7 Accreditation

An ISTQB® Member Board may accredit training providers whose course material follows this syllabus. Training providers should obtain 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 follow the general Accreditation Guideline published by the Processes Management and Compliance Working Group.

© International Software Testing Qualifications Board

0.8 Handling of Standards

International standardization organizations like IEEE and ISO have issued standards associated with quality characteristics and software testing. Such standards are referenced in this syllabus. The purpose of these references is to provide a framework (as in reference to ISO/IEC 25010 regarding quality characteristics) or to provide a source of additional information if desired by the reader. Please note that ISTQB® syllabi uses the standard documentation as reference. Standard documentation is not intended for examination. Refer to Section 10.1 for more information on standards.

0.9 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 CTEL-TM Level
- A list of terms that students must be able to understand
- 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 does not describe the entire knowledge area of software testing. It reflects the level of detail to be covered in Expert Level Test Management training courses.

The syllabus uses the terminology (e.g., the name and meaning) of the terms used in software testing and quality assurance (QA) according to the ISTQB® Glossary (see Section 10.3).

© International Software Testing Qualifications Board

0.10 How this Syllabus is Organized

There are eight (8) chapters with examinable content. The top-level heading for each chapter specifies the time for the chapter; timing is not provided below the chapter level. For accredited training courses, the syllabus requires a minimum of 34 hours and 5 minutes of instruction extended with 20 hours and 40 minutes of exercises in the workplace and/or at home. It is discretionary for the trainer to check the quality of the homework by the students. For a detailed information about the timing, refer to chapter Training Time (see Section 15.1). The hours of instructions are distributed across the eight chapters as follows:

- Chapter 1: 495 minutes Test Policies, Test Strategies and Test Objectives
  - The test manager analyzes organizational quality goals to define a consistent test policy, sets and evaluates long-term test objectives and priorities, and establishes metrics for process improvement.
  - The test manager distinguishes test strategies for different project contexts, identifies causes of test strategy failures, and creates or recommends comprehensive test strategy documentation that supports success and mitigate risks.
  - The test manager evaluates and enhances test policies and test strategies to confirm alignment with organizational goals across people, processes, test tools, systems, and test techniques, considering both short- and long-term needs.
- Chapter 2: 600 minutes Managing the Test Team
  - The test manager evaluates candidates for strategic test roles, designs training and development programs for testers, and considers key factors when managing employee or contractor hiring or terminating.
  - The test manager adopts leadership and communication approaches, confirms role clarity, identifies skill development opportunities through training and mentoring, and conducts performance reviews.
  - Trainees design collaboration activities that improve collaboration practices across projects and address short- and long-term motivational challenges within test teams.
- Chapter 3: 525 minutes Managing Across the Organization
  - The test manager applies test strategies to promote and defend testing, while identifying and effectively communicating both quantitative and qualitative benefits to stakeholders.
  - The test manager analyzes the implications of different tester placement options.
  - The test manager analyzes complex project environments to identify key relationships and develops test strategies for building effective partnerships with managers and other stakeholders.
  - The test manager practices effective communication with testing stakeholders, including non-testing staff, about critical test issues.
  - The test manager explains factors to consider multi-use tools and analyzes the organizational impact of proposed changes to such tools.

© International Software Testing Qualifications Board

- Chapter 4: 330 minutes Managing External Relationships
    - The test manager gives examples of third-party relationships in contracts, focusing on testing responsibilities.
    - The test manager prepares suitable service level agreements (SLAs) for projects involving external vendors.
    - The test manager discusses effective communication strategies for third-party engagements.
    - The test manager analyzes project scenarios to determine the appropriate level of involvement when third parties contribute to development or testing.
    - The test manager summarizes key considerations when merging internal and third-party test strategies.
    - The test manager analyzes the quality of external relationships in test contexts.
- Chapter 5: 420 minutes Project Management Essentials
    - The test manager applies statistical methods like standard deviation to interpret test estimations and use defect prediction models to anticipate test results.
    - The test manager analyzes trade-offs between quality, test schedule, budget, and test scope, and applies project risk analysis to manage uncertainties in test projects.
    - The test manager summarizes the role of the TME in change management and performs impact analysis of project-wide changes on test activities.
- Chapter 6: 495 minutes Test Monitoring and Test Control
    - The test manager identifies effective test metrics and evaluates test measurement and test reporting processes to provide meaningful test monitoring.
    - The test manager applies quality control (QC) techniques, analyzes test projects using available data, and implements control actions to address deviations from the test plan.
- Chapter 7: 315 minutes Evaluating Effectiveness, Efficiency and Satisfaction and Improving the Test Policy
    - The test manager explains the purpose of tracking and reviewing effectiveness, efficiency, and satisfaction measures in testing.
    - The test manager evaluates effectiveness, efficiency, and satisfaction measures in relation to test policy objectives.
    - The test manager uses managerial reviews to assess the quality of test policies and analyzes review results to develop plans for improving the test policy.
- Chapter 8: 105 minutes Ethical Considerations
    - The test manager explains fundamental ethical principles in test management and their importance for professional integrity and mutual stakeholders' trust.
    - The test manager analyzes behaviors in specific situations to identify potential ethical issues and determines appropriate corrective actions.

© International Software Testing Qualifications Board

1 Test Policies, Test Strategies and Test Objectives – 495 minutes

Keywords

analytical test strategy, consultative test strategy, methodical test strategy, model-based test strategy, process-compliant test strategy, reactive test strategy, regression-averse test strategy, test objective, test policy, test practice, test strategy

Domain Specific Keywords

organizational quality goal

Learning Objectives for Chapter 1:

1.1 Quality Goals and Test Policy

ELTM-1.1.1 (K4) Analyze the organizational quality goals to determine test policy consistency with these goals

ELTM-1.1.2 (K5) Evaluate test objectives that might have long term effects on software quality and on organizational, economic, and technical dependencies

1.2 Organizational-level Test Strategies

ELTM-1.2.1 (K4) Analyze project context to determine which individual test strategies, or blended test strategies, are most likely to succeed

ELTM-1.2.2 (K4) Analyze test strategy failures to identify causes and contributing factors

ELTM-1.2.3 (K5) Recommend a comprehensive test strategy that enables success factors and mitigates the risks of failures

1.3 Alignment of Test Policy and Test Strategy with the Organization

ELTM-1.3.1 (K5) Evaluate test policies and test strategies for organizational alignment, considering short-term and long-term perspectives across processes, people, tools, systems, and techniques

© International Software Testing Qualifications Board

Introduction to Test Policies, Test Strategies, and Test Objectives

Introduction

In today's complex software development landscape, expert level test management requires a clear understanding of how test policies guide testing, and how it is translated into test strategies and measurable goals. In the ISTQB® context, a test policy sets out the overarching principles and high-level test objectives in the organization. The test policy typically expresses the purpose and business value of testing. Test strategies operationalize the test policy, while test objectives define specific targets to measure test progress and success. This chapter explores these interrelated concepts and provides TMEs with the knowledge and tools to establish, maintain and improve effective, policy-driven test automation frameworks within their organizations. More information on organizational test strategy can be found in the ISO/IEC/IEEE 29119-3 standard.

1.1 Organizational Quality Goals and Test Policy

Introduction

Testing derives its value from alignment with business goals and organizational quality goals. The TME must establish clear connections between test objectives, test activities, and business value, defining a test policy that supports business success. This alignment requires an understanding of organizational quality goals, establishing measurable objectives, and determining metrics that demonstrate test contribution to overall success at the relevant organizational level (e.g., project, program, department, business unit or enterprise), and supports risk management.

1.1.1 Organizational Quality Goals and Test Policy Definition

Where defined, measurable organizational quality goals express customer, regulatory, and business needs. Examples include reliability targets, performance thresholds, security requirements, regulatory requirements, usability criteria, and certification needs. If these goals are not explicitly documented at the strategic level, the TME needs to draft and validate them with stakeholders (or derive proxies from applicable policies/standards) before translating them into the test policy. The test policy should directly support these organizational quality goals by stating the intended outcomes and value of testing; specific test activities are defined in the test strategy and test plans (e.g., master test plan or level test plan), not in the test policy.

The TME should identify all relevant stakeholders (including those that have approval authority for releases), then engage with them to understand quality priorities and translate those priorities into the test policy. This process involves among others identifying critical quality characteristics, understanding acceptable risk levels, and determining how testing can best contribute to the quality goals. The test policy clearly articulates the purpose, test scope, and value proposition of the test object in scope, whether this refers to project-level testing, program-level testing, or organization-level testing.

© International Software Testing Qualifications Board

Common test policy elements include defect detection and defect prevention, risk management through targeted testing, provision of quality information to decision-makers, and complying with quality standards. The test policy provides a focused statement of purpose and value specific to the test organization in scope and also defines overarching principles and rules. The test policy should reflect organizational context, including market requirements, regulatory requirements, and business constraints that influence test priorities.

Effective test policy definition requires balancing aspirational goals with practical considerations such as budget, timeline, and resource availability. From an expert level perspective, these factors should ideally result from the test policy rather than act as imposed constraints. They often require a negotiation loop between policy-driven needs and organizational constraints. The TME should confirm that the test policy is specific, measurable, achievable, relevant, and time-bound, providing clear direction for test activities while maintaining flexibility to adapt to changing organizational needs.

1.1.2 Test Objectives and Long-term Quality Considerations

Test objectives define specific, measurable goals that support the overall test policy and contribute to achieving long-term software quality. Test objectives are established during test planning, following an agreement on the test policy. When defining test objectives, the TME must consider organizational dependencies, budget limitations, and technical factors that may impact the effectiveness and maintainability of test activities.

In this context, organizational dependencies are factors within the organization that influence the ability to implement testing effectively. These may include team structures, skill availability, process maturity, and cultural norms. The TME should evaluate how organizational changes, such as Agile software development transformations or DevOps adoption, impact test objectives, and adapt the test policy and test strategy accordingly. Short-, mid-, and long-term organizational health (e.g., vitality, resilience, and effectiveness) requires sustainable test practices that support continuous improvement and knowledge of retention.

Economic dependencies encompass budget constraints, cost-benefit analysis of test investments, and return on investment (ROI) considerations. Test objectives should balance immediate quality needs with long-term economic sustainability, considering factors such as test automation investment, tool licensing costs, and training requirements. Economic evaluation should include total cost of ownership for test solutions and their impact on overall project economics and may also consider the total cost of quality (CoQ) across the entire SDLC, balancing the cost of delivering quality with the cost of internal failure and external failure, to reduce the loss of reputation.

Technical dependencies involve system architecture, technology stack limitations, tool landscape, and integration requirements that influence test strategies. The TME must consider technical debt, platform evolution, and emerging technologies when defining test objectives. Long-term technical sustainability requires test strategies that adapt to changing technology landscapes while maintaining quality standards.

Effective evaluation of test objectives requires criteria including strategic alignment, measurability, feasibility, and stakeholders' value. The TME should regularly assess the relevance of these test objectives, considering changing business needs, technology evolution, and lessons learned from previous test cycles.

Improvement opportunities emerge through metric analysis, trend identification, and benchmarking against relevant internal and external standards and surveys. The TME should establish baseline measurements, track performance over time, and identify areas requiring attention. Common improvement areas may include the process of standardization, skill development, test automation approach, and tool utilization.

© International Software Testing Qualifications Board

Successful metrics implementation requires clear definitions, consistent measurement methods, and regular review cycles, for example by applying the goal-question-metric (GQM) approach (Basili, 1984) to confirm alignment between objectives, questions, and the collected metrics. Metrics should provide clear, practical guidance for improvement of activities, rather than simply serving as passive indicators of performance. The metrics framework should evolve with organizational maturity and changing business needs and should be reviewed whenever changes in business strategy or test policy may affect which metrics are relevant.

1.2 Organizational-level Test Strategies and Practices

Organizational test practices define the overall principles and high-level test objectives within the organization, establishing the framework for how testing should be approached across all projects. Test strategies specify the particular test approach and test techniques to be used for testing in specific contexts or projects. While organizational practices provide overarching guidance and standards, test strategies are implemented within test plans to address project-specific requirements, constraints, and stakeholders' needs. The TME analyzes situational factors to select and blend appropriate test approaches within their test plans, providing alignment with organizational practices while adapting to non-functional requirements and project contexts for successful outcomes.

1.2.1 Test Strategy Selection and Analysis

Test strategy selection requires systematic analysis of project characteristics, organizational capabilities, and contextual factors. The TME must evaluate multiple test strategy options, considering their benefits, success factors, and inherent risks to determine optimal approaches at the organizational level, with adaptations for individual projects where necessary.

Analytical test strategies include requirements-based testing and risk-based testing, where the TME analyzes the test basis to identify coverage areas (i.e., test conditions). These test strategies excel when comprehensive documentation exists, and stakeholders' engagement is available. Conditions for success are early analysis, stable testware, and effective stakeholders' collaboration. Analytical test strategies may be unsuccessful when testware is poor; requirements change frequently, or stakeholders' input is unavailable.

Model-based test strategies leverage models for various software aspects such as operational profiles, system architecture, data and behavior, or usage patterns. These test strategies work well when appropriate models exist, and tool support is available. Risks are related to insufficient data and domain knowledge for model construction, abstraction of inaccuracies, or inappropriate tool selection.

Methodical test strategies use quality standards or domain-specific test techniques to provide structured frameworks for test activities. These test strategies offer systematic coverage of defined quality characteristics and enable consistent application of proven test techniques across different contexts. The key advantage lies in their reproducibility and standardization, making them particularly valuable when test conditions are well-defined, and acceptance criteria are established. However, methodical test strategies require careful adaptation when project contexts, technologies, or priorities change. The TME understands that while these test strategies provide solid foundations, they must be flexibly applied and continuously adjusted to remain effective as systems and requirements evolve.

Process-compliant test strategies follow established standards like ISO, IEEE, the FDA, Agile software development, or organizational processes. Success depends on proper understanding and implementation of chosen standards or processes, with alignment to project contexts. Defects occur through misapplication, inadequate training, or forcing standards into incompatible situations.

© International Software Testing Qualifications Board

Reactive test strategies involve dynamic (heuristic and/or empirical) test techniques where test designs adapt quickly to changes in business models, requirements, priorities, technologies, and emerging risks. These test strategies excel in rapidly evolving environments and can effectively respond to unexpected failures as they arise. However, the primary risk lies in potential delays in identifying critical failures, as test activities may lag development changes or miss important defects until late in the process. Success depends on maintaining adequate visibility into coverage and providing timely feedback loops to prevent late discovery of significant defects.

Consultative test strategies rely on stakeholders' input to define test objectives and priorities. They succeed when stakeholders understand business needs and provide consistent guidance. Defects arise from conflicting stakeholders' priorities, inadequate stakeholders' knowledge, or insufficient consultation coverage.

Regression-averse test strategies focus on preventing regression through extensive test automation and systematic confirmation testing. These test strategies are crucial for iterative development models but require automated test processes, dedicated and maintainable test automation infrastructure, and appropriate tooling. Defects typically stem from test activities not being ready for test automation, software and test automation of technical debt, inadequate tool selection, or insufficient test automation skills.

Test strategy blending often provides optimal solutions, combining complementary approaches while mitigating individual test strategy weaknesses. At the expert level, determining how to blend test strategies may involve structured techniques such as a weighted decision matrix or a SWOT analysis to assess strengths, weaknesses, opportunities, and threats of each approach before combining them. Effective blending requires understanding test strategy interactions, test resource allocation across approaches, and coordination mechanisms. Analysis should consider organizational maturity, project timeline, budget, risk appetite, test automation, and stakeholders' expectations when selecting test strategy combinations.

1.2.2 Test Strategy Failure Analysis

Test strategy failures can result from misalignment between test strategy characteristics and project realities, inadequate implementation, changing circumstances that invalidate original assumptions, incorrect implementation of the test policy, or even the absence of a relevant test policy. Systematic failure analysis enables learning and improvement for future test strategy selection.

Common failure patterns include strategy-context mismatches, where chosen approaches do not align with project characteristics. For example, applying methodical test strategies to rapidly evolving systems or using consultative test strategies when stakeholders lack domain knowledge. Test resource-related failures occur when test strategies demand capabilities beyond organizational capacity, such as model-based testing without statistical expertise or regression-averse test strategies without technical infrastructure.

Implementation failures arise from inadequate execution despite appropriate test strategy selection. Poor documentation practices undermine analytical test strategies, while insufficient stakeholders' engagement compromises consultative test strategies. Training deficiencies lead to reactive test strategy failures, and inadequate tool integration causes regression-averse test strategy breakdowns. Communication gaps between test strategy definition and test teams create alignment problems.

Environmental factors contribute significantly to test strategy failures. Organizational culture mismatches occur when formal test strategies conflict with informal practices or when process-oriented approaches clash with flexible organizational styles. Technology constraints limit test strategy options, particularly for test automation-dependent approaches. Schedule pressures may force premature strategy abandonment or inadequate implementation.

© International Software Testing Qualifications Board

Stakeholder-related failures include insufficient buy-in, conflicting expectations, or changing priorities that undermine test strategy effectiveness. Decision-maker turnover can invalidate consultative test strategies, while budget cuts may eliminate test resources needed for chosen test strategies. Test scope changes may render selected test strategies inappropriate for new requirements. Furthermore, partial outsourcing of test activities as a cost reduction measure may also impact the choice of test strategies to be implemented.

Test measurement inadequacies prevent early failure detection and course correction. Test strategies may appear successful while failing to achieve intended objectives due to inappropriate test metrics or insufficient test monitoring. Lack of baseline measurements makes improvement assessment impossible, while delayed feedback prevents timely adjustments.

Test strategies for addressing failures include test strategy pivoting, where alternative approaches replace failing test strategies, and test strategy blending, where additional test strategies supplement struggling approaches. Test resource reallocation may address capability gaps, while stakeholder re-engagement can resolve communication problems. Sometimes controlled test strategy abandonment with graceful transition to alternatives proves the most effective.

Failure analysis should examine root causes rather than symptoms, considering both immediate triggers and underlying systemic issues. This analysis informs future test strategy selection, organizational capability development, and process improvements to prevent recurring failures.

1.2.3 Creating Test Strategy Documents

Purpose, audience, and ownership of the test strategy. The goal of a test strategy is to provide a coherent framework for achieving organizational quality objectives through testing. It is developed for all relevant stakeholders, including executives, technical teams, and project managers, to align their efforts toward these quality objectives. The test strategy owner may be the TME or equivalent, an executive role, accountable for its creation, maintenance, and successful implementation. Executive stakeholders require high-level summaries focusing on business value, risk management, and resource requirements. Technical stakeholders need detailed implementation guidance, tool specifications, and integration procedures. Project stakeholders require coordination of information, deliverable schedules, and communication protocols. Stakeholders also need to understand their roles and time requirements so they can plan accordingly.

Recommended content of test strategy documentation. Test strategy documentation aims to guide test efforts toward achieving defined quality objectives by providing roadmaps for implementation, defining approaches, enabling success factors, and establishing risk mitigation measures as well as defining acceptable risks which do not require risk mitigation. Depending on its purpose, effective documentation may balance strategic vision with practical implementation guidance or emphasize one over the other to meet specific organizational or project needs.

According to widely used industry guidance such as the ISO/IEC/IEEE 29119 standard, essential documentation elements include an executive summary that presents the test strategy rationale and key decisions, the test scope definition that establishes the test boundaries, and the test objectives, and also the test strategy selection justification explaining the chosen approaches. Risk assessment sections identify potential failures and risk mitigation test strategies, while test resource requirements specify necessary capabilities, tools, and staffing.

The test strategy specification details selected approaches with implementation guidance. Each component, such as risk-based testing, test automation approach, or non-functional testing, should include test objectives, success criteria, required test resources, dependencies, and integration points with other test strategies. Detailed procedures for test strategy execution, including templates, checklists, and workflow descriptions, are documented either as part of the test strategy itself or in supporting process documentation, enabling consistent implementation.

© International Software Testing Qualifications Board

The test strategy development process begins with situational analysis where the TME assesses organizational context, project constraints, stakeholders' requirements, and technical environments. This analysis phase involves gathering input from key stakeholders, reviewing available test resources, and identifying potential risks and challenges.

The process continues with approach selection, where the TME evaluates different test strategies against identified requirements and constraints. Decisions are made collaboratively with project stakeholders to select optimal combinations of methodical, reactive, analytical, or other test strategies based on project-specific factors.

Test strategy formulation follows, involving detailed test planning of selected approaches, test resource allocation, timeline development, and risk mitigation measures. The TME aligns the test plan with the test strategy, organizational practices, and project objectives.

The process concludes with validation and approval phases, where the developed test strategy undergoes stakeholders' review, pilot testing when feasible, and formal approval. Implementation planning establishes deployment timelines, training requirements, and test monitoring mechanisms to support successful test strategy execution.

Test strategy maintenance and continuous improvement require ongoing attention throughout the project lifecycle. As projects evolve and circumstances change, test strategies must be regularly reviewed and updated to remain relevant and effective. This maintenance process begins with establishing clear governance frameworks that define how test strategy documentation will be reviewed, approved, and modified over time. Successful test strategy maintenance relies on systematic change management processes that capture emerging requirements, lessons learned, and environmental changes that may impact test approaches. The TME regularly assesses test strategy effectiveness through defined metrics and stakeholders' feedback, using this information to guide necessary adjustments and improvements.

QA plays a crucial role in maintaining test strategy coherence, involving peer reviews, stakeholder validation sessions, and regular alignment checks with organizational policies and standards. Throughout this process, effective communication provides that all stakeholders remain informed of test strategy changes and understand their implications for project execution.

The ultimate goal is to create living documentation that evolves intelligently with project needs while maintaining strategic focus and stakeholders' alignment, enabling test organizations to continuously improve their approach based on real-world experience and changing business contexts.

1.3 Aligning the Test Policy with the Test Strategy within the Organization

1.3.1 Aligning the Test Policy with the Test Strategy

Test policies and test strategies require continuous evaluation and improvement to maintain organizational alignment and optimize test value delivery. The TME must assess current approaches against organizational context factors, stakeholders' need, and evolving contexts to identify improvement opportunities and implement enhancements.

Evaluation framework for test policy and test strategy alignment. Test policy and test strategy evaluation require systematic assessment across multiple dimensions such as practicality, completeness, and consistency.

© International Software Testing Qualifications Board

Practicality evaluation examines implementation feasibility across all applicable test levels, considering test resource constraints, skill availability, and technical infrastructure, specifically in relation to how these factors enable or hinder the execution of the test policy and the test strategy as defined. Practical analysis includes whether test strategies can be consistently applied, whether required test tools and techniques are accessible, and whether implementation timelines align with organizational capabilities and strategic objectives.

Completeness evaluation assesses whether test policies and test strategies address all essential elements for successful implementation and whether they cover the full scope required for organizational alignment. This includes examining coverage of test objectives, stakeholders' requirements, risk management approaches, quality gates, and success criteria. Completeness analysis should identify gaps in documentation, missing stakeholders' considerations, and inadequate guidance for implementation teams that prevents full alignment with organization context.

Internal consistency evaluation examines the alignment between test policy objectives without contradicting established principles. External consistency, in this context, refers to the coherence between different test strategy components and the overarching test policy, with explicit focus on confirming both remain aligned with the organization’s quality goals and operational environment.

Short- and long-term perspective evaluation. Short-term perspective evaluation focuses on immediate implementation effectiveness, measured by achievement of planned coverage, schedule adherence, and process compliance, as well as project success rates and stakeholders' satisfaction, all in the context of fulfilling test policy and test strategy objectives. Assessment considers test resource efficiency (i.e., actual versus planned usage), defect detection percentage (DDP), and delivery timeline adherence. Metrics may include test execution productivity, test automation coverage growth, and stakeholders' feedback scores, provided they are relevant to assessing the success of test policy and test strategy execution.

Long-term perspective evaluation considers strategic sustainability, (i.e., the ability of the test approach to remain effective over time), and organizational capability development, meaning the growth of skills, processes, and test resources that strengthen the test organization’s performance in executing its test policy and test strategy. It also examines adaptability to changing business environments. Long-term assessment examines test process maturity evolution, knowledge retention and transfer, technology investment returns, and competitive advantage maintenance. Strategic metrics include test organization capability growth, process improvement implementation success, and adaptation speed to new technologies or methodologies.

Organizational assessment. Process evaluation examines how well-defined processes support and integrate with the test policy and test strategy, their alignment with broader organizational workflows, standardization effectiveness, and continuous improvement mechanisms. Assessment should consider the currency of process documentation, compliance measurement, and improvement of process effectiveness.

A people evaluation assesses team capability alignment with test strategy requirements, skill development program effectiveness, and organizational culture fit, focusing on how human factors enable test policy and test strategy goals. This includes evaluating training program outcomes, career development path clarity, and team satisfaction levels. A people evaluation considers competency gaps, succession of test planning effectiveness, and knowledge-sharing mechanisms, specifically in terms of supporting organizationally aligned test execution.

A test tool and systems evaluation examines technology alignment with test policy and test strategy objectives, test tool ecosystem integration, and infrastructure scalability in the organizational context. Assessment should consider test tool utilization rates, integration effectiveness, maintenance overhead, and technical debt, so that test tools actively support strategic and test policy alignment. Test tool evaluation includes analyzing test automation framework sustainability, test reporting system effectiveness, and data management capabilities.

© International Software Testing Qualifications Board

A test technique assessment evaluates whether selected test techniques are appropriate for executing a test policy and a test strategy in different contexts and whether they contribute to maintaining organizational alignment. This includes examining test technique coverage adequacy, application consistency, and test result quality. These test technique assessments consider emerging method evaluation, training effectiveness, and continuous learning implementation, so that innovations remain consistent with strategic direction and test policy principles.

Improvement implementation strategy. Improvement identification requires gap analysis, systematic analysis of test results, stakeholders feedback integration, and benchmark comparison, explicitly linking identified issues to gaps in organizational alignment between a test policy and a test strategy. The TME should prioritize improvements based on impact potential, implementation of feasibility, and strategic alignment. Improvement planning should consider changing management requirements, test resource allocation needs, and timeline constraints relevant to the updated test policy and test strategy.

Implementation planning involves establishing improvement roadmaps, defining success criteria, and creating test monitoring mechanisms to track alignment progress. Plans should address stakeholders' communication, training requirements, and risk mitigation strategies. Implementation should include pilot programs, gradual rollout strategies, and feedback collection mechanisms focused on validating improved alignment.

Continuous improvement cycles support ongoing optimization through regular evaluation, adjustment, and enhancement activities, directly tied to test policy and test strategy objectives. These cycles should incorporate lessons learned, emerging best practices, and changing organizational needs. Improvement of sustainability requires establishing governance frameworks, measurement systems, and knowledge capture mechanisms to preserve alignment over time.

Organizational diversity. Different organizational contexts require tailored approaches to test policy and test strategy alignment. The TME must evaluate organizational culture, business model characteristics, and regulatory environments to determine appropriate customization levels. This includes assessing departmental variation needs, project-specific adaptations, and consistency maintenance requirements, while confirming core alignment principles are preserved.

Successful alignment balances standardization benefits with contextual adaptation needs, so that policies provide sufficient guidance while allowing appropriate flexibility for diverse organizational requirements and changing business conditions.

© International Software Testing Qualifications Board

2 Managing the Test Team – 600 minutes

Keywords

(none)

Domain Specific Keywords

DEI principles, psychological safety, S.M.A.R.T. goal methodology

Learning Objectives for Chapter 2:

2.1 Building the Test Team

ELTM-2.1.1 (K5) Evaluate candidate suitability for strategic test roles, considering emerging technologies, DEI principles, and adaptability to DevOps/test automation.

ELTM-2.1.2 (K4) Outline the major points of a training program for new people

ELTM-2.1.3 (K2) Explain considerations that must be made when terminating an employee or contractor

2.2 Developing the Test Team

ELTM-2.2.1 (K5) Evaluate leadership and communication approaches using strategic stakeholders' management and metrics aligned with organizational goals

ELTM-2.2.2 (K2) Explain the importance of an individual understanding their role and responsibilities within a test team

ELTM-2.2.3 (K4) Analyze the TME skills to identify areas for skill development and learning options including training and mentoring

ELTM-2.2.4 (K3) Apply performance reviews that address both high performers and poor performers, setting appropriate goals and development plans for each

2.3 Leading the Test Team

ELTM-2.3.1 (K5) Evaluate the information and guidance needed by the test team, adjust management and leadership approach based on circumstances to determine the most effective method of communication

ELTM-2.3.2 (K4) Analyze team dynamics and organizational structure to design team-building activities that foster trust, loyalty, and communication across distributed teams

ELTM-2.3.3 (K2) Explain the challenges of motivating a test team, both short term and long term

© International Software Testing Qualifications Board

Introduction to Managing the Test Team

Introduction

Effective test team management represents a critical competency for TMEs in software development environments. Test management must encompass strategic hiring decisions, team development planning, adaptive leadership approaches, and performance optimization. The evolution of Agile software development, DevOps, and distributed work arrangements has transformed test team dynamics, requiring new approaches to collaboration, communication, and motivation. The TMEs must master both technical competencies and human-centered leadership skills. This chapter addresses four essential management domains: building effective teams, developing capabilities, leading through adaptive communication, and fostering collaboration through team building activities.

2.1 Building the Test Team

Introduction

Building a test team requires decisions around hiring, onboarding, developing, and offboarding testers in line with the test policy and human resources practices of the organization. The TME must evaluate candidates (or external teams) thoroughly, create comprehensive onboarding programs, and handle employment transitions professionally, considering the organizational context, constraints, needs, and required skills.

2.1.1 Resume Evaluation, Interview Design, and Candidate Selection

Effective candidate selection requires systematic resume evaluation followed by strategic interview design to make optimal hiring decisions. This comprehensive process provides thorough assessment of technical capabilities, cultural fit, and growth potential.

Resume evaluation. Candidate selection prioritizes strategic alignment with organizational objectives over operational screening steps. Evaluation focuses on adaptability to evolving approaches such as DevOps, AI-driven testing, and test automation expansion. The following aspects must be considered: long-term capability growth, leadership potential, and advanced DEI principles, (i.e., diversity, equity, and inclusion) to build teams that reflect organizational values and can sustain performance in rapidly changing environments. Operational screening processes, as covered in (ISTQB-TM), remain a prerequisite. Assessment of resume quality provides insights into communication skills and attention to detail. Defect-free, well-structured documentation suggests candidates capable of producing quality testware. Gaps in employment history, frequent job changes without progression, or misalignment between claimed skills and project responsibilities require investigation.

Defining an interview strategy. Interview approaches need to assess not only the current role fit but also alignment with the organization's future direction and strategic initiatives. The recruitment process should be strategically aligned with the test policy and test strategy goals outlined in the previous chapter, so that candidate acceptance criteria directly support organizational objectives and transformation initiatives. A candidate's ability to lead change may be evaluated, because it contributes to a culture of psychological safety, diversity, and inclusion. For technical roles, forward-looking problem scenarios are included (e.g., integrating AI-based tools into test pipelines) alongside skills assessment. Behavioral interviews can be used to prove adaptability, resilience, and capability to influence cross-functional teams in dynamic environments.

© International Software Testing Qualifications Board

Interview process design. The interview structure should be tailored to the specific role and its criticality within the organization. For critical positions (e.g., test managers or senior test leaders), multi-stage interview processes need to be structured, incorporating phone/video screening, technical assessments, panel interviews, and practical demonstrations. For standard team positions, a streamlined approach with focused technical and behavioral assessment may be more appropriate. When engaging external companies or contractors, the interview process should be adapted to evaluate the vendor's team capabilities, test methodology, and cultural fit, often involving presentations of their proposed approach, case studies from similar engagements, and meetings with key personnel who will be assigned to the project. Diverse interview teams should be included across all hiring scenarios, providing varied perspectives and reducing unconscious bias. Standardized question sets enable consistent candidate comparison while allowing flexibility for follow-up exploration based on role requirements and hiring context.

Comprehensive candidate assessment. Candidates should be evaluated across multiple dimensions including technical expertise, communication effectiveness, analytical thinking, and team collaboration potential. The assessment can check adaptability to organizational culture, future adaptability to evolving practices such as DevOps, AI and test automation, learning agility, and professional growth trajectory. Balance is needed between current capabilities and development potential, particularly for emerging technologies and evolving test practices. Advanced diversity and inclusion principles should be incorporated, aiming to build teams with varied backgrounds, perspectives, and experiences, which can improve coverage and innovation in global contexts. Current team members can also be involved in candidate assessment.

Structured interview design. A structured interview increases the reliability and validity of candidate evaluation by confirming that all candidates are assessed against the same role-related competencies. The interview should be designed before candidate meetings, by defining competencies to be evaluated, standardized questions, practical exercises where appropriate, and explicit evaluation criteria. Each competency should be assessed using multiple sources of evidence, such as behavioral questions, technical scenarios, practical demonstrations, and discussion of previous experience. Questions should focus on competencies required for the role rather than personal characteristics. Behavioral questions based on previous experience (e.g., describing how the candidate approached risks during a DevOps transformation) can be combined with situational questions that explore decision making in future scenarios (e.g., introducing AI-supported test generation while maintaining test quality). Interviewers should distinguish between mandatory competencies and desirable capabilities, allowing consistent comparison while recognizing different career paths. Structured scoring guides should define observable indicators for each rating level, reducing subjective interpretation by interviewers. Interview notes should capture factual evidence supporting each score rather than general impressions. Follow-up questions may explore candidate responses in greater depth but should remain aligned with the predefined competencies to maintain fairness and comparability.

Bias awareness framework. Strategic recruitment requires awareness that cognitive biases may influence hiring decisions despite good intentions. A bias awareness framework helps interviewers recognize and reduce systematic errors while maintaining competency-based selection. Typical biases include:

- Confirmation bias (i.e., seeking evidence supporting an initial impression)
- Halo/horn effect (i.e., allowing one characteristic evaluation to influence evaluation of other characteristics)
- Affinity bias (i.e., favoring candidates with backgrounds similar to the interviewer)
- Anchoring (i.e., placing excessive weight on early information)
- Recency bias (i.e., overemphasizing the latest interview)
- Stereotype bias based on demographic or social characteristics

© International Software Testing Qualifications Board

Bias may also arise from assumptions regarding career gaps, abnormal career paths, or communication styles that differ from interviewer expectations. Bias mitigation should be integrated throughout the recruitment process. Before interviews, competencies and evaluation criteria should be clearly defined. During interviews, standardized questions, independent scoring by multiple interviewers, and evidence-based evaluations should be used before discussion among the interview panel. After interviews, documented scoring should be reviewed to identify inconsistencies, unsupported judgments, or systematic patterns affecting particular candidate groups. These practices complement DEI principles by promoting fair, objective, and transparent selection decisions while maintaining the required competency standards for strategic test roles.

Decision-making process. Weighted scoring frameworks assess alignment with organizational objectives (e.g., capability to support AI-driven testing) allowing candidates to compare essential and preferred criteria. Evaluation rationale should be documented, supporting hiring decisions and providing feedback for unsuccessful candidates. Advanced diversity, equity, and inclusion strategies can be applied to reduce bias and increase team diversity while maintaining competency-based selection standards.

2.1.2 New Employee Training Program

Comprehensive training programs accelerate new hire integration and establish consistent organizational practices. Effective programs combine technical skill development with cultural assimilation. There should be no difference in training programs and skill enhancement for internal and external staff. Training approaches for external organization personnel or contractors may, however, require different contractual considerations.

Foundation training components include an organizational overview covering organization culture, values, and test policy, test frameworks, standards, and procedures specific to the organization, and the tool ecosystem including test management, defect management, test automation frameworks, and test environment access procedures.

Progressive skill development. The first step focuses on system access, basic tool navigation, and process documentation review. Next follows hands-on test activities with shadowing experienced team members. Subsequent weeks of progress through increasingly complex test conditions with reduced supervision.

Mentorship integration. Experienced team members are assigned as mentors to provide ongoing guidance and feedback. Regular check-in schedules provide consistent support throughout the onboarding period. Structured knowledge transfer sessions are created to cover undocumented institutional knowledge and best practices.

Assessment and milestone verification is performed by defining measurable competency and skill development checkpoints providing effectiveness, implementing practical assessments demonstrating ability to execute test activities independently, documenting training completion/cancellation, and identifying development areas for ongoing professional growth.

Documentation and continuous improvement include maintaining updated training materials reflecting current tools and processes, gathering feedback from new hires and mentors to refine program effectiveness, and creating reusable training assets by reducing preparation time for future onboarding cycles.

Training approaches incorporate virtual reality simulations, microlearning modules, and adaptive learning technologies enhancing engagement and retention rates.

© International Software Testing Qualifications Board

2.1.3 Employment and Contractor Termination Considerations

Termination of contract requires careful consideration of legal, operational, and ethical factors. Proper planning protects organizational interests while maintaining professional standards.

Legal and test policy compliance. Collaboration with people and culture (also known as human resources) supports adherence to employment law, organizational policies, and union agreements where applicable. Documentation requirements vary by jurisdiction and termination type, requiring thorough preparation. Performance improvement plans may be mandatory before termination for cause.

Knowledge transfer and documentation part. The TME identifies critical knowledge held by departing individuals including system configurations, test data locations, and specialized procedures. Next, the TME enables comprehensive handover processes to provide continuity of test activities and secures intellectual property and organizational assets before termination.

Timeline and test communication planning. The TME coordinates termination timing with project schedules to minimize disruption to test activities. The TME plans communication strategies towards team members and stakeholders and considers security implications including system access revocation and confidentiality requirements.

Contractor-specific considerations. The TME reviews contractual obligations including notice periods, deliverable completion requirements, and transition responsibilities. The TME evaluates early termination clauses and associated financial implications.

Termination practices emphasize dignity, transparency, and support for leaving employees while protecting organizational interests.

2.2 Developing the Test Team

Effective test team development requires strategic planning, clear role definition, continuous skill enhancement, and performance management. TMEs must create environments that foster individual growth while aligning organizational objectives. The TME uses a skill matrix with detailed required skills of the test team.

2.2.1 Test Team Development Planning and S.M.A.R.T. Goal Methodology (SMART)

Creating effective development plans requires comprehensive analysis of individual capabilities, organizational requirements, and strategic objectives. This process promotes optimal resource allocation and meaningful professional growth.

Individual assessment framework. The TME analyzes tester backgrounds through competency mapping, identifying technical skills, domain expertise, certification levels, and experience patterns. The TME evaluates soft skills including communication effectiveness, analytical thinking, and collaboration capabilities. The TME considers career aspirations, learning preferences, and motivation factors influencing development success.

The TME reviews current role performance against established benchmarks, identifying strengths to leverage and gaps requiring attention and assesses potential for advancement or lateral movement within the organization. The TME assesses learning agility through previous skill acquisition and adaptation to new technologies or methodologies.

© International Software Testing Qualifications Board

Organizational needs analysis. The TME examines strategic test initiatives requiring specific competencies including test automation frameworks, emerging technologies, compliance requirements, or specialized test domains. The TME identifies succession test planning and backup needs for critical roles and knowledge transfer requirements. The TME also considers budget constraints, timeline expectations, and test resource availability affecting the development of investments.

The TME analyzes test team composition for skill distribution and collaboration dynamics. They also evaluate external factors such as industry trends, regulatory changes, or technological evolution impacting required competencies.

SMART. SMART is a structured framework for creating effective, actionable objectives that drive meaningful progress. Each letter represents a critical component that transforms vague intentions into concrete, achievable targets.

Specific (S) objectives require clear articulation of exactly what will be accomplished, who is involved, and what the expected results look like. Rather than setting broad aspirations, specific goals eliminate ambiguity by defining precise success criteria and deliverables.

Measurable (M) elements confirm that progress can be tracked and evaluated through quantifiable metrics, certification achievements, or demonstrable skill improvements. These benchmarks provide concrete evidence of advancement and enable course corrections when needed.

Achievable (A) targets balance ambition with realism, considering individual capacity, available organizational support, and reasonable timeframes. Goals should stretch capabilities while remaining within the realm of possibility given current test resources and constraints.

Relevant (R) goals align personal development with broader career progression and organizational objectives. This connection provides that individual growth contributes meaningfully to test team success and strategic priorities.

Time-bound (T) components establish clear milestones with regular progress checkpoints and firm completion deadlines. Time-bound goals create urgency and accountability while preventing indefinite postponement.

When implementing SMART, avoid common pitfalls such as focusing on uncontrollable metrics like defect discovery rates or production failures. Instead, concentrate on outcomes that can be influenced. Additionally, integrate diversity and inclusion objectives into development planning to leverage varied perspectives and strengthen test team innovation capabilities.

Integration and monitoring. Individual development plans should be aligned with test team objectives and organizational strategy. The TME creates accountability mechanisms through regular review sessions and test progress reporting. They establish feedback loops enabling plan adjustments based on changing circumstances or emerging opportunities.

Development planning should incorporate adaptive learning pathways, microlearning approaches, and competency-based progression models reflecting professional development practices.

2.2.2 Role Clarity and Test Team Responsibilities

Clear role definitions promote effective test team functioning and individual accountability. Understanding responsibilities prevents conflicts, reduces duplication, and enables optimal task allocation.

Individual role understanding. Test team members must comprehend their specific responsibilities within test activities (including test design, test execution, and test reporting) and QA. Understanding test scope boundaries prevents role confusion and supports appropriate escalation procedures. Clear expectations enable self-directed work and autonomous decision-making within defined parameters.

© International Software Testing Qualifications Board

Test team integration and collaboration. Roles interconnect through shared objectives, deliverable dependencies, and communication protocols. Understanding colleague responsibilities facilitates effective collaboration and mutual support. Knowledge of test team dynamics enhances collective problem-solving and knowledge sharing capabilities.

Organizational context. Test team roles align with broader organizational structures and external stakeholders' expectations. Understanding test reporting relationships, decision-making authority, and escalation procedures supports appropriate professional interactions. Role clarity supports effective performance evaluation and career development planning.

Specialized position awareness. Test organizations should include diverse specialized roles such as test automation engineers, performance analysts, security testers, accessibility specialists, or AI-specific roles. Understanding these specializations helps individuals identify career paths and development opportunities while appreciating colleague expertise and contribution to diversity.

2.2.3 Skills Development and Learning Options

Continuous skill development maintains relevance in evolving test landscapes while addressing individual and organizational capability gaps. Strategic learning approaches maximize development effectiveness and ROI.

Skill development areas. Technical competencies require ongoing enhancement including test automation frameworks, emerging tools, programming languages, and test methodologies. Skill development areas should explicitly include AI-related skills, such as using AI-powered test tools, understanding AI model behaviors, and integrating AI-driven test automation into existing workflows, so that the test team remains relevant in an evolving industry. Domain knowledge development encompasses business understanding, regulatory requirements, and industry-specific practices relevant to organizational context.

Soft skills development addresses communication effectiveness, leadership capabilities, analytical thinking, and collaborative problem-solving abilities. Cross-functional skills including DevOps practices, business analysis, cloud technologies, and Agile software development expand professional versatility and career opportunities.

Training option selection. Formal training through professional organizations, certification programs, and structured courses provides systematic knowledge acquisition with recognized credentials. Internal training leverages organizational expertise while building institutional knowledge and test team cohesion.

Self-directed learning through trusted online platforms, technical documentation, and industry publications offer flexible, cost-effective skill development. Hands-on experience through pilot projects, proof-of-concepts, and experimental implementations provides practical application opportunities.

Mentoring and knowledge transfer. Internal mentoring relationships pair experienced professionals with development team members, facilitating knowledge transfer and cultural integration. External mentoring through professional networks and industry associations provides broader perspective and career guidance.

Reverse mentoring arrangements enable senior professionals to learn emerging technologies from junior colleagues while fostering collaborative learning environments. Cross-functional mentoring with developers, business analysts, and operations teams enhances understanding of the broader SDLC.

Learning path optimization. Progressive skill building through foundational knowledge followed by advanced specialization promotes sustainable development. Practical application opportunities reinforce theoretical learning while demonstrating competency acquisition. Regular assessment and feedback guide learning effectiveness and path adjustments.

© International Software Testing Qualifications Board

Learning approaches should incorporate microlearning modules, virtual reality simulations, and adaptive learning technologies enhancing engagement and retention while accommodating diverse learning preferences and schedule constraints.

2.2.4 Performance Review Management

Effective performance reviews require differentiated approaches for varying performance levels while maintaining fairness, consistency, and development focus. Strategic review processes support both individual growth and organizational objectives.

High performer management. The TME recognizes exceptional contributions through specific achievement of acknowledgment and impact documentation. They identify stretch opportunities including leadership roles, complex projects, or cross-functional assignments with challenging capabilities while maintaining engagement. They explore advancement pathways through promotion possibilities, role expansion, or specialized responsibility areas.

The TME addresses potential retention risks through career development discussions, competitive compensation reviews, and professional growth investments. High performers are leveraged as mentors, knowledge sharers, and change champions supporting team development and organizational improvement initiatives.

Poor performer improvement. The TME diagnoses performance defects through root cause analysis (RCA) distinguishing skill gaps, motivation challenges, role misalignment, or external factors affecting productivity. They develop targeted improvement plans with specific behavioral expectations, skill development requirements, and measurable progress indicators and define an acceptable timeframe for the demonstration of improvement.

The TME establishes support mechanisms including additional training, mentoring assignments, or workload adjustments enabling success. They create accountability structures with frequent check-ins, clear milestone expectations, and progress reporting. Role reassignment or test team restructuring can be considered if improvement efforts prove ineffective.

Goal setting strategies. The TME designs performance-level appropriate objectives for balancing challenge and achievability. High performers receive ambitious targets encouraging innovation and leadership development. Poor performers receive focused, achievable goals, building confidence, and demonstrating capability improvement.

The TME provides goal alignment with organizational priorities while addressing individual development needs. They establish successful metrics by emphasizing controllable factors rather than external variables beyond individual influence.

Review process effectiveness. The TME prepares comprehensive performance documentation including objective work product evaluation, stakeholders' feedback, and behavioral observation records. They conduct structured review conversations focusing on factual performance discussion, development opportunity identification, and future expectation clarification.

Performance management emphasizes continuous feedback, real-time recognition, and agile goal adjustment reflecting proper workplace dynamics and generational preferences for frequent, meaningful professional dialogue.

Using metrics for test team health and performance. Metrics should track, with appropriate frequency, both performance outcomes and test team well-being to enable balanced management decisions. Examples include throughput trends, skill growth rates, and test team satisfaction surveys. Avoid metrics that incentivize counterproductive behaviors, such as measuring defect counts without context.

© International Software Testing Qualifications Board

2.3 Leading the Test Team

Effective test team leadership requires adaptive management approaches, strategic communication, and comprehensive understanding of test team dynamics. TMEs must navigate complex organizational structures while fostering collaboration, motivation, and high performance across diverse test team configurations.

2.3.1 Adaptive Leadership and Communication Strategy

Successful test team leadership demands continuous evaluation of situational factors and strategic adjustment of management approaches to optimize test team effectiveness and communication outcomes. Using models such as Tuckman's stages of group development (Tuckman, 1965) may help the TME to adapt their management approach regarding test team management.

Information and guidance assessment. The TME evaluates team information needs through systematic analysis of project complexity, stakeholders' expectations, regulatory requirements, and technical dependencies. They include stakeholders' management practices to provide alignment, secure test resources, and reinforce the test organization's value. Metrics should be presented to provide actionable insights without creating perverse incentives. The TME should consider individual test team members' experience levels, domain knowledge, and skill development requirements when determining information depth and delivery frequency and should finally assess external communication demands including executive test reporting, cross-functional collaboration, and client interaction requirements.

The TME analyzes current information flow effectiveness by identifying gaps, redundancies, and communication barriers. The TME evaluates existing documentation quality, accessibility, and relevance to current project objectives, and consider temporal factors affecting information currency and decision-making timeframes.

Situational leadership adaptation. The TME adjusts management style based on test team maturity, project phase, organization culture, and stakeholders' pressures. More directive approaches may be applied during crisis situations. Inexperienced test team members require structured guidance. The TME implements coaching styles for developing team members needing skill enhancement and confidence building.

The TME uses supportive leadership for competent test team members facing motivational challenges or external pressures. They deploy delegating approaches with experienced, self-motivated professionals capable of autonomous decision-making. They consider cultural factors, generational differences, and individual personality types when selecting appropriate leadership styles.

Communication method optimization. The TME selects communication channels based on message urgency, complexity, audience preferences, and geographic distribution, and uses face-to-face interactions (if possible) for sensitive discussions, complex problem-solving, and relationship building. They implement digital collaboration tools for routine updates, document sharing, and asynchronous communication across time zones.

The TME designs communication frequency and format considering project volatility, stakeholders' concern levels, and test team confidence requirements. They create communication protocols addressing escalation procedures, decision-making authority, and information security requirements, and establish feedback mechanisms supporting bidirectional communication effectiveness and continuous improvement.

© International Software Testing Qualifications Board

Strategic communication planning. The TME develops comprehensive communication strategies aligning with organizational objectives, project timelines, and stakeholders' expectations. They create communication calendars coordinating test team meetings, stakeholders' updates, and critical decision points. They also design crisis communication procedures addressing rapid response requirements and stakeholders' management during challenging situations.

Leadership incorporates emotional intelligence, cultural competency, and digital communication mastery reflecting workplace dynamics and distributed test team management requirements. They also include proactive stakeholders' management to secure necessary test resources, maintain alignment with organizational objectives, and reinforce the strategic value of testing within the wider business context. Stakeholders' management strategies include regular engagement, transparent test reporting, and negotiation skills to reconcile conflicting priorities.

2.3.2 Test Team Building and Distributed Test Team Management

Effective test team building requires systematic analysis of test team composition, organizational constraints, and cultural factors to design activities fostering collaboration, trust, and communication effectiveness across geographic and cultural boundaries.

Test team dynamics analysis. Test team composition should be analyzed, including experience levels, personality types, cultural backgrounds, and professional specializations affecting collaboration patterns. The TME identifies communication preferences, work styles, and interpersonal dynamics influencing test team effectiveness. The TME assesses existing trust levels, conflict patterns, and collaborative strengths requiring reinforcement or improvement.

The TME evaluates individual motivations, career aspirations, and personal circumstances affecting test team engagement. The TME considers previous test team experiences, leadership preferences, and feedback mechanisms shaping current test team dynamics.

Organizational structure assessment. The TME analyzes test reporting relationships, decision-making processes, and test resource allocation patterns affecting test team autonomy and effectiveness. The TME evaluates organizational culture regarding risk tolerance, innovation encouragement, and collaborative behaviors. The TME also considers test policy constraints, budget limitations, and approval processes affecting test team building initiative implementation.

The TME assesses physical and virtual infrastructure supporting collaboration including meeting spaces, technology platforms, and communication tools. The TME examines time zone distributions, cultural holidays, and work schedule variations affecting distributed test team coordination.

Distributed test team considerations. The TME addresses geographic separation challenges through virtual collaboration tools, structured communication protocols, and relationship-building activities. The TME designs cross-cultural activities acknowledging cultural differences while building shared test team identity and mutual understanding (Meyer, 2016). The TME creates inclusion strategies so that remote test team members receive equal participation opportunities and recognition.

The TME may implement rotation programs to enable face-to-face interaction when feasible and establish buddy systems by pairing distributed test team members for informal support and knowledge sharing. The TME designs asynchronous collaboration methods accommodating time zone differences and individual work preferences.

© International Software Testing Qualifications Board

Psychological safety. Creating and maintaining psychological safety within the test team allows all members to feel secure in raising concerns, reporting defects, and challenging assumptions without fear of negative consequences. A psychologically safe environment supports open communication, encourages constructive disagreement, and fosters innovation, particularly in high-pressure delivery contexts. The TME should model transparency, respond constructively to feedback, and address blame culture promptly.

For distributed test teams, special attention should be paid to provide equal voice in meetings and creating informal channels where members can share concerns confidentially.

The TME also needs to protect the team from abuse from outside the team (e.g., unnecessary pressure, blame, and interruptions). The TME should be the first point of contact to not allow every stakeholder to directly pressure individual testers and to shield the team from emotional conflict. Unrealistic expectations should be challenged with facts and evidence rather than emotion.

However, protection does not mean shielding the team from all criticism. It means filtering unproductive behavior while making sure legitimate concerns are addressed.

Activity design and implementation. The TME develops formal test team building programs including problem-solving exercises, collaborative projects, and professional development opportunities. They create informal relationship-building activities such as virtual coffee sessions, test team lunches, and celebration events acknowledging achievements and milestones.

The TME designs knowledge-sharing sessions enabling test team members to present expertise, share experiences, and contribute to collective learning. They implement mentoring programs connecting experienced professionals with developing test team members across geographic boundaries.

Test team building incorporates virtual reality experiences, digital collaboration platforms, and hybrid event formats accommodating diverse participation preferences and organizational constraints. Activities should be inclusive by design, providing accessibility across cultures, abilities, and geographies, and promoting equal participation for all test team members.

2.2.3 Test Team Motivation Challenges

Test team motivation presents unique challenges requiring understanding of intrinsic and extrinsic motivational factors affecting both immediate engagement and sustained performance over extended periods.

Short-term motivation challenges. Immediate motivational issues often stem from repetitive test activities, tight deadlines, and pressure from development teams or other stakeholders. Testing can become monotonous when executing extensive regression testing suites or performing routine validation activities. Test team members may experience frustration when discovering defects late in SDLCs, struggling with discovering defects, facing resistance from developers, or dealing with unrealistic test schedule expectations.

Recognition deficits occur when test contributions receive insufficient acknowledgment compared to development achievements. Test teams often work behind the scenes, making contributions less visible to organizational leadership. Project pressure can create stress affecting test team morale and individual performance levels.

Long-term motivation challenges. Career progression concerns arise when test positions lack clear advancement pathways or professional recognition within organizations. Skill stagnation occurs when test team members perform similar activities without opportunities for growth or learning new technologies. Compensation of disparities compared to development roles can affect long-term retention and job satisfaction.

© International Software Testing Qualifications Board

Professional identity challenges emerge when test roles are undervalued or misunderstood within organizational culture. Limited exposure to business strategy and decision-making processes can reduce engagement and sense of contribution significance. Technology evolution may create anxiety about skill relevance and future employability.

Burnout management. Burnout is a common risk in testing due to sustained project pressures, repetitive tasks, and high defect resolution demands. Effective burnout prevention includes workload balancing, task variety to reduce monotony, and adequate recovery periods between high-intensity phases.

Managers should monitor signs of disengagement, declining quality, or increased absenteeism and address root causes early. Open dialogue about stressors, realistic deadline settings, and supporting mental well-being initiatives contribute to sustainable test team performance and long-term retention.

Motivational strategy considerations. Effective motivation requires balancing individual aspirations with organizational needs and project constraints. Motivators should be distinguished from so-called hygiene factors (Herzberg 1959). Recognition programs should acknowledge both individual achievements and test team contributions. Professional development opportunities including training, certification, and conference attendance to demonstrate organizational investment in test team member growth.

Motivation approaches emphasize purpose-driven work, autonomy in task execution, and mastery development opportunities reflecting workforce expectations and generational preferences for meaningful professional engagement. Psychological safety should be cultivated so that testers feel safe to raise defects without fear of negative consequences, while burnout management practices promote sustainable workload and long-term retention.

© International Software Testing Qualifications Board

3 Managing Across the Organization – 510 minutes

Keywords

test team advocacy, quality enablement, cost of quality

Domain Specific Keywords

psychological safety, time-to-learn

Learning Objectives for Chapter 3:

3.1 Advocating for the Test Team

ELTM-3.1.1 (K3) Apply strategies to promote, advocate for, and defend the test team

ELTM-3.1.2 (K4) Analyze the quantitative and qualitative benefits of the test team using defined metrics and proxies to build a business case for a given stakeholder group

3.2 Placement of the Test Team

ELTM-3.2.1 (K5) Evaluate a test team's placement within the organization against independence of testing, information flow, test resource availability, and organizational culture to recommend a placement based on the evaluation

ELTM-3.2.2 (K4) Analyze the implications of different test team placement options with respect to test effectiveness, stakeholder collaboration, and organizational quality maturity

3.3 Building Strategic Relationships

ELTM-3.3.1 (K4) Analyze a given project environment to identify key relationships and select partnership-building strategies based on a stakeholder analysis

3.4 Stakeholder Communication

ELTM-3.4.1 (K3) Apply effective communication strategies for different test stakeholder groups by selecting the appropriate communication channels, content, and timing

3.5 Integrating Tools Across the Organization

ELTM-3.5.1 (K2) Explain the factors to consider when dealing with multi-use tools

ELTM-3.5.2 (K4) Analyze the impact of a proposed change to a multi-use tool across technical, operational, and organizational dimensions, and propose mitigation strategies for the identified risks

© International Software Testing Qualifications Board

Introduction to Managing Across the Organization

Introduction

Test management must treat quality as an integral part of value creation. A test team's success depends not only on delivering high-quality software but on how it is positioned within the organization: enabling the test process, building strong stakeholder relationships, providing supportive leadership, and integrating tools that support collaboration. Mastering these skills lets the team secure test resources, build credibility, and establish sustainable partnerships that increase test effectiveness across the entire SDLC.

This chapter shows how test management promotes sustainable quality through transparent coordination, evidence-based decisions, and a psychologically safe environment. While this chapter addresses managing relationships and coordination across the organization as a whole, the specific management of external suppliers and vendors is covered in Chapter 4.

Measurable success criteria: The competencies in this chapter are demonstrated through concrete, observable outcomes rather than abstract understanding. Advocacy and value communication (see Section 3.1) are evidenced by a business case built on defined proxies, (e.g., escaped defect rate (EDR), production failure reduction, and mean time to repair (MTTR)), with decision thresholds that trigger specific actions. Placement decisions (see Section 3.2) are evidenced by a weighted assessment matrix that yields a ranked, evidence-based recommendation. Relationship building (see Section 3.3) is evidenced by a stakeholder analysis from which partnership strategies are selected and tracked through relationship indicators. Stakeholder communication (see Section 3.4) is evidenced by a stakeholder-specific communication plan defining channel, content, and timing, together with a measure of its effectiveness. Tool integration (see Section 3.5) is evidenced by a three-dimensional impact analysis (i.e., technical, operational, and organizational) with a risk register and a go/no-go decision.

3.1 Advocating for the Test Team

Introduction

TMEs practice constructive engagement and transform challenges into shared learning opportunities. Communication shifts from persuasion to transparent collaboration, supported by tools such as quality impact dashboards, typically realized through application lifecycle management (ALM) or business intelligence dashboarding tools, which visually link test results, (e.g., risk coverage, test automation effectiveness, and requirement validation), to business performance indicators such as customer satisfaction, release stability, and operational resilience. This approach shifts the role of test management toward systemic empowerment and continuously promotes quality as a shared responsibility across the organization.

3.1.1 Test Team Advocacy Strategies and Systemic Leadership

Test team advocacy is a strategic, systematic process that builds test team visibility, strengthens quality culture, and positions testing as a key enabler of sustainable business success. Through proven quality enablement strategies, the team gains recognition for its contributions and establishes itself as a systemic partner for continuous learning, quality improvement, and business value.

© International Software Testing Qualifications Board

Key quality enablement strategies include:

- **Proactive quality promotion and visibility:** Regular communication of success is data-driven and value-oriented, using meaningful test and quality reports from test management tools. The value of testing is made tangible and visible through transparent dashboards that link quality metrics directly to business outcomes, such as customer satisfaction, cost savings from defect prevention, and risk reduction. Cross-functional meetings, project showcases, and knowledge-sharing sessions highlight test expertise and demonstrate testing's contribution to product and business development. Thought leadership is established through participation in process optimization initiatives, organizational committees, and technical forums. Internal consulting and coaching of other teams builds collaborative partnerships and strengthens mutual respect. As a result, testing is perceived as a strategic quality enabler, not a policing function. Success is measured by using standard metrics such as DDP and by systemic indicators such as learning cycle effectiveness, feedback loop efficiency, and cultural health, including psychological safety and participation in continuous improvement. This continuous improvement operates as a learning loop following the Plan-Do-Check-Act (PDCA) cycle: improvement targets are derived from defect data and retrospectives (Plan), process changes are implemented (Do), their effect is remeasured through indicators such as time-to-learn and reduction in recurring defect classes (Check), and validated improvements are standardized or the loop is repeated (Act).
- **Supportive leadership and constructive engagement:** When problems or criticism arise, the test team focuses on constructive engagement rather than a defensive stance. Defects are treated as joint learning opportunities without assigning blame. The team moderates open retrospectives, root cause analyses, and lessons learned workshops. Fact-based documentation and objective analyses are used for joint root cause analysis and sustainable improvement, not for defense. Structured techniques keep this analysis rigorous and blame-free: the 5 Whys method traces a symptom to its underlying cause through iterative questioning and is well suited to linear causal chains; cause-and-effect diagrams organize potential causes into categories such as people, process, tooling, requirements, and environment and is suited to multi-factor problems; and Pareto analysis prioritizes the few defect categories responsible for most failures, focusing improvement where it matters most. Transparent communication about test constraints, dependencies, and realistic possibilities creates mutual understanding, fostering a culture of trust.
- **Organizational positioning and strategic quality enablement:** The test team consistently aligns its goals with business priorities and incorporates a quality-driven test mindset into strategic planning at an early phase. Through proactive risk identification and mitigation (rather than simply listing risks) and solution-oriented recommendations, testing is established as a value-adding factor. Executive sponsorship is achieved by regularly demonstrating business impact and positioning testing as a champion of quality that enables innovation and speed, rather than a potential bottleneck. This systemic approach transforms testing from a reactive control function into a proactive leadership role that establishes quality as a collective responsibility and competitive advantage.

Career development and team growth. A sustainable test organization invests in its members' professional growth. Test managers establish clear career paths that recognize both technical depth (e.g., test automation and security testing) and managerial breadth (e.g., test lead and test manager), supported by structured mentoring, individual development plans, cross-functional assignments, certification pathways (e.g., the ISTQB), and regular career conversations that address long-term aspirations beyond project reviews.

© International Software Testing Qualifications Board

Quality enablement also uses social media presence, conference participation, and external recognition to enhance the organization's reputation and individual careers. This external visibility builds internal credibility and creates additional leverage for internal negotiation. Sustainable advocacy balances clarity with collaboration, so that protecting the test team never compromises the organizational relationships it depends on.

Leading with psychological safety. The TME actively shapes a climate in which defects are treated as learning opportunities, and quality concerns can be raised openly through direct access to decision-makers, blame-free retrospectives and root cause analyses, anonymous feedback channels, and leadership that rewards curiosity rather than punishing it. In such teams, defects are detected earlier, risks are reported proactively, and continuous improvement becomes the norm, leading to measurably higher software quality and resilience.

3.1.2 Value Communication and Stakeholder Engagement

Test managers must be careful not to overstate the causal link between test activities and customer satisfaction. While high-quality testing reduces defects and improves system stability, both of which are prerequisites for positive user experiences, customer satisfaction ultimately depends on a broader set of factors, including correct requirements, market fit, usability, pricing, and support quality. Therefore, test teams should frame their contribution to customer satisfaction as enabling and reducing risk: they determine that the delivered product meets its specified quality characteristics, thereby preventing dissatisfaction caused by defects or poor performance. To measure this contribution, test teams use proxies such as EDRs, production failure frequency, MTTR, and system availability, (i.e., metrics that are empirically associated with customer satisfaction but do not claim exclusivity). This honest framing builds credibility with stakeholders and avoids the trap of promising what testing alone cannot deliver.

Value communication is an essential process that involves a thorough analysis of testing benefits within a specific organizational context. To be effective, this communication must be strategically presented to various stakeholders, considering their unique perspectives and decision-making criteria.

To communicate test value effectively, test managers must distinguish between three types of metrics, each serving a different purpose and audience:

- **Product metrics** assess the quality of the delivered software, (e.g., EDRs, system availability, and post-release defects). These metrics are primarily of interest to technical stakeholders and development teams, as they reflect the test results of test activities.
- **Process metrics** evaluate the efficiency and effectiveness of the test process itself, (e.g., MTTR, DDPs, test automation ROI, and feedback loop efficiency). These metrics help test managers and process owners identify improvement opportunities and demonstrate the value of test investments.
- **Project metrics** track progress against the project plan and support predictability, including schedule adherence, scope clarity, budget compliance, and release confidence. These metrics are essential for communicating with project managers, product owners, and executives who need to make informed decisions about resource allocation and release readiness.

© International Software Testing Qualifications Board

This categorization provides that the right metrics are communicated to the right stakeholders, avoiding information overload and enabling data-driven decision-making. The following sections elaborate on each category and provide guidance on their application in different organizational contexts. Beyond categorizing metrics by the audience, it is useful to distinguish leading from lagging indicators. Lagging indicators are outcome-based and measured after the fact, (e.g., EDR, post-release defects, and customer satisfaction). They confirm past performance but cannot be influenced retrospectively. Leading indicators are predictive and observed during the process, (e.g., DDP, coverage, feedback loop efficiency, learning rate, and psychological safety score). They signal likely future outcomes and let the test manager act before problems materialize. Effective value communication combines both: leading indicators to guide and lagging indicators to confirm.

The metrics mentioned above relate to different phases of the SDLC; their significance only arises from this positioning. Defect-related indicators can be categorized along the SDLC from prevention through detection to resolution:

SDLC Phase	**Defect concept / activity**	**Associated metric / proxy**	**Cost-of-Quality Reference**
Requirements / Design	Defect prevention (shift left)	Reduction of early detected defects	Prevention costs
Development / Test	Defect detection	Defect detection / discovery rate	Appraisal / internal failure
Release / Production	Escaped defects	EDR, production failures	External failure costs
Operation / Maintenance	Defect resolution	MTTR, system availability	External failure costs
Retrospective / Improvement	Recurrence prevention	Reduction in recurring defect classes, time-to-learn	Prevention (follow-up period)

Table 1. Categorization of Defect-related Indicators

These phase-specific measurements become comparable and actionable only through end-to-end traceability across the SDLC: requirements, design work products, test cases, and defects are linked so that a defect can be followed from the requirement in which it originated to the failure in which it was resolved. This requires the underlying tools, (e.g., requirements management, version control, test management, defect management, and continuous integration/continuous delivery (CI/CD)), to be integrated. Full SDLC coverage is addressed in Section 3.5.

© International Software Testing Qualifications Board

Operationalizing metrics using decision thresholds and triggers. Metrics only become actionable when they are linked to clear decision thresholds. Test managers must define, in collaboration with stakeholders, what constitutes good versus critical performance for each metric, and what actions to take when thresholds are breached.

- EDR: A threshold of > 2% may trigger an RCA and process review; > 5% may halt release and require corrective action.
- MTTR: A threshold of > 48 hours for critical defects may trigger escalation and test resource reallocation; > 72 hours may require a formal defect review.
- Production failures: An increase of > 10% per quarter may trigger a quality audit and test process improvement initiative.
- Coverage: Falling below 80% may trigger additional test design and execution; below 60% may require a release decision review and risk reassessment.
- Psychological safety score: A score below 3.5/5 may trigger team workshops and leadership coaching; below 3.0/5 requires immediate intervention.

These thresholds should be:

- Collaboratively defined with stakeholders to promote buy-in and realism
- Context-dependent. Thresholds that apply to a safety-critical system differ from those for a low-risk internal tool.
- Reviewable and adjustable. Thresholds should be revisited periodically as the product and organization evolve.
- Documented and communicated, so that all stakeholders understand the criteria for decision-making and the consequences of exceeding thresholds

By operationalizing metrics through clear thresholds, test managers transform data into actionable intelligence, enabling timely, well-founded decisions that balance quality, risk, and business value.

Quantitative benefit analysis. Quantitative benefits include cost avoidance (e.g., reduced production failures), efficiency gains (e.g., test automation ROI), time-to-market acceleration, and risk mitigation (e.g., quantified through security testing, compliance validation, and operational stability enhancement using established organizational risk assessment and classification criteria). Cost avoidance is demonstrated through defect prevention metrics, where the savings are realized by detecting and resolving defects before release, thereby preventing production failures (i.e., failures caused by defects that escaped into the production environment), customer support costs, and reputation damage. Efficiency gains are evidenced by test automation implementation, process optimization, and test resource utilization enhancements that show a positive ROI for testing (see Section 7.1.2). Time-to-market acceleration is measured through early defect detection, continuous integration support, and increased release confidence.

Qualitative benefit assessment. Beyond the customer-satisfaction contribution discussed above, qualitative benefits include improved system reliability and faster defect resolution, as well as regulatory compliance and audit readiness. The latter protect the internal legitimacy and effectiveness of the test organization and safeguard operational continuity.

© International Software Testing Qualifications Board

Stakeholder-specific communication strategies. Communication should be tailored to each stakeholder group:

- Executive communication emphasizes business impact, competitive positioning, and strategic risk management that align with organizational priorities and financial objectives.
- Communication with technical stakeholders focuses on process efficiency, test tool effectiveness, and advancement in quality methodology and test practices, and adherence to budgets.
- Communication with project managers emphasizes how test strategies improve test schedule predictability, test scope clarity, delivery confidence, and responsiveness, including speed of fixes, updates, and enhancements.
- Customer-facing communications emphasize QC, reliability guarantees, and user experience enhancement.
- Communication with AI strategists and data science teams focuses on model performance metrics, data quality and bias considerations, AI security and compliance risks, cost-benefit trade-offs of AI tooling, and the implications of technical debt introduced by AI components. It also addresses the limitations of test capacities when dealing with non-deterministic AI outputs and the need for new quality approaches such as model validation and continuous monitoring.

Benefit presentation techniques. Business cases are developed to connect test activities to business outcomes through real-world examples, case studies, and comparative analysis. Visual representations including dashboards, infographics, and trend analyses are created to support data-driven decision-making. Regular test reporting cycles provide consistent value demonstration while building stakeholders' confidence and understanding. Baseline measurements enable before-and-after comparisons that demonstrate test impact and continuous improvement.

Value proposition development. Test managers create comprehensive business cases to substantiate and prioritize test investments. These include cost-benefit analysis considering the CoQ (i.e., defect prevention, appraisal, internal failure, and external failure costs), risk assessment with quantification of potential failure costs, and competitive comparison that support test resource allocation decisions. Scenario modeling is used to demonstrate test value under different project conditions (e.g., higher coverage versus reduced time-to-market and avoided revenue loss through early defect detection) and organizational contexts.

Value communication also leverages predictive analytics, machine learning insights, and real-time test monitoring capabilities to increase credibility and support evidence-based stakeholder engagement in data-driven organizational cultures.

Business-quality alignment frame. To make quality measurable as a business enabler, test managers use lightweight alignment instruments that connect quality to business objectives and create a common basis for dialogue:

- **Quality value mapping / risk-based test alignment canvas:** A one-page canvas that maps business risks, quality characteristics, test focus, and measurement criteria; it is maintained collaboratively in workshops and serves as a living compass for a risk-based test strategy.
- **Business impact scoring for quality metrics:** test metrics are weighted by their business impact, distinguishing leading from lagging indicators, so that the most consequential quality signals receive the most attention.

These frameworks promote dialogue-based coordination and make quality a measurable business enabler.

© International Software Testing Qualifications Board

3.2 Placement of the Test Team

Test team placement within organizational structures significantly impacts effectiveness, independence of testing, and test results. Organizations adopt diverse structures requiring strategic positioning decisions that balance quality focus, information access, and stakeholder collaboration while maintaining independence of testing and professional credibility.

3.2.1 Organizational Placement Selection

To select the most effective placement for a test team, a systematic evaluation must be conducted. This process involves thoroughly analyzing organizational factors, project requirements, and quality objectives. Test team placement significantly affects independence of testing, information access, collaboration, and effectiveness. The goal is to determine the optimal test reporting structure and operational position for the test team.

Organizational structure analysis. Existing hierarchies, (e.g., functional departments, matrix structures, and cross-functional teams), are evaluated to identify placement options, considering reporting relationships, decision-making authority, and resource allocation as they affect test team autonomy. Organizational culture (e.g., quality priorities, risk tolerance, and collaborative practices) and communication patterns are assessed to determine the positioning that best supports information access, influence, and acceptance.

Quality-focused placement options. These range from dedicated QC organizations and centers of excellence (CoEs) to embedded test roles in Agile software development/DevOps, hybrid models combining independent oversight with embedded collaboration, and, increasingly, AI-based analysis of organizational structure to recommend placement while preserving independence of testing. Each model trades off independence of testing, business context, and delivery speed differently.

From an organizational perspective, placement options fall into two categories: the organizational model that defines how the test team is structured, and the reporting host that defines where it reports. The two are independent axes, since a test team hosted in, say, the IT department may still follow a centralized or embedded model. The following table compares the primary options in both categories, their characteristics, advantages, disadvantages, and suitable contexts.

Placement option	Description	Key advantages	Key disadvantages	Best suited for
Organizational models (i.e., how the test team is structured)				
Centralized (e.g., an independent QC organization reporting to a quality executive)	A separate QC organization where all testers report independently through a quality-focused chain, preserving objectivity from development pressure	• Strong independence of testing • Consistent standards and tooling • Specialized career paths	• Organizational silos • Limited business context • Slower feedback cycles	Organizations requiring high independence of testing and regulatory compliance (e.g., aerospace, medical, and finance)
Embedded (decentralized)	Testers are permanently assigned to individual	• Close collaboration	• Reduced independence of testing under	Agile software development/DevOps with cross-functional

© International Software Testing Qualifications Board

Placement option	Description	Key advantages	Key disadvantages	Best suited for
	development teams (e.g., Agile software development/DevOps), fully integrated into the delivery process.	and fast feedback • Strong business / domain context • Shared ownership of quality within the test team	delivery pressure • Inconsistent test practices across test teams • Harder to sustain specialist skills	teams and rapid delivery cycles
Hybrid (matrix)	Testers are embedded in development teams but report functionally to a central QC organization (i.e., dual operational / professional reporting lines).	• Balances independence of testing with collaboration • Combines business context with professional standards • Supports career development	• Role confusion from dual reporting • Conflicting quality-vs-delivery priorities • Complex accountability	Large organizations with diverse portfolios; Agile software development transformations balancing speed and quality
Federated CoE	A CoE sets standards, tools, and guidance for testers who work embedded in delivery teams but belong to the CoE professionally.	• Standardization and knowledge sharing across teams • Maintains expertise and career paths • Scales test resources and reporting	• CoE can drift from project reality • Overhead and bureaucracy • Limited direct control over test execution	Large or distributed organizations needing consistency and specialized expertise across many teams
Reporting hosts (i.e., where the test team reports)				
IT department	The test team reports within the IT/technology department, a reporting host. The test team may still be centralized or embedded.	• Strong technical alignment • Shared infrastructure and tooling budgets	• Isolation from business stakeholders • Limited user / business understanding • Testing subordinated to	Technology- and infrastructure-driven organizations (e.g., internal IT, finance, and insurance) with lower business domain risk

© International Software Testing Qualifications Board

Placement option	Description	Key advantages	Key disadvantages	Best suited for
		• Proximity to development and operations	IT delivery goals	
Project Management Office (PMO)	The test team is positioned within the PMO, reporting through project-governance structures across the portfolio.	• Cross-project visibility and standardization • Consistent governance and reporting • Alignment of testing with project milestones	• Limited technical depth • Bureaucratic overhead • Testing is seen as governance rather than engineering	Large, portfolio-driven organizations needing standardized cross-project oversight and progress reporting

Table 2. Test Team Placement Options

Information access optimization. Placement selection shapes access to requirements, design specifications, and architectural decisions that are critical for test planning and execution. Proximity to business stakeholders, subject matter experts, and end users supports domain-knowledge acquisition and validation, and enables participation in planning meetings, design reviews, and quality-relevant decisions through both formal and informal communication channels.

Skill-based positioning considerations. Test team placement aligns with the required competency profile which includes technical expertise, domain knowledge, and test specializations such as test automation, security, or performance efficiency. The TME also weights the training, career-progression, knowledge-transfer, and mentoring opportunities that different placement options offer.

Cultural and enabling aspects. The organizational placement is crucial for supportive conditions. Direct access to decision-makers and product owners without hierarchical barriers fosters psychological safety, allowing the test team to raise risks and quality concerns openly and enabling earlier defect detection and a genuine culture of trust. The test team positions itself not as a gatekeeper but as a service-oriented provider of quality. In close cooperation with development, product, and business units, it establishes quality SLAs, (e.g., response times to critical defects, review cycles for test strategies, test environment availability, or quality levels per release), that define mutual expectations and make quality a shared responsibility.

Placement strategies incorporate remote work capabilities, distributed test team coordination, and virtual collaboration tools that enable flexible organizational structures while maintaining test effectiveness and quality focus.

© International Software Testing Qualifications Board

3.2.2 Placement Impact Analysis and Assessment

A comprehensive placement assessment requires a thorough analysis of multiple organizational factors, stakeholder relationships, and test results. To carry out this assessment consistently rather than ad hoc, the TME follows a repeatable, weighted method:

- Identify the candidate placement options for the given context (e.g., centralized, embedded, hybrid, and federated/CoE).
- Define the assessment factors and their weights with the stakeholders, (e.g., independence of testing/objectivity, information access, test resources and support, organizational culture, and change adaptability), weighting them by the organization's priorities (e.g., regulated domains weight independence of testing higher).
- Gather evidence per factor using the techniques already described (e.g., analyze reporting relationships, budget/resource processes, information access, cultural fit, and flexibility).
- Score each option against each weighted factor in a comparison matrix and compute a weighted result per option.
- Analyze the trade-offs across test effectiveness, stakeholder collaboration, and quality maturity, identifying conflicts (e.g., independence of testing versus collaboration).
- Recommend the highest-scoring option and document the rationale and residual risks.

This analysis evaluates the effectiveness of a test team's current position and identifies optimization opportunities. For each weighted factor, the TME assesses:

- **Independence of testing and objectivity assessment.** Reporting relationships and pressure points (e.g., schedule/budget) that could bias testing, and the team's decision-making freedom over test scope, test approach, and exit criteria.
- **Information flow and communication analysis.** Availability of formal documentation and informal channels, participation in decisions, and the quality of stakeholder relationships and conflict resolution.
- **Test resource and support evaluation.** Budget, tooling, training, and career-progression and retention factors across placement options.
- **Change management and adaptability assessment.** Flexibility for changing requirements and restructuring, transition costs, and scalability across teams and locations including Agile software development transformations and hybrid work environments.
- **Test results and maturity:** quality-delivery effectiveness (e.g., EDR, coverage, and delivery adherence) and long-term quality-maturity development.

Organizational quality maturity refers to staged test process maturity models, in particular the Test Maturity Model integration (TMMi), whose five levels (i.e., Initial, Managed, Defined, Measured, Optimization) provide a recognized baseline for benchmarking placement and improvement decisions. TMMi is test-specific and complements the more general Capability Maturity Model Integration.

© International Software Testing Qualifications Board

3.3 Building Strategic Relationships

Effective relationship management is one of the most important success factors for test managers in complex, regulated, and often distributed organizational environments. Strong relationships operate on three levels of increasing scope, internally within the test organization, externally with development, business, operations, and third-party providers, and across multiple independent organizations working as one. At every level they create the basis for shared learning, knowledge exchange, and a systemic quality culture, enabling early information flow, collaborative problem-solving, and shared responsibility for quality rather than isolated power dynamics. Building these relationships begins with a systematic understanding of the organizational landscape, including stakeholder influence and interest. The following addresses each level in turn.

In distributed environments, geographical distance, time-zone differences, and the specifics of virtual collaboration make relationship-building harder. Cultural backgrounds, individual communication preferences, and professional motivations also shape relationships and must be analyzed carefully to develop targeted strategies for durable partnerships with managers, development teams, and other stakeholders, fostering a culture where everyone shares responsibility for quality.

Two helpful tools support systematic analysis: a stakeholder matrix (i.e., influence and interest) and a network map to visualize informal connections, always with a focus on learning and collaboration potential.

The success of relationships can be measured using indicators such as:

- Number of joint improvement initiatives (e.g., shared retrospectives or process optimizations)
- Reduction in escalations through early, proactive coordination
- Increasing stakeholder proactivity in quality issues (e.g., stakeholder-initiated risk reports and unsolicited quality feedback)

These metrics reveal how relationships contribute to a learning, resilient quality organization.

Internal relationship management. Identifying critical relationships and their influence within one's own organization is essential for test success. The aim is to build a strong, cohesive, and learning-oriented test team while embedding quality as a shared responsibility. Test managers collaborate with:

- Development managers. Focused on collaborative quality planning, technical coordination, and defect resolution.
- Project managers. Focused on integrating testing into project planning, risk management, and delivery coordination
- Business stakeholders. Includes analysts, requirements engineers, and domain experts providing validation criteria
- Infrastructure teams. Responsible for test environment management, security, and operations

Informal networks complement formal structures, not for political maneuvering, but to allow knowledge to flow freely and break down silos. To systematically build and strengthen these internal relationships, targeted measures are required. These include delegation strategies that empower team members to independently establish contacts, as well as mentoring programs that help junior colleagues learn relationship skills through experience. The establishment of cross-functional and cross-country liaison roles (i.e., bridge-builders) significantly enhances collaboration across internal boundaries and fosters seamless connections between teams and locations.

© International Software Testing Qualifications Board

In addition, internal relationships are strengthened through pair testing and communities of practice (e.g., test guilds), in which test teams regularly exchange best practices, evaluate tools, and learn from each other. Knowledge is institutionalized to avoid dependence on individuals and to create resilience. This creates a vibrant internal network that promotes testing practices in Agile software development and a learning-oriented culture.

External relationship management. The strategic establishment and maintenance of relationships with external partners, (i.e., customers, suppliers, or authorities), is a central pillar for project success. These connections are consciously established and maintained as learning partnerships. The focus is on mutual knowledge exchange and the continuous, joint improvement of quality and processes that extend beyond individual project delivery. This process begins with a targeted analysis: Which key individuals on the partner side hold influence and can facilitate access to broad expertise, experience, and perspectives? On this basis, relationship strategies are tailored individually; they are consistently geared towards the specific needs, preferences, and cultural context of each individual external stakeholder.

The implementation of this strategy requires both structured and flexible elements. On one hand, structured measures, such as regular alignment meetings, clearly defined communication protocols, and joint planning formats provide the relationship with a reliable framework. On the other hand, flexibility and agility are equally important. Communication should be adapted as closely as possible to the individual person and situation. At the same time, clear, pre-agreed escalation and conflict resolution procedures help enable all parties to remain professional and constructive, even during challenging project phases. Joint retrospectives with customers and partners, where lessons learned are openly shared and concrete quality improvements are prioritized together, prove particularly effective. This creates stable, trust-based partnerships that drive long-term learning cycles and sustainably strengthen quality.

A particular caution applies when testers engage directly with customers and users. Building rapport through open dialogue is valuable, but it must not blur into the appearance of agreeing to changes or new features. A user who raises a seemingly small change can easily interpret a friendly, receptive conversation as a commitment, when in reality such requests can only be captured for consideration, (e.g., added to a backlog or raised with the product owner), and remain subject to separate prioritization, change control, and approval. Testers should therefore listen and record such input openly while making clear, without harming the relationship, that they are not authorized to commit to scope changes and that the request will be routed through the appropriate channel.

With contracted suppliers and vendors, however, communication is often governed by the contract rather than by goodwill. Contractual terms may restrict or prohibit direct communication, make it billable on a charge-for-time basis, or require that all interaction be channeled through a single point of contact such as the project manager or contract owner. Test managers must therefore understand the governing contract before assuming open communication is possible and adapt their relationship strategy accordingly. The detailed management of supplier and vendor relationships is addressed in Chapter 4.

Cross-organizational relationship management. The highest level of complexity in relationship management is cross-organizational collaboration. This is no longer about the bilateral management of a single partnership, but rather the ability to operate within a network or alliance of multiple independent organizations. The key here is the conscious use of informal connections, not as leverage for individual actors, but as a channel for open knowledge exchange, joint learning and collective quality development.

This type of cooperation requires highly integrated and agile work methods. Shared practices, such as test-driven development or pair testing between teams from different organizations, foster a culture of shared responsibility and collective ownership of quality. To enable this close collaboration in often distributed and global settings, relationship management is essential. It uses digital collaboration platforms, virtual networking strategies (e.g., virtual workshops), and cultural competency frameworks (i.e., clearly agreed-upon communication and collaboration rules) to develop effective partnerships across diverse corporate boundaries.

© International Software Testing Qualifications Board

The successful orchestration of such networks lies in creating a robust shared infrastructure. Test managers should deliberately establish communication agreements and configure tool environments to seamlessly support both synchronous and asynchronous collaboration.

Only through this systematic quality enablement of information and learning flows can a loose alliance become a genuine, high-quality cross-organizational team, one whose impact is significantly greater than the sum of its parts and which drives sustainable shared learning and systemic quality improvement.

3.4 Stakeholder Communication

Effective communication with stakeholders is a transparent, dialogue-based coordination process that promotes two-way exchange. It encourages everyone involved, from the team to senior management, to develop a common understanding of quality, progress, and risks. Instead of one-sided reporting, the focus is on continuous dialogue to anchor quality as a collective responsibility and enable well-founded, jointly supported decisions.

Test managers establish diverse channels tailored to stakeholders: real-time dashboards and collaborative platforms for everyday transparency, supplemented by personal interactions, workshops, and retrospectives for in-depth discussions and clarifications. The choice of method, timing, and content is based on information needs, decision-making cycles, and roles of those involved, always with the aim of avoiding information overload and creating real added value.

Key elements of a dialogical approach include:

- **Jointly developed quality agreements.** In cross-functional workshops, stakeholders and the test team jointly define what good quality means, including agreed upon quality characteristics, acceptance criteria, and consequences in the event of deviations. These agreements serve as a living reference framework and are regularly reviewed and adjusted.
- **Visualized quality targets.** Quality scorecards or quality impact dashboards show the current quality status in real time within the context of business objectives (e.g., risk reduction versus time-to-market, or customer satisfaction versus EDR). These scorecards are interactive, filtered by role, and explicitly invite discussion and prioritization, rather than simply reporting progress.
- **Institutionalized feedback loops.** Regular quality alignment workshops (e.g., monthly or accompanying releases) replace one-sided written reports with moderated, interactive sessions. Here, test results are reflected upon together, risks are discussed, lessons learned are shared, and next steps are agreed upon collectively. Supplemented by anonymous feedback channels and retrospectives, this creates a genuine learning loop.

Communication content focuses on actionable information that directly supports stakeholder responsibilities and decision-making requirements. Metrics and key performance indicators (KPIs) are presented in ways that explicitly link them to business objectives, enabling stakeholders to clearly see how test results support organizational goals and investment value.

Data visualization, interactive formats, and increasingly AI-supported personalization (e.g., automated summaries and optimal channel/timing suggestions) are making communication more accessible and inclusive, especially in distributed teams.

© International Software Testing Qualifications Board

In their communication, test managers also make transparent whether they are monitoring (i.e., observing progress – "Where are we?") or controlling (i.e., taking corrective action – "What do we do about it?"); the two are covered in detail in Chapter 6.

The measure of success is not only the delivery of information, but also actual understanding, the quality of dialogue, and the resulting joint actions.

3.5 Integrating Tools Across the Organization

Shared tools that cross organizational boundaries are more than just technical infrastructure: they shape how testing, development, business, and operations collaborate and how quality knowledge is preserved. This section addresses two responsibilities of the test manager; the factors to weigh when such multi-use tools are selected and operated, and the systematic assessment of any proposed change to them.

3.5.1 Factors to Consider When Dealing with Multi-Use Tools

When integrating tools that serve multiple purposes and user groups across the organization, test managers must consider several key factors to promote successful adoption and sustainable value creation.

Stakeholder needs and cross-functional alignment:

- Early involvement of all user groups (i.e., testing, development, business, operations) in tool selection and configuration
- Establishment of common standards for data quality, nomenclature, and access rights
- Regular review of tool effectiveness through cross-functional feedback mechanisms

Integration and infrastructure:

- Seamless integration into the entire delivery pipeline (e.g., monitoring)
- End-to-end traceability across the SDLC by linking requirements, version control and releases, test management, defect management, and deployment, so that any item can be followed from requirement to deployment. Integrated ALM platforms are the typical means of providing full SDLC coverage.
- Flexibility to support cloud, microservices, and hybrid architectures
- Scalability to accommodate organizational growth and evolving project needs

Governance and ownership:

- Clear, collaborative models for decision-making, funding, and change management
- Clear ownership of the data held in shared tools; for externally provided tools this is governed contractually (see Chapter 4)
- Defined roles and responsibilities for tool administration and maintenance
- Transparent upgrade and retirement processes to minimize disruption

Learning and knowledge quality enablement:

- Tools should be designed to document quality knowledge and make it accessible.
- Support for retrospective findings, lessons learned, and continuous improvement
- Automated linking of defects to improvement initiatives to keep knowledge reusable

© International Software Testing Qualifications Board

Support for open communication:

- Tools should provide anonymous feedback mechanisms so that risks and improvement ideas can be raised candidly.
- Integrated retrospectives and feedback functionalities that reinforce a learning culture and keep input visible to all groups

Training and adoption support:

- Adequate training resources and onboarding materials for all user groups
- Ongoing support and communities of practice to share best practices
- Change management to address resistance and provide smooth transitions

Once such shared tools are in place, any proposed change to them must be assessed systematically rather than ad hoc, because a single change can ripple across every connected user group.

3.5.2 Impact Analysis of Proposed Changes to Multi-Use Tools

When a change to a multi-use tool is proposed, test managers must systematically analyze the potential impact on the test organization across multiple dimensions.

Technical impact assessment:

- Evaluate compatibility with existing test automation frameworks and CI/CD pipelines.
- Assess integration effort with other tools and systems.
- Identify potential disruptions to ongoing test activities.

Operational impact assessment:

- Determine the impact on test execution, reporting, and defect management processes.
- Assess the effect on team productivity and workflow efficiency.
- Evaluate changes to data quality, traceability, and auditability.

Organizational impact assessment:

- Identify affected stakeholder groups and their specific needs.
- Assess impact on cross-functional collaboration and information flow.
- Evaluate alignment with organizational quality objectives and standards.

Risk and mitigation analysis:

- Identify risks associated with the proposed change (e.g., data loss, downtime, and skill gaps).
- Develop mitigation strategies and contingency plans. Targeted mitigation strategies against loss of defect management and visibility include enforcing a single source of truth, monitoring tracking health, providing traceability across tool changes, maintaining redundancy during transitions, using tool-independent dashboards, defining clear governance and ownership, and running periodic audits.
- Plan phased rollout and rollback procedures.

© International Software Testing Qualifications Board

Cost-benefit evaluation:

- Quantify implementation and ongoing operational costs.
- Assess expected benefits (e.g., efficiency gains and quality improvements).
- Compare with alternative solutions or maintaining current state.

Communication and change management:

- Develop a communication plan for all affected stakeholders.
- Plan training and support during transition.
- Establish feedback channels to monitor adoption and address issues.

From defect data to process improvement – **a repeatable method.** Building on the integrated tool data described above, defect data only becomes a driver of process capability when it is processed through a defined, repeatable method rather than ad hoc. The following steps operationalize this:

- **Capture and classify.** Capture defects with consistent attributes (e.g., introduction phase, detection phase, severity, root cause category, and affected component). A common taxonomy (e.g., orthogonal defect classification) makes the data analyzable.
- **Aggregate and detect patterns.** Aggregate by category and phase; calculate phase containment or DDP and EDR; identify recurring defect classes and where defects are introduced versus discovered.
- **Root-cause analysis.** Investigate the dominant clusters in blame-free retrospectives using structured techniques (e.g., 5 Whys method and RCA) and separate individual cases from systemic process causes.
- **Derive process actions.** Translate the root causes into concrete process changes (e.g., earlier reviews, additional test types, requirement clarifications, quality gates, and test automation).
- **Prioritize the actions.** Rank the candidate process changes by expected impact against required effort, so that high-impact / low-effort improvements (i.e., quick wins) are addressed first and high-effort / low-impact ones are deferred or dropped. An impact-versus-effort matrix makes this trade-off explicit and keeps improvement focused where it pays off most.
- **Set a measurable target.** Define a target value with stakeholders (e.g., reduce defect class X by Y%, and exceed the threshold DDP in phase Z).
- **Implement and remeasure.** Implement the change and measure again in the next cycle; track time-to-learn, learning rate, and reduction in recurring defect classes.
- **Institutionalize.** Incorporate validated improvements into standards/playbooks and review the cycle periodically.

For example, a team observing a recurring cluster of integration defects (i.e., capture and classify) runs a Pareto analysis that traces 60% of them to one interface (i.e., aggregate and detect patterns). A blame-free 5 Whys method reveals an ambiguous interface contract as the root cause (RCA); the team introduces a contract-review gate (i.e., derive process actions), targets a 50% reduction over two releases (i.e., set a measurable target), and confirms the drop in the next cycle (i.e., implement and remeasure) before standardizing the gate (Institutionalize).

Measuring success beyond metrics. Across all five areas of this chapter, the success of the test team is judged not by standard product metrics (e.g., DDPs) alone, but by learning- and culture-oriented indicators that highlight systemic improvement and sustainability. These can be categorized as process metrics (e.g., learning rate and feedback loop efficiency) and cultural indicators (e.g., psychological safety score). Important supplementary metrics include:

© International Software Testing Qualifications Board

Learning progress:

- Time-to-learn (i.e., the duration from defect detection to process/product improvement)
- Learning rate from retrospectives (proportion of measures implemented)
- Fail-fast rate with subsequent learning rate

System improvements:

- The number of cross-functional improvement initiatives implemented per quarter
- Retention rate and career progression metrics (e.g., promotions, skill certifications achieved, and training hours completed)
- Reduction in recurring defect classes
- Shortening feedback loops (e.g., from release to release)

Cultural indicators:

- Psychological safety score (e.g., anonymous team surveys)
- Stakeholder proactivity (e.g., independent risk reports)
- The number of constructive suggestions for improvement per person/quarter

© International Software Testing Qualifications Board

4 Managing External Relationships – 330 minutes

Keywords

entry criteria, exit criteria, test approach, test strategy

Domain Specific Keywords

metric, service level agreement, third-party

Learning Objectives for Chapter 4:

4.1 Types of External Relationships

LO 4.1.1 (K2) Give examples of third-party relationships regarding test responsibilities

4.2 Contractual Issues

LO 4.2.1 (K3) Prepare a suitable service level agreement for a given project with an external vendor

4.3 Communication Strategies for Third-Parties

LO 4.3.1 (K4) Select an appropriate communication strategy for a given third-party engagement

4.4 Integrating from External Sources

LO 4.4.1 (K4) Analyze a given project to determine the appropriate level of internal test involvement in combination with a third party

4.5 Merging Third-Party Test Strategies

LO 4.5.1 (K2) Summarize the items that must be considered when merging test strategies with a third-party organization

4.6 Verifying Third-Party Quality

LO 4.6.1 (K5) Assess the quality of an external relationship

© International Software Testing Qualifications Board

Introduction to Managing External Relationships

Introduction

It is common for a TME to encounter various project situations where there is a third-party vendor involved in developing, testing, and/or delivering parts of a system or the whole solution. The TME must be able to identify when these situations occur and understand the level of quality expected of the various deliverables. A library or component needs different depths of testing with respect to a subsystem or a complete solution delivered. The more the system is delivered as a complete solution, the more it needs to be tested in the entire functionality due to a high complexity with a huge number of test conditions to be covered.

In addition, considerations must be made regarding the test schedule, test approach, testware, and communication required to facilitate the smooth delivery of a high-quality system. This external relationship may be with a vendor (e.g., someone who is paid to provide the designated services) or an external group within the same organization (e.g., testing performed by a group in a different part of the organization). There may be more than one of these entities, and they can be combined internally and externally to the organization, which augments the complexity of synchronization and management between them. These entities will be referred to jointly as “third party” (Black 2009).

In both types of relationships, it is extremely important to define clear contracts, establish clear communication channels, define stakeholders, set clear expectations, agree on test reporting, documentation of executed activities, and/or tasks. Promoting mutual trust, planning regular meetings, understanding how to achieve project goals, and mitigating risks are vital for the success of external relationships.

4.1 Types of External Relationships

Introduction

There are several types of external relationships with third parties:

- Deliver a completed turnkey system
- Develop a system, but the testing is the responsibility of the recipient organization
- Develop part of the code that must be integrated with the recipient’s code
- Be responsible for all the testing of an internally developed system
- Share responsibility for the testing of an internally developed system with the internal test team
- Provide overview of activities, while not actively participating in either development or testing
- Consult and provide guidelines for any process definition or improvements
- ... other, or any combination of the types mentioned

When dealing with multiple third parties working on a single project, communication between all implied parties may also need to be actively managed. Each of these relationships may have different requirements for communication, documentation, and coordination. The TME must be aware of these differences as well as any contractual constraints that may be present.

© International Software Testing Qualifications Board

4.2 Contractual Issues

Contracts are necessary when working with external test resources. Although contract administration is usually handled through the legal department for external parties, the TME must be aware of the contractual issues that may affect testing and relations and provide technical and methodological input as the expert of test services.

Specific contract items that may influence testing include:

1. **SLA.** Response time to questions, new functionality requests, releases, defect fixing, turnaround times, availability of support staff and method of contacting them, entry criteria, exit criteria (e.g., in Agile software development, it is often called definition of ready and definition of done)
2. **Deliverables.** Testware (e.g., specifications, user guides, and help files), release notes, known defects lists, test evidence, notification of areas not ready for testing, test data used for internal validation
3. **Quality level.** Agreements about QC (e.g., reviews, testing coverage, and acceptable number of residual defects by severity and priority)
4. **Use of tools.** Agreement on tools for communication including ownership of data maintained in tools. Defect management and test management tools can help facilitate effective and consistent communication between multiple parties.

Additional requirements may apply depending on the contract, the industry, the system, and contractual relationships. For example, projects involving medical devices or military systems require government approval.

TME may be involved in the third-party selection process. However, it is important for the TME to actively define and govern the contractual definition to take the best approach to the external partner and avoid misunderstandings between them.

The TME should have significant input into the content of the contract, particularly any items related to test strategy. The solution that is to be delivered under the contract must be accurately described to allow for validation of the finished system. The TME should participate in pre-contract reviews and confirm that QA and test clauses are included in supplier quality agreements.

4.3 Communication Strategies for Third Parties

Contractual governance must be translated into an operational communication plan. A communication strategy with third parties is based on strategies and interfaces defined in the formal contract, which describes methods to convey information, expectations, and messages to external stakeholders.

These strategies are defined so that communication is clear, consistent, aligned with organizational goals, and tailored to the needs and roles of the third parties involved.

Communication with a third party may vary in the amount, frequency, details, style, format, formality and other things, depending on, for example, the system, locations involved, cultural differences, and the criticality and complexity of the project.

The TME must be aware of each of these factors before defining and enabling a communication strategy for the project. If the project employs development and component testing in an offshore facility, it might make sense to require weekly detailed test progress reports, daily defect reports, semi-weekly conference calls and even an agreement on the language communication to be used.

© International Software Testing Qualifications Board

Using standard tools for communication (see Section 4.2), accessible by all teams' members, and common defect management and test management tools can help facilitate effective and consistent communication between multiple parties. This means that appropriate access and responsiveness must be available with these tools, and 24/7 support of the tools may be required as well. It may also be necessary to publish usage guidelines, templates (e.g., how to describe a new requirement), meeting minutes, and severity/priority schemes, to all project members to avoid misunderstandings. This is particularly important when the third party is doing the development, while a local test team is performing the testing.

External communication extends beyond development and test teams. It must also include:

- Suppliers (e.g., third-party vendors)
- Regulatory bodies (e.g., compliance with organization, national, or global standards)
- Partners (e.g., consultants and outsourced firms)
- Customers or end-users (e.g., feedback on final product including production environment).
- Community (e.g., groups)

While the TME may contribute to these communications, there may be monitors and controls in place for them. The TME must understand who is authorized to communicate with third parties and must understand their role in initiating, contributing, or reviewing these communications.

The test manager or the TME can conduct risk assessment workshops which are an effective way to involve these stakeholders.

Meeting schedules should be established at project initiation and accommodate all time zones. Issues to consider while planning include::

- Communication language (e.g., English for formal meetings and local language for informal sessions)
- Time zones: avoid scheduling meetings at unreasonable hours for any participant
- Physical locations, (e.g., factors in travel or remote participation needs)
- Vacation periods (e.g., school vacation in Europe versus Asia)
- Public holidays: account for regional differences
- Cultural differences: to support the correct level of representation

A summary of specific communication styles is provided in the appendix of (Evans 2004).

© International Software Testing Qualifications Board

4.4 Integrating from External Sources

The TME must define the team's level of involvement in testing the final system. This should be documented in the contract as a specific clause. The degree of involvement depends on the type of deliverable provided by the third party:

- **Developed and tested system (turnkey).** All testing should be completed with this system. The test team should plan to perform acceptance testing per acceptance criteria (e.g., verification of contractual requirements and specifications). System integration testing may be needed if this system is being integrated with others prior to release as a complete package.
- **Developed and tested only at the component level.** Coverage metrics should be reviewed before the system is accepted into component integration/system testing. The test team should expect to perform component integration/system testing and any acceptance testing that may be required. This is the same scenario that occurs when an external team develops part of the code that is subsequently integrated with internally developed code.
- **Developed but untested at the component level.** Ideally, the test team would have the technical capabilities to perform component testing, but component testing is usually better performed by the development team. In some cases, it may be more expedient to skip component testing and proceed directly with component integration/system testing. Tracking the CoQ metrics is particularly important in this case to determine the cost to the project of bypassing component testing (see Chapter 7).

A third party performs all the testing for an **internally developed system**. This is a test management issue, and the TME must either have defined or agreed to the defined test strategy that will be used to perform this testing. This includes understanding the test approach, the testware, the test tools and any test automation that should be developed and delivered along with the tested system. Communication needs in this type of project may differ from one where only the development is outsourced due to the need to actively manage the test effort as it progresses.

Complex projects may require a more extensive level of integration and handoff testware. It is important for the TME to understand any requirements that are particular to their domain and to confirm that there are handoff checkpoints (i.e., milestones) that will validate the completion of all acceptance criteria.

A contract may be the only enforceable acceptance criteria for the project. As noted above, if the contractual specifications are insufficient to determine an acceptable level of quality, the TME should track metrics that can be used for future agreements to raise quality levels to meet the internal needs of the organization. The TME should have a major influence in determining and enforcing the quality standards for a system. If this authority is not clearly defined in the contract, it is imperative that the TME determines that the necessary metrics are gathered. The TME should use collected metrics to build evidence-based business cases for updating contractual QA clauses and enforcing minimum quality standards in subsequent supplier agreements. Metrics such as DDP, defect trends or component testing coverage will be helpful in setting this requirement.

© International Software Testing Qualifications Board

4.5 Merging Third-Party Test Strategies

One of the most difficult projects for a TME to manage is one in which some parts of the testing will be performed by a third party. This may be across multiple locations, multiple organizations, and/or multiple internal/external user groups. Unless clearly specified in the contract (see Section 4.2), every supplier can potentially use their own local development and test environment, bringing extra complexity for the TME. There are many levels of complication in these scenarios. These include:

- Merging disparate tools (e.g., defect management tools and test management tools)
- Defining ownership of and access rights to the data
- Defining and unifying test automation strategy/tools
- Defining test levels and responsibilities for those test levels
- Defining shared test objectives and quality goals
- Understanding the various test strategies that may be employed by the various members of the project
- Defining acceptable entry criteria and exit criteria
- Defining quality test data (e.g., the need for synthetic data generation or data masking strategies, especially regarding data privacy regulations such as the General Data Protection Regulation)
- Maintaining quality test data
- Defining a common glossary of terminology, including testing and domain-specific terms if not clearly defined, (e.g., use of ISTQB® Glossary terms)
- Defining common metrics and test reporting frequency that will be used to control the project
- Addressing supply chain integrity and possible security/confidentiality issues

  Addressing security and confidentiality issues involves implementing robust measures such as multi-factor authentication, and access control to protect sensitive data from unauthorized access. Clear policies and secure communication protocols are crucial for maintaining privacy. Continuous test monitoring, third-party risk management, and a strong security culture further safeguard against evolving threats.
- Identifying areas of responsibility and touch points between test teams. A lack of defined overlaps may lead to gaps whereas excessive overlaps lead to inefficiency.

Since these issues will become more critical as the project progresses, it is important to resolve the test strategy and test tool questions before the project starts. This requires planning and coordination which may not have been included in the schedule or project plan. Regardless, it is the work that must be done and is a time investment that will pay back over time. Another complication that sometimes occurs when trying to reach agreement on these items involves the availability of the test counterparts in other organizations. Since test resources may be assigned as the project progresses, planning and coordination of test resources may not be available when the test planning is done. It is important to confirm that test planning time is allocated in the project schedules and even in the contracts to provide efficient use of the test resources when they are available.

© International Software Testing Qualifications Board

When multiple organizations participate in the testing of a system, the test strategies must be aligned. This does not mean everyone has to use the same test strategy, test techniques, or test automation, but it does mean that the overall test results meet the needs as defined in the test strategy for the project. There is usually one owner of the test strategy and that is usually the TME for the organization that is procuring the services of the other organization(s). One way to help unify the test strategy is to provide that there is a risk assessment process that is used to evaluate the various risk factors, and that the prioritized risk factors are addressed in the overall test strategy. Unifying the understanding of the risks and agreeing on the risk mitigation plan helps to bring about a common implementation of the test strategy.

4.6 Verifying Third-Party Quality

One of the challenges for the TME is to confirm that deliverables match expectations and requirements. Predictable deliverables must be clearly defined at the project's start, preferably within the project documentation or contracts. Objective quality measurements for all incoming software must be defined and assessed before the software is accepted. There must also be a process for the TME to reject the incoming software if the quality criteria are not met. This can be accomplished by defining measurable entry criteria and exit criteria for each phase of the project. These criteria are sometimes aligned with project milestones, although milestones may be more schedule-oriented (i.e., scheduled milestones) than quality-oriented (i.e., quality gates).

The following are examples of entry and exit criteria that can be used to determine the acceptance of software into various test levels of the test process:

- Exit criteria from component testing, (e.g., statement coverage meets or exceeds 85%)
- Entry criteria into component integration testing, (e.g., static analysis of code is complete with no outstanding defects)
- Exit criteria from component integration testing, (e.g., all components of functional areas are integrated so that interfaces are verified to be working correctly)
- Entry criteria into system testing, (e.g., no outstanding blocking defects and all known defects documented)
- Exit criteria from system testing, (e.g., all performance efficiency requirements are met)
- Entry criteria into system integration testing, (e.g., no outstanding high priority/severity defects open)
- Exit criteria from system integration testing, (e.g., all identified system interfaces have been tested)
- Entry criteria into acceptance testing, (e.g., all planned testing by the test team(s) has been completed and the test results meet the specified acceptance criteria)
- Exit criteria from acceptance testing, (e.g., sign off by the accepting parties)

The criteria may also consider:

- Context-specific quality goals (e.g., risk level, maintainability, or compliance readiness)
- Collaborative readiness evaluation (e.g., shared reviews between vendor and client teams)

© International Software Testing Qualifications Board

The list of entry criteria and exit criteria for an actual project should be more extensive, and the stringency of the criteria should vary with the product domain, experience with the third party, and the requirements stated in the contract/agreement. It is important to remember that the criteria are measurable in an objective way and that they are agreed upon at the beginning of the project. To be useful, the criteria must be achievable. The TME must be sure of having the authority to accept or reject a work product before publishing criteria that might not be enforceable.

Milestones are important for project tracking and for determining if testing is on schedule. By defining clear entry criteria and exit criteria, milestones must be defined in such a way that successful completion of a milestone can be demonstrated. This often results in milestones being tied to the criteria examples defined above. Milestone tracking is often the responsibility of the project manager but should also be tracked by the TME when the milestones pertain to the test schedule or quality-related issues.

© International Software Testing Qualifications Board

5 Project Management Essentials – 420 minutes

Keywords

cost of quality, defect prediction, impact analysis, mutation analysis, project risk, risk analysis, test estimation

Domain Specific Keywords

change management, cost-benefit analysis, outlier, standard deviation, value-based prioritization

Learning Objectives for Chapter 5:

5.1 Test Estimation and Defect Prediction

ELTM-5.1.1 (K3) Use standard deviation to interpret test estimation

ELTM-5.1.2 (K4) Select a defect prediction model

5.2 Project Risk Management

ELTM-5.2.1 (K4) Analyze trade-offs between quality, test schedule, budget, and test scope of a test project

ELTM-5.2.2 (K3) Apply project risk analysis

5.3 Change Management

ELTM-5.3.1 (K2) Summarize the role of the TME in the change management process

ELTM-5.3.2 (K4) Illustrate impact analysis for project-wide changes on testing

5.4 Test Management Considerations for Project Types

ELTM-5.4.1 (K3) Apply a given test management approach to a specific project type

© International Software Testing Qualifications Board

Introduction to Project Management Essentials

Introduction

Effective test management is critical to the success of any software test effort, especially in complex, large-scale, or high-risk projects. This chapter focuses on several aspects of test management for TMEs, providing the skills to estimate test effort accurately, manage risks effectively, and navigate changes in dynamic project environments. Knowledge of these aspects allows TMEs to develop the competence to make data-informed decisions, balance competing project constraints, and align test activities with overall project goals.

5.1 Test Estimation and Defect Prediction

5.1.1 Uncertainty in Test Estimation

The most used techniques for test estimation are described in (ISTQB-FL) and (ISTQB-TM). They include metrics-based techniques (e.g., analogy-based estimation, estimation based on ratios, and extrapolation) and expert-based techniques (e.g., wideband delphi, planning poker, and three-point estimation). This syllabus focuses on advanced topics related to test estimation, including test measurement error and outlier identification.

Test estimation is the process of predicting the effort, time, and test resources required to complete test activities. Test effort is typically measured in time-based or size-based units to quantify the amount of work required in software testing (e.g., person-hours, person-days, calendar time, test points, function points, and user story points). Performing a work breakdown structure (WBS) before test estimation is a good practice because it lays a strong foundation for accurate, transparent, and manageable test estimation. WBS decomposes the process into smaller, more understandable tasks. Smaller components are easier to estimate than large, aggregated ones. Test estimation errors are reduced because test estimates are made at a granular level, allowing test team members to understand the test scope better.

Some test estimation techniques require multiple data points of the estimated value (e.g., values provided independently by multiple experts or historical data from multiple similar projects). The final estimate is then determined by a measure of central tendency, such as the mean, median, or mode. In such cases, the sample standard deviation can also be calculated. Assuming that $x_1, x_2, \ldots, x_n$ are the data points representing the estimated value, the sample standard deviation s is calculated as

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}}$$

where n is the number of data points and $\mu = \left(\frac{1}{n}\right)\sum_{i=1}^{n} x_i$ is the arithmetic mean of the values $x_1, \ldots, x_n$.

Standard deviation defines the amount of variation in test data and plays a crucial role in the following activities related to test estimation.

© International Software Testing Qualifications Board

Measuring the uncertainty associated with the test estimation. A small standard deviation (e.g., less than 10% of the mean) implies strong agreement among estimators, indicating low uncertainty, so estimates can be considered consistent and reliable. A moderate standard deviation (e.g., 10-30% of the mean) indicates moderate uncertainty, suggesting that assumptions should be carefully reviewed. A large standard deviation (e.g., more than 30% of the mean) suggests disagreement or ambiguity, indicating high uncertainty, which may render the test estimation unreliable. In such a case, some more information should be gathered, or the test estimation scope should be refined, (e.g., by using the WBS).

Providing test estimation errors. If the test estimates are independent, the central limit theorem implies relationships between the percentage of data items that fall within an interval around the mean value μ and the width of this interval, expressed in terms of s. For example, approximately 68% of all data items fall within the interval $\mu \pm s$, approximately 95% within $\mu \pm 2s$, and approximately 99.7% within $\mu \pm 3s$. This relation allows the TME to express the estimated value in probabilistic terms.

Identifying outliers. (i.e., data points that fall far away from the mean). If data is normally distributed, a data point x is defined as an outlier if $|x - \mu| > k \cdot s$. Typical k values used are 2 (mild) and 3 (extreme). If one test team member provides an estimate that is significantly different from the rest (i.e., an outlier), this helps prompt discussion: is it a misunderstanding or a hidden risk? Outliers can be discarded, but first, they should be investigated as to why they exist. If data is skewed, the interquartile range (IQR) should be used instead of the standard deviation. $IQR = Q3 - Q1$, where $Q3$ and $Q1$ are the third and the first quartile of the data. $Q1$ is the value exactly halfway from the start of the ordered data to the median. $Q3$ is the value exactly halfway from the median to the end of the ordered data. Outliers are then defined as values that fall below $Q1 - 1.5 \cdot IQR$ or above $Q3 + 1.5 \cdot IQR$.

5.1.2 Defect Prediction Models and Techniques

Defect prediction forecasts where defects are likely to occur in a software component or system. It identifies defect-prone components before testing begins. Predicting defects enables a more effective allocation of test effort. It transforms test estimation from a flat effort model into a risk-driven strategy, focusing on where critical defects are most likely to occur. Critical components predicted to be defect-prone get more coverage, AI-augmented testing, or manual exploratory testing effort, while non-critical components may get minimal testing. Defect prediction models can adjust test case volume, test automation or review test intensity, and test execution time depending on predicted defect counts. Predicted defect counts can also be a variable in regression or machine learning-based test effort prediction models.

The most used defect prediction models are binary classifications (i.e., defective versus non-defective component), multiclass classifications (e.g., low, medium, and high defect risk), and regression (predicting the number of defects per component).

Standard defect prediction models focus on files or components and are typically based on code metrics (e.g., size and complexity) and defect patterns. Just-in-time defect prediction models (Kamei 2013) evaluate each code change and predict if this specific change is likely to introduce a defect. Just-in-time models are typically based on change history (e.g., code churn, number of commits, change frequency, and volatility of changes) and test team characteristics (e.g., number of developers working on the file, and developers' experience).

© International Software Testing Qualifications Board

Statistical models rely on quantitative software metrics and use mathematical relationships to forecast defect proneness. This is useful when historical defect data is available and the relationship between variables and defects is known or follows a known distribution. Examples include:

- Logistic regression predicts a binary outcome (i.e., defective/non-defective components) based on continuous and categorical predictors.
- Linear regression predicts the number of defects as a continuous output.
- Poisson regression predicts defect counts assuming a Poisson distribution (see example (Web1)).

Machine learning models learn complex patterns and nonlinear relationships from data and outperform statistical models, especially when the data is large in scale, or not easily modelled statistically. Examples include decision trees, random forests, support vector machines, and neural networks. These models typically need a large volume of labeled training data. Hence, they work well in large or long-lived projects with version control and defect management systems. Machine learning models may use non-handcrafted metrics (e.g., tokenized code), which can be difficult or impossible to interpret.

Mutation analysis is based on fault seeding, a fault-based test technique, adapted as a defect prediction technique. It is useful in the absence of real data. A mutation is a small syntactic change made to the source code using a predefined mutation operator. Each mutant introduces a plausible defect that mimics a potential programming defect. A test suite kills a mutant if at least one of its tests yields different test results for this mutant compared to the original code. Mutation analysis assumes that the effectiveness of a test suite in detecting actual defects is equivalent to its effectiveness in killing mutants.

Let X be the (unknown) total number of real defects,

D – the number of detected defects,

M – the number of generated mutants,

K – the number of killed mutants.

We assume that $D / X = K / M$, which allows us to calculate $X = (D \cdot M) / K$, the total number of real defects.

The difference $X - D$ represents the estimated number of remaining defects in the code.

Mutation analysis is a heuristic estimate, and its accuracy depends on the “representative nature” of the mutation operators used.

The Rayleigh model (Kan, 2002) is a defect arrival model used to forecast the number of defects over time. It is useful when assumed that the defect discovery rate follows a Rayleigh distribution, (i.e., it quickly rises to a peak and then slowly tapers off. The maximum of defects is reached when approximately 40% of the total number of defects are detected.

© International Software Testing Qualifications Board

5.2 Project Risk Management

5.2.1 Trade-off Decision Making

Managing the test process requires complex decision-making that involves balancing the four key constraints of quality, test schedule, budget, and test scope of a test project. The TME is expected to understand how each of these dimensions interacts with the others and to make informed decisions when trade-offs are necessary, often under pressure or in conditions of uncertainty.

Each of the four constraints is related to different aspects of a test project:

- Quality is related to test results and coverage, risks, defects (e.g., number, density, and severity), quality characteristics (e.g., based on the ISO/IEC 25010 product quality model), conformance to requirements, and user satisfaction
- Test schedule is related to delivery timelines, test execution windows, and deadlines
- Budget is related to test resource allocation, test effort, cost of labor, test tools, and test environment
- Test scope is related to features under test, test levels, test types, and coverage type

These constraints are interdependent, and unavoidable trade-offs exist between them. For example, increasing quality often increases cost and time; reducing the schedule may compromise scope or quality; reducing the budget may necessitate a reduction in scope or quality; and so on.

Different tools, such as cost-benefit analysis, value-based prioritization, or the CoQ model, can support trade-off decisions made by the TME. These tools allow the TME to make rational, defensible trade-off decisions when managing the key test project constraints.

Cost-benefit analysis is a quantitative decision-making technique that compares the costs of an action or decision with the expected benefits it yields. Benefits and costs must be comparable; therefore, they are typically expressed in monetary terms. In testing, this typically involves evaluating the cost of performing specific test activities against the value they add. Examples of supporting trade-off decisions by cost-benefit analysis include:

- Justifying test scope reduction if testing a low level risk feature has high cost but low benefit
- Supporting a budget increase if the benefit of defect prevention outweighs the cost
- Deciding where to cut or expand testing when time is limited

For example, performing an inspection increases short-term costs and the test schedule, but may have a positive impact on quality due to early defect detection, which in turn may save costs of defect fixing due to a lower number of production defects.

© International Software Testing Qualifications Board

Value-based prioritization is a qualitative or semi-quantitative technique that prioritizes features, test cases, or test activities based on the value they deliver to stakeholders (e.g., business importance, customer impact, and regulatory need). Prioritization is based on factors such as business criticality, functionality, usage frequency, customer visibility, risk level, or legal and compliance requirements. Examples of supporting trade-off decisions by value-based prioritization include:

- Enabling intelligent reduction of test scope when under time or cost pressure, focusing on high-value areas
- Justifying why some defects are acceptable in low-value areas (i.e., a quality trade-off)
- Aligning test effort with business goals, avoiding unnecessary over-testing

For example, suppose an online retail system must go live before a holiday sale. In that case, value-based prioritization may lead to exhaustive testing of the checkout process and promotional engine, while deferring testing of the wish list or product reviews which are lower-value areas.

The CoQ model (Crosby 1978) helps the TME make strategic trade-off decisions by providing a financial perspective on quality efforts and their test results. CoQ is the total cost associated with good quality, as well as costs incurred due to poor quality. It breaks down into four categories:

- Defect prevention costs: investments in preventing defects, (e.g., training, process improvement, and early test design)
- Defect appraisal costs: costs of testing components or systems, (e.g., dynamic testing and reviews)
- Internal failure costs: costs from defects found before release, (e.g., rework, debugging, and confirmation testing)
- External failure costs: costs from defects found after release, (e.g., customer complaints, SLA penalties, and brand damage)

The key idea behind the CoQ model is that investing more in defect prevention and appraisal costs can reduce internal failure and external failure costs, but only up to a point. Over-investing may lead to diminishing returns. CoQ enables the TME to evaluate where money is best spent in the test process and how adjusting test effort affects the total cost of testing. CoQ supports long-term decision-making. For example, investing in prevention (e.g., improving the test process) may reduce future internal and external failure costs across multiple releases.

5.2.2 Project Risk Analysis

The ISTQB® Advanced Level Test Management syllabus focuses primarily on product risks. This syllabus addresses the broader domain of project risks and discusses more advanced topics on risk analysis. The TME contributes to integrating project risks into the overall project risk management and is expected to actively collaborate with project leadership to evaluate how these risks impact the test process.

© International Software Testing Qualifications Board

Project risks can originate from a wide range of sources, each with distinct implications for the test process. Examples include:

- Product risks, which can jeopardize coverage or delay readiness for testing
- Organizational risks, which can disrupt the effectiveness, efficiency, and predictability of test activities
- Process-related risks, which can have a risk impact on test quality and efficiency
- External risks, which can affect a test project in ways that are often unpredictable and beyond the direct control of the project team

Key methods for risk identification and risk assessment include expert judgment and brainstorming, historical data and lessons learned, risk checklists and taxonomies, stakeholders' interviews and surveys, risk workshops and risk storming, process and metric analysis, and RCA.

Risk analysis can be approached using qualitative, quantitative, or hybrid methods, each with its strengths and limitations.

Qualitative risk analysis assigns risk values on an ordinal scale. These values are often mapped to numbers (e.g., 1-5). While easy to implement and helpful for early phase or low-data environments, qualitative analysis can lack precision and may lead to misinterpretation when ordinal values are mathematically manipulated. For example, multiplying ordinal values is mathematically incorrect, as ordinal values do not represent measurable quantities, and multiplication assumes a ratio scale. Moreover, due to the commutative property of multiplication, a risk with a low risk likelihood and high risk impact may receive the same risk level as one with a high risk likelihood and low risk impact, which may be misleading because they might need another mitigation strategy. This is often handled by weighting the value of the risk impact so that it is double or triple the value of the risk likelihood when the numbers are combined to give a single rating.

Quantitative risk analysis assigns risk values on a ratio scale (Fenton 2020). This enables more objective comparisons, sensitivity analysis, and probabilistic modeling (e.g., Monte Carlo simulations), supporting more precise and data-driven decision-making. However, this method requires reliable historical data and advanced analysis skills, which may not be feasible in all projects.

The **hybrid approach** aims to balance both perspectives, often employing estimation methods to define confidence intervals rather than single-point estimates. This avoids the false precision of purely quantitative models while improving the ambiguity of simple qualitative rankings. Hybrid methods can better express uncertainty but still depend on sound judgment and stakeholders' alignment. Operations on intervals follow the standard interval arithmetic, e.g., $[a,b] + [c,d] = [a+c, b+d]$, and $[a,b] \cdot [c,d] = [a \cdot c, b \cdot d]$.

5.3 Change Management

5.3.1 TME Role in Change Management Process

Change management is a structured approach to transitioning individuals, teams, and organizations from a current state to a desired future state. It is described in service management methodologies such as Information Technology Infrastructure Library (ITIL, 2026). Change management provides that changes to project scope, requirements, systems, or environments are systematically proposed, evaluated, approved, and implemented with minimal disruption to project objectives.

© International Software Testing Qualifications Board

The typical stages in the change management process in testing include:

- **Initiation of change request** is proposed through a formal request, often triggered by new business needs, risk discoveries, technological developments, or regulatory updates.
- **Impact analysis.** The change is assessed for its potential impact on test scope, quality, timelines, costs, test resources, system integrity, and test strategies.
- **Change approval.** A change control board or other governance body reviews the analysis and either approves, defers, modifies, or rejects the change.
- **Implementation planning.** A plan is developed for incorporating the approved change, including modifications to code, documentation, test plans, test resources, and test schedules.
- **Execution and verification.** The change is implemented and verified through updated test cycles to confirm quality and continuity of performance.
- **Review and closure.** Post-implementation reviews are conducted to evaluate the effectiveness of the change and document lessons learned.

As part of the impact analysis stage, the TME evaluates how proposed changes will affect the test strategy, test plans, test conditions, test data management, and test environments. This includes identifying new test conditions, regression tests to execute, and potential risks to cover. The TME collaborates with developers, business analysts, and project managers to estimate test effort and test resource needs.

Upon approval of a change, the TME updates the testware to reflect new test objectives, timelines, and test resource allocations. The TME confirms that appropriate regression tests are executed to determine if functionalities remain intact after the change is implemented.

The TME reports to stakeholders about test-related risks introduced by changes and provides test progress updates during test execution to promote visibility and accountability. The TME helps maintain traceability between change requests, test conditions, and test cases to provide comprehensive coverage and facilitate auditability.

5.3.2 Impact of Project-wide Changes on Testing

Project-wide changes are inevitable in most SDLCs, especially in dynamic business or technical environments. These changes, while sometimes necessary for business alignment or technological progress, can significantly disrupt the effectiveness, efficiency, and predictability of the test process. Managing these changes is a key factor in confirming the overall quality and success of a project. Examples of project-wide changes include:

- Changes in requirements and business objectives
- Changes in technology stack or architecture
- Scope limitations or expansions
- Shifts in project timelines or delivery schedules
- Budget adjustments
- Organizational changes
- Regulatory or compliance-driven changes

© International Software Testing Qualifications Board

Effective change management and early impact analysis determine that the test basis is aligned with current project assumptions, constraints and objectives, allowing the test team to focus on the most relevant risk areas to provide feasible and realistic test delivery. In Agile software development and DevOps, changes are frequent and expected. A TME must integrate change responsiveness into the test process, facilitate rapid impact assessment and adaptation, and confirm that the test automation strategy and the exploratory testing strategy remain resilient to change. Good change management practices make the testing more adaptive, lean, and integrated with CI/CD pipelines.

Project-wide changes may impact any test activity within the test process. In test planning, change may require reassessment of test objectives, risk prioritization, and test scope. Without proper change management, the test schedule may no longer be feasible; test execution may conflict with development changes, and test resource availability may be disrupted.

Test monitoring and test control depend on test progress (e.g., test plan, test schedule, test scope, and risks). When change occurs, the original baseline may become obsolete, and test monitoring becomes misleading if not recalibrated. Poor change management may result in inaccurate test progress reporting, misleading test metrics, poor risk visibility, inability to control test scope, reactive rather than proactive decision-making, broken communication loops, and incomplete or inaccurate test reporting.

In test analysis and test design, change may force testers to revise test conditions and test cases or modify test environment or test data. Changes in requirements, test design, or test scope may invalidate existing test cases or necessitate new ones. If these changes are not managed properly, test cases may no longer reflect current system behavior; critical defects may go undetected due to insufficient or obsolete coverage, and test effort is wasted on tests that no longer provide value. Project-wide changes often affect timelines, test resources, and deliverables.

In test implementation and test execution, a change may impact test tooling and test automation frameworks. It may require re-baselining or re-ordering of test cycles or regression testing of test suites. If change is not communicated or tracked properly, test scripts and test procedures may become obsolete or misaligned, causing false-positive results or false-negative results. New or updated functionalities may not be adequately covered. This may cause test execution delays, compromised test execution metrics, and escaped defects.

During test completion, changes to project scope, requirements, or deliverables may render the original exit criteria invalid. Poor change management may cause test completion to be premature or delayed. Test completion may occur with unresolved defects, risk gaps, or incomplete or inconsistent testware (e.g., traceability matrices, test case repositories, and test completion reports). This may make handovers to maintenance, support, or audit teams problematic. The purpose of test completion is also to identify process improvements and lessons learned. If changes are unmanaged, the root causes of delays, defects, or test inefficiencies may not be traceable, leading to repeated errors in future projects, causing technical debt.

To systematically perform impact analysis of project-wide changes on testing, the TME can apply several analytical techniques. Examples include:

- Impact analysis: identifies which test conditions are affected and determines the extent of rework needed
- Requirements traceability analysis: identifies which test conditions are impacted by changes in requirements and helps update affected areas while maintaining coverage
- Risk analysis: determines whether new tests need to be designed for emerging risks and supports test effort prioritization based on risk level
- Test effort estimation: enables reallocation of test resources and supports decisions about test scope reduction or test automation to meet deadlines
- Coverage analysis: reveals gaps or redundant tests caused by test scope or design changes and determines that modified or new areas are thoroughly tested

© International Software Testing Qualifications Board

5.4 Test Management Considerations for Project Types

5.4.1 Managing in Various Project Types

The TME often oversees projects where some or all development is handled by external suppliers or from their own organization but located abroad. These situations bring unique challenges, (e.g., late testing involvement, minimal documentation, unknown quality levels, or unclear ownership), even though clear contracts exist with suppliers or internal processes are defined. The TME must strike a balance between applying proven test practices and adjusting to a new test process. Several common types or partial SDLC projects require a tailored test management process and/or test strategy.

Integration projects, especially those with externally developed components, typically involve the test team later than in internal projects. Delivered code is often assumed to be component tested and ready for component integration testing. Since the TME may not have participated in the requirements or design phases, test planning and test preparation time is limited.

A key step is identifying the expected quality level. Was the software previously tested? Is it considered stable or still under development? Smoke testing helps verify major functionality, followed by targeted exploratory testing or checklist-based testing. This helps assess whether further functional testing or non-functional testing is required.

For third-party or commercial-off-the-shelf software, assumptions about quality must be validated through system integration testing. In some cases, formal acceptance testing is required, with the TME defining acceptance criteria, executing acceptance tests, and reporting test results. A failed acceptance test may lead to more comprehensive testing or even rejection of the incoming software depending on the contractual quality criteria.

Maintenance projects. Maintenance releases, whether planned updates or emergency fixes, require careful testing to prevent regressions. Regression testing suites, especially automated ones, are essential for quickly validating existing functionality while allowing manual testers to focus on the new changes.

In urgent scenarios, development teams may bypass initial test levels, shifting more responsibility to the test team. Thorough system testing is crucial. Impact analysis can help target areas affected by changes. Additionally, scope creep is common in maintenance projects, so the TME must assess and communicate the risks of including extra fixes or features.

Hardware/software and embedded systems. Testing embedded or hardware-integrated systems involves unique challenges due to complex configurations and prototype equipment. Test scheduling must account for potential equipment failures and system downtimes. The TME needs close collaboration with hardware and firmware teams to manage frequent changes and integration defects. Established SLAs from the hardware teams are helpful to address defects with the hardware and to troubleshoot failures.

A robust integration strategy is key. Using a dedicated integration test environment to test unverified hardware/firmware before full system testing can reduce disruptions. Coordinating component delivery and aligning schedules across development teams provides efficient and timely testing.

Safety critical systems. Safety-critical projects demand strict compliance with regulations and industry standards. Test teams must be engaged throughout the SDLC, participating in reviews, documentation, and traceability efforts. Documentation must be controlled through versioning, with all changes verified and approved. These projects often involve mandatory use of specific tools, documentation practices, and coverage levels. The TME must comply with all regulatory and safety-related requirements to support product certification and release.

© International Software Testing Qualifications Board

6 Test Monitoring and Test Control – 495 minutes

Keywords

quality control, test control, test monitoring, test progress report, test reporting

Domain Specific Keywords

cause-and-effect diagram, control chart, dashboard, goal-question-metric, measurement, metric, Pareto chart, scatter plot

Learning Objectives for Chapter 6:

6.1 Metrics and Measurement

ELTM-6.1.1 (K3) Prepare a set of the most effective metrics to be used for test monitoring

ELTM-6.1.2 (K5) Evaluate a test measurement and test reporting process

6.2 Test Reporting

ELTM-6.2.1 (K2) Distinguish between internal and external test reporting

ELTM-6.2.2 (K3) Apply appropriate data visualization techniques to create effective test progress dashboards

ELTM-6.2.3 (K4) Select the proper level and media for providing test progress reports

6.3 Interpreting the Test Results

ELTM-6.3.1 (K3) Apply basic quality control tools

ELTM-6.3.2 (K4) Analyze a given test condition based on available data

ELTM-6.3.3 (K3) Apply test control actions to resolve deviations from the test plan

© International Software Testing Qualifications Board

Introduction to Test Monitoring and Test Control

Introduction

Measuring and communicating test progress and test results is essential for providing transparency, informed decision-making, and continuous improvement. This chapter discusses the skills to define effective metrics, evaluate measurement and test reporting properly, and interpret test results in context. These competencies enable TMEs to maintain control over the test process, clearly communicate progress to stakeholders, and take corrective actions when necessary (Fenton 2000).

6.1 Metrics and Measurement

6.1.1 Selecting Test Metrics

The TME prepares and uses metrics that provide meaningful insights into test progress, quality, and effectiveness. The goal is to facilitate informed decision-making through the analysis of both quantitative and qualitative data. Examples of test metrics are given in (ISTQB-TM).

Effective metrics are not just about measurements. They are about driving action, enabling insight, and supporting decision-making. Metrics are collected to (Humphrey 1989):

- **Understand**, to establish baselines, standards, and goals, derive models of the test processes, examine relationships between test process parameters, and provide better estimates of test effort, costs, and test schedules
- **Evaluate**, to perform cost/benefit analysis, analyze trends, or determine if established goals are being met
- **Control**, by providing triggers based on variances, thresholds or control limits
- **Predict**, to estimate the future values of different attributes

Poor metrics can mislead, distort reality, or waste test effort. The TME must confirm their metrics have the following core characteristics:

- **Relevance.** The metric must measure something that directly supports the goals of the test effort or the overall project. For example, measuring EDR is relevant if the project aims to reduce production defects.
- **Clarity.** The metric must be clearly defined, unambiguous, and easily understood by all stakeholders. For example, when the test case passed rate metric is used, it must be clear if it includes only executed tests or if it covers partially automated test suites.
- **Consistency.** The metric must be measured consistently over time and across comparable projects or test teams. For example, if code coverage is measured using different tools or configurations, cross-project comparisons may become invalid, since different tools often measure the same metric differently (Alemerien 2014, Horváth 2019).
- **Actionability.** Metrics should lead to insights that can inform a decision, trigger corrective action, or drive continuous improvement. For example, a rising trend in “defects reopened due to incomplete fixes” can trigger RCA or developer training.
- **Traceability.** The metric can be traced back to its source data, allowing verification and validation. For example, if a test execution metric is questioned, stakeholders must be able to trace it to specific test logs or test tool outputs.

© International Software Testing Qualifications Board

- **Timeliness.** The metric is available early enough to influence decision-making. For example, if the DDP is only calculated after the final acceptance phase, it becomes a post-mortem measure with limited risk impact. However, it may be valuable for planning the next project.
- **Feasibility and efficiency.** The metric can be collected at a minimal cost, effort, or disruption to the test process. For example, automated test pass rates from CI/CD pipelines are more feasible than manually aggregating test results.
- **Predictive value.** The metric provides early warnings or forecasts about potential future defects. For example, a component with high cyclomatic complexity is prone to defects.

A popular approach for defining and selecting meaningful software metrics is the GQM approach. This method helps the TME confirm that every metric collected is purpose-driven and aligned with project goals (Basili, 1984). The GQM approach consists of three main steps:

- **Goal** (conceptual level). Define what to achieve. Set clear, high-level goals related to the test process. These goals should be specific to the project context and might relate to quality, efficiency, risk reduction, or customer satisfaction.
- **Question** (operational level),. Determine what needs to be known. Formulate specific questions that, when answered, will determine whether the goal is being met. These questions break the goal into measurable elements.
- **Metric** (quantitative level). Define what will be measured. Identify the specific data or indicators needed to answer each question. These become the required metrics.

6.1.2 Evaluating the Test Measurement and Test Reporting Process

Evaluating test measurement and test reporting involves assessing whether the data collected and reported effectively supports decisions related to product release readiness, process improvements, resource management, and risk mitigation. The goal is to determine how well the process tracks progress toward defined test objectives and quality goals and whether it delivers actionable insights to stakeholders. This evaluation should identify strengths, gaps, inefficiencies, and opportunities for continuous improvement. Key aspects of this evaluation include:

Evaluating objectives and context alignment. The TME verifies whether the measurement objectives remain relevant and clearly aligned with current project needs and organizational goals. This includes evaluating:

- Whether the measurement correctly supports informed decision making and analyses, such as release readiness, defect trends, quality assessment, and productivity
- Whether the process accounts for contextual factors such as project constraints, the SDLC, technology used, regulatory requirements, and test resources
- Whether test reporting objectives (e.g., risk management and test progress) and test scope (e.g., test levels and release cycles) are still appropriate and meaningful

Assessing stakeholder satisfaction and information utility. The TME evaluates whether stakeholders (e.g., product owners, test leaders, executives) are receiving the relevant and actionable information in a format they can use. This includes:

- Determining if stakeholders are using test reports to make informed decisions
- Assessing if information is presented at the right level of detail for each audience
- Identifying unmet needs, redundant information, or information overload
- Gathering direct feedback from stakeholders to refine test report content and delivery

© International Software Testing Qualifications Board

Validating metric relevance. The TME evaluates whether the selected metrics are reliable, valid, and well defined. Reliability means that multiple measurements of an identical entity give the same or similar results (i.e., the standard deviation is low). Validity means that the metric accurately measures what it is expected to measure (i.e., it captures the real value of the attribute it is intended to describe). To have metrics that are both reliable and valid, there must be agreement on standard definitions for the entity and its attributes that are being measured (Westfall, 2009).

Each metric should be measured on an appropriate measurement scale. Four main measurement scales are:

- Nominal assigns an item to a classification that categorizes the attribute of the entity.
- Ordinal classifies the attribute of an entity by order, but there are no assumptions about the magnitude of the differences between categories. This allows the mathematical relational operators to be applied to measurement values (<, >, ≥, ≤, =, ≠).
- Interval, where the exact distance between the scales is known. This allows the mathematical operations of addition and subtraction to be used.
- Ratio provides the exact distance between the scales and there is an absolute zero point. This allows all mathematical operations to be used, including multiplication and division.

Basic metrics are measured directly. Derived metrics are calculated from one or more basic metrics, and their measurement scale depends on the scales of the basic metrics used and the mathematical operations applied. Hence, for a derived metric, the TME verifies that only mathematical operations that are valid for the lowest-scale metric involved are allowed.

To measure, the entity being measured must be determined. The TME checks if its attributes (i.e., measurable properties) are accurately described. Finally, a mapping system, called the measurement method, must be evaluated and accepted. In software testing, for many metrics, no standard mapping system has been widely accepted (e.g., defect severity can be measured as critical, major, or minor, or on a 4-point scale, 1, 2, 3, or 4).

Reviewing data sources and collection methods. The TME examines the accuracy, timeliness, and consistency of data sources and how data is collected. Key questions include:

- Are current tools and sources (e.g., CI/CD pipelines and defect management systems) delivering trustworthy data in an efficient manner?
- Is the frequency of data collection appropriate for decision timelines?
- Are responsibilities for data entry, validation and oversight clearly assigned and executed?
- Are there issues with missing, duplicated, incomplete, or incorrect data?

Evaluating report formats and visual communication. The TME assesses whether test reports and visualizations (see Section 6.2) clearly convey insights and are appropriate for the intended audience. The TME reviews test report formats and dashboards regarding clarity, readability, and consistency, and evaluates whether visualizations (e.g., charts and graphs) aid or hinder interpretation. The TME checks for standardization in terminology and layout and identifies any usability defects that may affect comprehension or decision-making.

Checking test reporting frequency and schedule. The TME evaluates the effectiveness of the test reporting schedule and test automation. Test reports should be delivered frequently enough to support timely decisions. They should be synchronized with development and test cycles. The test report generation should be automated where feasible (e.g., creating dashboards from test tools, as described in Section 6.2) to improve efficiency and reduce errors.

© International Software Testing Qualifications Board

Evaluating baselines and targets. The TME reviews how actual measurements compare to defined baselines and targets. Baselines and realistic targets or thresholds should be set for each metric based on historical data or industry benchmarks. Targets should be specific, measurable, achievable, relevant, and time-bound (SMART). The TME determines if the defined variances or control limits are acceptable.

Assessing the measurement and test reporting procedures. The TME evaluates whether the measurement and test reporting procedures are well-documented, standardized, repeatable, followed consistently, and test results are validated appropriately. The measurement plan should reflect actual practices. Measurement and test reporting should be aligned with the test strategy and broader quality management framework. Responsibilities should be clearly defined and upheld.

Evaluating report distribution channels. The TME determines whether test reports are reaching the intended recipients securely, efficiently, and on time and appropriate distribution channels are being used. The TME makes sure that access controls are in place for sensitive or compliance-related information. The TME evaluates if there are delays or bottlenecks in delivery affecting decision-making.

Pilots/dry runs. The TME may conduct a dry run or limited pilot to verify the clarity and interpretability of metrics, data quality and reliability, and ease of collection and analysis. This tests the usefulness and accuracy of the measurement process. The TME gathers feedback which can be used to refine the approach.

Test monitoring, reviewing, and improving. The TME tracks whether metrics are supporting intended goals, identifies gaps, redundancies, or misinterpretations, and adjusts metrics, tools, or processes based on evolving needs and feedback. Regarding the test reporting process, the TME collects stakeholders' feedback regularly, reviews whether test reports are aiding decision-making, and updates report formats or metrics as project or organizational needs evolve. The purpose is to continually improve the effectiveness of the measurement and test reporting process.

6.2 Test Reporting

6.2.1 Internal and External Test Reporting

Internal test reports are circulated within the project team (e.g., testers, test managers, developers, Scrum Masters, product owners, DevOps engineers, and project managers). Their purpose is to manage day-to-day testing, identify and resolve defects quickly, track test progress and test resource usage, facilitate collaboration, enable test leader and test team coordination, and drive test team-level decision-making. Internal test reports are typically characterized by high detail, technical depth, raw data, and frequent updates (daily to weekly). They focus on information essential for the project team, such as test progress, defects, blocked tests, or coverage. The tone and style of internal test reports are informal to semi-formal, technical, and test team focused. While internal test reports may be less formal than external test reports, the quality of the data is just as critical, particularly when management decisions will be based on the information in these test reports. Typical formats of these test reports include spreadsheets, dashboards, internal emails, test team updates, or test reports from defect management tools.

© International Software Testing Qualifications Board

External test reports are produced for external stakeholders. Their purpose is to provide transparency, demonstrate compliance, product quality, and test thoroughness, communicate risk and readiness to release, track contractual commitments, and provide business-level insights and accountability. External test reports are typically characterized by summarized, high-level insights tailored for business or oversight audiences. Their frequency is either milestone-based, or when a phase ends (e.g., test completion reports and regulatory reports). They focus on overall product quality, risk status, defect trends, and compliance metrics. The tone and style of external test reports are formal, business-focused, clear, and polished. Typical formats of these test reports include formal documentation, presentations, audit logs, or executive dashboards.

Both types of test reports should be aligned with consistent metrics and definitions to avoid confusion. Internal test reporting should be automated whenever possible to provide agility. In contrast, external test reporting should undergo a thorough review to determine accuracy, clarity, and an appropriate tone. In both types, traceability from raw data to high-level summaries should be confirmed for auditability.

6.2.2 Data Visualization in Test Progress Dashboard

Data visualization plays a critical role in test progress dashboards by transforming raw test data into clear, actionable insights for stakeholders at various levels. Data visualization supports decision-making (e.g., release readiness), enables trend tracking and early defect detection, and engages both technical and non-technical stakeholders.

The goal of visualization is not decoration, but to convey test progress, quality, risks, and readiness concisely and persuasively. Therefore, the chosen visualization should have the simplest form that accurately tells the story the TME needs to tell. The visualization type should be aligned with the data and stakeholders, and color coding should be used effectively (e.g., green = good and red = risk/problem). Labels, legends, timeframes, and thresholds should be provided to avoid misinterpretation. This syllabus describes the following data visualization techniques.

Tables allow for presenting detailed data where exact values are important (e.g., the number of defects by their status in different months). They are compact and well-structured, but not as visually engaging as graphs and charts. They do not provide a summary view or trends.

Pie charts show proportions of a whole (e.g., defect distribution by severity). They are easy to read and suitable for highlighting dominant segments. However, they are poor at showing minor differences or changes over time.

Bar charts are used for comparing categories (e.g., number of test cases by test results). They are intuitive and straightforward, making them ideal for side-by-side comparisons. However, they can get cluttered with too many categories, and they also do not show trends over time.

Stacked bar charts show part-to-whole relationships across multiple categories. They display overall quantity and composition simultaneously and are ideal for iteration-by-iteration or component-wise breakdowns. However, it is not easy to compare specific components across the bars.

Grouped bar charts display multiple bars side by side for each category, allowing comparisons of multiple sub-categories within the same group. They are suitable for comparing values within and across user groups but may be hard to read when there are many categories or user subgroups.

Line charts show changes or trends over time (e.g., test results or defect trends). They are excellent for spotting patterns and forecasting. However, they are less effective with sparse or non-sequential data and require careful scaling of the axes.

© International Software Testing Qualifications Board

Box charts (i.e., also known as box-and-whisker diagrams) visualize data distribution, variability, and outliers. A box chart summarizes a set of numerical data by showing the minimum and maximum values, excluding outliers, the lower and upper quartiles, the median, and outliers which are plotted as individual points beyond the whiskers. They allow for the comparison of distributions and do not assume a normal distribution. However, non-technical stakeholders might need help with interpretation. Additionally, box charts are less meaningful when there are very few data points.

Gauges visualize a single KPI (e.g., test completion or release readiness). They are very intuitive for business stakeholders and instantly show status (e.g., safe, warning, and danger zones). However, they are limited to one data point at a time and are not suitable for trend or comparative data.

6.2.3 Preparing Test Progress Reports

Effective communication of test progress is essential for aligning stakeholders, managing expectations, and facilitating informed decision-making. The TME is responsible for selecting appropriate levels of abstraction and communication media to deliver impactful, timely, and targeted test progress reporting.

Selecting the proper level of the test progress report is vital because it communicates relevant information to the right stakeholders, in the right format, at the right time. Different stakeholders have different information requirements. Well-leveled test progress reports support stakeholders in understanding the current situation, identifying risks early, and making timely decisions. When test progress reports consistently deliver the right insights, stakeholders trust the test process and are more confident about the product quality. Three typical levels of test reporting can be distinguished:

- **Strategic** is required by senior management and executives. It focuses on high-level KPIs, business risks, and general coverage trends in the context of project milestones. Test reports at this level are written in non-technical, outcome-driven language that aligns with business objectives.
- **Tactical** is required by project managers, development leads, and test leaders. It focuses on progress against test plans, test resource usage, defect trends, risk/requirements coverage, test execution trends, and blocking defects. Test reports at this level employ a balanced mix of technical and management terminology.
- **Operational** is required by testers, developers, and engineering. It focuses on daily test execution details, test case status, test environment readiness, and defect status. Test reports at this level use technical and detail-oriented language.

Once the test reporting levels are defined, the TME decides on the form in which the test reports will be delivered to stakeholders:

- Static test reports (e.g., PDF files and spreadsheets) are suitable for formal documentation, audits, and executive summaries. They are archivable, shareable, versionable, and structured.
- Dashboards are real-time visualization tools. They are suitable for tactical and operational test reporting, as they can provide detailed information in real time. They are up-to-date, often interactive, and feature drill-down capability, making them extremely useful for performing various types of analyses.
- Presentations (e.g., slide decks) are suitable for stakeholders and milestone reviews. Such test reports are typically presented in a meeting, allowing the presenter to provide additional detailed information if needed. Presentations are customizable, contain focused messaging, and are a form of visual storytelling.

© International Software Testing Qualifications Board

- Emails and instant messaging are suitable for quick updates or urgent notifications. They are fast, informal, and very accessible. This form of test reporting is also common in Agile software development, which promotes lightweight documentation.
- Meetings and verbal briefings are suitable for alignment, question and answer sessions, and complex defect explanation. They are interactive, clarify context, and allow substantial discussion.
- Flipchart or whiteboard-based test reporting (e.g., burndown charts, Scrum boards, and Kanban boards) is typically used by Agile software development teams that are co-located or use tool-supported virtual equivalents. Test reports that use this form are low-tech, but provide immediate visibility, which promotes shared understanding and transparency. Such test reports engage collaboration by stimulating discussions and problem-solving, encouraging test team ownership of quality. They are easy to update in real time.

When selecting the right media for test reporting, the TME should consider different selection criteria, such as:

- Type of test reporting (internal or external, (see Section 6.2.1)
- Stakeholder's role and information needs
- Project phase and milestones
- Criticality and sensitivity of information
- Tool and technology availability
- Regulatory and documentation requirements
- Time constraints and frequency requirements

The higher the stakeholder's position in an organization, the more relevant general trending information may be than a detailed report. Charting and visually appealing graphs become increasingly important as the technical expertise of the consumer declines. Because project managers may have little time to devote to reviewing test reports, quick visuals, easily interpreted charts, and trending diagrams are handy for conveying a point quickly and accurately (see Section 6.2.2).

6.3 Interpreting the Test Results

6.3.1 Quality Control Tools

By applying basic QC tools, the TME can systematically interpret data, detect patterns and anomalies, identify the root causes of defects, and prioritize corrective actions. This capability directly supports process improvement, defect reduction, better stakeholder communication, and enhanced decision-making. When employed correctly, these techniques help prevent recurring defects, reduce costs and time overruns, and increase confidence in the QC process. This syllabus outlines four examples of analytical tools used for QC.

A **control chart** is a statistical QC tool. It monitors one of the process attributes over time by identifying improbable patterns that may indicate that the process is out of control. A control chart has a centerline (CL), upper control limit (UCL), and lower control limit (LCL), calculated using statistical techniques and based on data collected while the process is running and considered under control. UCL and LCL are based on the standard deviation of the process (see Section 5.1.1) and set at ±3 standard deviations from the CL. The TME can identify unlikely patterns that may suggest some problems with a process parameter. These patterns include (Westfall, 2009):

© International Software Testing Qualifications Board

- A single data point below the LCL or above the UCL
- Any two out of three data points in a sequence beyond two standard deviations from the CL
- Any four out of five data points in a sequence beyond one standard deviation from the CL
- Any eight successive points on the same side of the CL
- Any eight successive points in a trend (all up or all down)

A **cause-and-effect diagram** (i.e., also known as a fishbone diagram or Ishikawa diagram (Iskihawa 1985)) is an RCA tool. It helps to organize potential causes of a problem into categories, (e.g., people, processes, tools, materials, environment, and management). The process begins by defining the effect, then branching out into major categories and sub-causes through brainstorming and iterative “why?” questioning.

A **scatter plot** graphically represents the relationship between two variables (e.g., the number of test cases executed, and the number of defects found). A scatter plot allows the TME to identify correlations between factors, which can uncover meaningful patterns. For example, a strong correlation between code churn and defect density (DD) may indicate components that are high-risk. It is essential to remember, however, that correlation does not necessarily imply causation.

A **Pareto chart** combines a bar and line graph (see Section 6.2.2) to display defect causes or defects in descending order of frequency, often following the 80/20 rule (80% of defects come from 20% of causes). Pareto charts help the TME prioritize focus areas by identifying the most common sources of defects or test failures. They enable efficient defect prevention and resolution, prioritize corrective actions, and maximize return on quality improvement efforts.

6.3.2 Data-Based Test Process Analysis

To be effective, assessing the current test process state and product quality based on metrics should always be data-driven. This section looks at each group of activities within the test process and discusses how metrics collected during these test activities should be interpreted.

Test planning. Adherence to the estimated test schedule and budget is critical to all test projects. Planned requirements coverage should align with the project's scope and business priorities. Effort metrics can be used to assess the feasibility of the test project quickly. Test environments and tools readiness are needed to execute the plan. Historical data enables the identification of high-risk or defect-prone areas that require more rigorous testing. If risk levels or test resource constraints are too high, features or test objectives should be reprioritized.

Test monitoring and test control. Tracking deviations in test progress allows the TME to identify delays early and adjust test scope or test resources accordingly. Analyzing trends in DDPs helps to determine if testing is adequate or should be redirected. Monitoring the ratio of failed tests identifies quality concerns in specific areas. Effort variance data determines whether a reallocation of test resources is necessary. Evaluating actual coverage (e.g., requirements coverage) detects scope creep or under-tested features. The TME should evaluate how well telemetry, logging and monitoring data enable RCA and continuous improvement. The TME should also determine that the data is consistent with data reported by other teams (e.g., developers).

Test analysis and test design. Assessing the completeness of traceability determines if all requirements are testable and covered. Low numbers of test conditions per requirement may indicate under-specification or a lack of understanding of the requirement. Test design coverage metrics verify the thoroughness of the test conditions that have been derived. High ambiguity or inconsistency rates in requirements signal the need for further clarification or stakeholders' engagement. Defects found during the review cycles and during test design, if resolved, prevent those defects from escaping to the next phase. If a defect is identified but not resolved, additional work is incurred due to its existence. However,

© International Software Testing Qualifications Board

the time spent addressing a known defect is usually less than that spent detecting a previously unknown defect. Cost, in terms of money or time, is often associated with these defects. That cost is typically reported, usually in terms of cost savings, as part of test reporting. Defect trending is also essential at this phase. If numbers of defects are higher than expected in the review cycles, this may indicate a lower quality of the work products than was anticipated. If so, multiple review cycles may be necessary to achieve an acceptable level of quality before these work products progress to the next phase in the SDLC.

Test implementation and test execution. Readiness of test cases, test data, and test environments promotes timely test execution, proper test execution order, test team efficiency, and the avoidance of bottlenecks. Testing without 100% readiness may be an acceptable, even a preferred strategy. However, if the missing elements are required for the highest risk level items, it will force higher risk level testing toward the end of the test cycles and skew metrics, and that is generally not acceptable. Coverage must be closely monitored. If coverage is not proceeding according to plan, risks may appear where they were not planned. Passed/failed information determines if the expected passed/failed rate is being achieved. If a higher number of test cases is failing than was expected, additional test cycles may be required. If a higher number of test cases is passing than was expected, the TME needs to determine if testing is sufficient, where the extra time in the test schedule should be focused, and if plans should be adjusted for future testing. Test flakiness rate shows the level of test stability. Analyzing test automation coverage evaluates efficiency opportunities and test repeatability. Defect severity distribution can help with test prioritization and escalation strategies. Test effectiveness broken down by test types informs the TME how productive the various test types are in terms of defect detection, risk coverage, and tester skill usage.

Test completion. Achieved coverage allows for assessing if the exit criteria are met. Evaluating DD and severity trends determines residual risk and release readiness. Identifying open defects and their business impact supports making go/no-go decisions. Lessons learned can be used to refine future test planning, improve test estimates, and reduce escaped defects. Incorporating stakeholders' feedback validates the quality of the test process and test results.

6.3.3 Performing Test Control

Activities described in Sections 6.1.1-6.3.2 allow the TME to collect proper metrics and, based on their analysis, apply reasonable, effective, and efficient test control to resolve deviations from the test plan. The following control directives are described and categorized by test activity within the test process:

Test planning

- Re-prioritize features or test objectives based on risk, business value, or test resource constraints if coverage is misaligned, or test environment/tool readiness is delayed.
- Adjust budget or timelines proactively if effort metrics show infeasibility.
- Reduce test scope on lower-priority items to focus on high-risk, defect-prone areas identified via historical data.
- Escalate infrastructure delays if test environments or tools are not ready, with fallback planning (e.g., simulating test environments or temporarily using a test double).

© International Software Testing Qualifications Board

Test monitoring

- Shift test resources or adjust timelines if test effort variance is high or test progress lags.
- Focus testing on components and systems with high failure rates to isolate and resolve quality defects early.
- Redirect the test strategy if the DDP drops abnormally. This could indicate poor test design or coverage gaps.
- Manage scope creep by either freezing the test scope or obtaining change approvals if coverage requirements exceed the planned amount.
- Introduce parallel test execution if coverage or defect trends suggest increasing residual risk.

Test analysis and test design

- Initiate requirement refinement sessions or stakeholders' clarifications if ambiguity or inconsistencies are high.
- Introduce additional review cycles if review-phase defects exceed the threshold to prevent defect propagation.
- Increase test granularity (i.e., more test conditions per requirement), where low coverage indicates weak understanding.
- Temporarily halt test design until critical traceability gaps are resolved to prevent building irrelevant or misaligned tests.

Test implementation and test execution

- Enforce a test readiness quality gate for high-risk areas if full readiness cannot be met; delay test execution or reprioritize tests accordingly.
- Add exploratory testing or targeted tests where test automation effectiveness is low.
- Refocus manual testing on complex or high-severity areas when test automation is underperforming.
- Pause release-critical testing if high severity defects emerge; escalate for rapid triage and resolution.
- Schedule additional test cycles if the test failure rate exceeds the expected rate or reduce the test scope to stabilize product quality.

Test completion

- Delay completion or release if exit criteria are not met, or critical defects remain unresolved.
- Weaken or waive exit criteria if they cannot be met due to time, test resource, or budget constraints. This must be applied very cautiously and only when the risk is understood and deemed acceptable, residual risk is communicated and accepted by stakeholders, and remaining open defects are low severity or have known workarounds.
- Prioritize defect fixes based on business impact and severity to reduce residual risk.
- Document and act on lessons learned; introduce process improvements to reduce deviations in the next test cycle.
- Adjust test process KPIs based on stakeholders’ feedback for continuous quality improvement.

© International Software Testing Qualifications Board

General test control tactics across test activities include:

- Using risk-based testing (i.e., shifting focus dynamically on higher-risk areas as new risk information emerges)
- Implementing early warning indicators (e.g., using trends to trigger pre-defined test control directives)
- Using adaptive test resource allocation (e.g., moving testers across tasks/components based on metrics like effectiveness or DD)
- Using escalation protocols by creating thresholds, (e.g., if failure rate > X% then trigger management or cross-team escalation)

© International Software Testing Qualifications Board

7 Evaluating Effectiveness, Efficiency and Satisfaction and Improving Test Policy – 315 minutes

Keywords

defect density, escaped defect rate, test framework

Domain Specific Keywords

metric, satisfaction

Learning Objectives for Chapter 7:

7.1 Effectiveness, Efficiency and Satisfaction Metrics for the Test Process

ELTM-7.1.1 (K2) Explain the purpose of tracking effectiveness, efficiency and satisfaction metrics

ELTM-7.1.2 (K4) Analyze effectiveness, efficiency, and satisfaction metrics for the test process, identifying opportunities for improvement

7.2 Effectiveness, Efficiency and Satisfaction Metrics for the Test Policy Objectives

ELTM-7.2.1 (K5) Evaluate the effectiveness, efficiency, and satisfaction metrics for a test policy as a means of monitoring its quality

7.3 Test Policy Improvement

ELTM-7.3.1 (K3) Use managerial reviews for test policy assessment

ELTM-7.3.2 (K4) Analyze the results of a managerial review to formulate plans for test policy improvement

© International Software Testing Qualifications Board

Introduction to Evaluating Effectiveness, Efficiency and Satisfaction and Improving Test Policy

Introduction

TMEs must continuously evaluate the effectiveness and efficiency of testing and the satisfaction with testing to make informed decisions and identify opportunities for improvement. Therefore, TMEs should be familiar with qualitative and quantitative evaluation methods and tools, as well as the potential pitfalls and sources of misguidance in evaluations. TMEs should review the test processes, organizational test practices, and test policies on an ongoing basis and update them as required. To do this, they must understand how to improve target orientation by conducting continuous, transparent and fair evaluations, and by communicating and test reporting appropriately.

7.1 Effectiveness, Efficiency and Satisfaction Metrics for the Test Process

7.1.1 Tracking the Test Process Metrics

Test process metrics are used to do test monitoring quantitatively and provide an objective basis for improvement (Singh, 2011). They have an operational/tactical scope and focus on the delivery performance of the test process. They provide answers to questions related to goals (Basili, 1984) for an intended audience, (e.g., test managers, testers, QA leaders, and developers). Metrics are monitored iteratively on a daily or iteration basis. The metrics should be adapted if required. Note that metrics should not be used in isolation as single views on test processes or on a test policy, but rather in combination with several metrics and as time series to see trends of the metrics over time.

The TME is responsible for incorporating test process metrics into organizational test practices, so that they can be used throughout the organization, within the same business unit, or on related projects. The TME should be aware of the interactions between metrics, KPIs and organizational culture and performance. What does it mean, for example, if a project is highly efficient but not highly effective, or vice versa? What if a project is highly efficient but at the same time experiences high tester turnover?

Effectiveness metrics examine the extent to which the test process produces the required work products and test results, including risk mitigation, (i.e., how well testing was done). Effectiveness metrics should be tracked to determine that testing finds relevant defects, validates requirements, and provides product quality.

Efficiency metrics examine the effort, duration and test resource expenditure required to achieve these work products and test results, (i.e., how much value was delivered at what cost). They should be tracked to assess how well test resources are being used to achieve test results.

Satisfaction metrics examine how well the test process delivers actual and perceived value to the test stakeholders. They focus on how well test results meet the expectations of stakeholders and project team members. These metrics should be tracked to measure the perceived quality of the product and the value delivered by the test process.

© International Software Testing Qualifications Board

7.1.2 Using Metrics to Identify Opportunities for Test Process Improvement

Metrics should always be defined specifically for the test policy, test process, and test projects in place. This section provides a representative set of metrics that can be used.

Effectiveness metrics

Defect density (*DD*) quantifies the concentration of defects discovered relative to the functional or structural size of the test object:

$$DD_X = \frac{Total\ Defects\ Found}{Total\ Number\ of\ Item\ X\ Inspected}$$

where item X is measured in units, (e.g., KLOC, function points, user story points, or requirements). High DD suggests the need for improvement in coding standards, using code reviews, refactoring, static analysis, strengthening requirements-to-design traceability, or early validation. Low DD may indicate a genuinely high product quality (e.g., after shifting effort to regression testing, test automation efficiency, or test suite optimization) or undetected defects (e.g., need to improve test design, coverage, risk-based testing, exploratory testing, or test environments).

DD should not be reported in isolation but paired with defect severity to distinguish between high-impact architectural issues and low-priority cosmetic defects.

Test case effectiveness (TCE) measures the test object through the lens of testing itself:

$$TCE = \frac{Total\ Defects\ Found}{Total\ Test\ Cases\ Executed} \cdot 100\%.$$

Low TCE may indicate that test cases are overly "happy-path" or simply weak, suggesting to use boundary conditions, negative paths, or error scenarios and to perform test case reviews. Because a low value can also stem from high-quality software, always pair TCE with coverage metrics to accurately assess whether the test results indicate a weak test suite or a mature product.

Escaped defect rate (EDR) measures the percentage of defects that escaped the test process:

$$EDR = \frac{Defects\ Found\ in\ Production}{Defects\ Found\ During\ Testing\ +\ Defects\ Found\ in\ Production} \cdot 100\%.$$

High EDR may indicate poor shift-left practices, inadequate coverage, weak risk focus, environment mismatch, weak regression testing, or ineffective defect resolution. To stabilize EDR, define a defect time window (e.g., the first 90 days post-release); otherwise, the metric fluctuates. While DDP (ISTQB-TA) measures how many defects were caught, EDR helps assess the efficiency of the removal process; together, they provide a holistic view of QC performance.

EDR should be reported with CoQ, as external failures usually cost more than internal ones.

Test result accuracy (TRA) measures the reliability of the test results:

$$TRA = \frac{Total\ Tests\ Executed\ -\ (False\ Positives\ +\ False\ Negatives)}{Total\ Tests\ Executed} \cdot 100\%.$$

Low TRA indicates vague, incomplete, or incorrect expected results, unclear pass/fail criteria, or ambiguous requirements, suggesting strengthening test oracles, clarifying requirements and acceptance criteria, using precise measurable assertions, and conducting test case reviews focused on expected results. Because false-negative results are often only identified retrospectively, calculating TRA requires a standardized observation window, similar to EDR. For more actionable insights, TRA can be split into two

© International Software Testing Qualifications Board

distinct sub-metrics: pre-deployment accuracy that tracks false-positive results, and post-deployment accuracy that tracks false-negative results.

Efficiency metrics

Test execution efficiency (TEE) quantifies the productivity of the test team:

$$TEE = \frac{Total\ Number\ of\ Test\ Cases\ Executed}{Total\ Effort\ (in\ Person\ Hours\ or\ Person\ Days)}$$

Low TEE may indicate inefficient test design, poor test environment or tooling support, skill or knowledge gaps, suboptimal test planning, a high amount of defect rework, test cases need refactoring, test environments need stabilizing, training on tools and domain knowledge is required, poor build and release stability, or the need for smoke testing before detailed testing.

Test automation efficiency ($TAEff$) is measured with respect to time (TAE_T) or productivity (TAE_P):

$$TAEff_T = \frac{Manual\ Test\ Execution\ Time}{Automated\ Test\ Execution\ Time},$$

$$TAEff_P = \frac{Number\ of\ Automated\ Test\ Cases}{Efforts\ to\ Script + Efforts\ to\ Maintain}.$$

Low $TAEff$ may indicate poor test automation design, test automation framework defects, architectural inflexibility, high maintenance overhead, skill or knowledge gaps, test environment or tooling inefficiencies, test automation code needs refactoring, test environments need stabilizing, test automation best practices and coding standards need applying, or training on tools, languages and test frameworks is required.

Return on investment (ROI) determines whether testing adds economic value.

The primary ROI driver in manual (M) testing is risk mitigation:

$$ROI_M = \frac{Estimated\ Cost\ of\ Avoided\ Failures - Cost\ of\ Manual\ Testing}{Cost\ of\ Manual\ Testing} \cdot 100\%,$$

where estimated cost of avoided failures is calculated as the sum of the products of failure probabilities and failure impacts as cost for all defects found during manual testing. Low ROI may indicate low-risk areas, over-testing, or test redundancy.

The primary ROI driver in automated (A) testing is labor substitution:

$$ROI_A = \frac{(n \cdot Manual\ Testing\ Cost\ Equivalent) - (Development\ Cost + Maintenance\ Cost)}{Development\ Cost + Maintenance\ Cost} \cdot 100\%,$$

where n is the number of test runs.

The manual testing cost equivalent is the total cost to run the same automated tests manually once. The development and maintenance costs are those for the test automation.

© International Software Testing Qualifications Board

Satisfaction metrics

Defect rejection rate (DRR) measures the percentage of defects rejected by the development team:

$$DRR = \frac{Total\ Defects\ Rejected}{Total\ Defects\ Reported} \cdot 100\%.$$

A high DDR may indicate ambiguous test basis, limited domain knowledge, or unstable test environments causing irreproducible failures. However, high DRR can be acceptable when testing highly innovative features with frequently changing requirements. Providing context prevents DRR from being used against the test team.

Customer satisfaction score ($CSAT$) is typically derived from surveys using standardized scales, such as a 5-point Likert scale (Likert, 1932). An example of $CSAT$ on this scale is:

$$CSAT_{top} = \frac{Total\ Number\ of\ 'Agree'\ (4)\ or\ 'Strongly\ agree'\ (5)\ Responses}{Total\ Number\ of\ Responses} \cdot 100\%.$$

Additionally, the average score reflects the overall sentiment of all stakeholders, including those who are dissatisfied. An increasing average indicates that TME changes are effective, even if the top two response rates remain unchanged.

Net promoter score (NPS) measures long-term loyalty and advocacy by asking how stakeholders are to recommend the test team or test services. Scores of 0–6 are detractors, 7-8 are passives (i.e., satisfied but unenthusiastic, not advocates, and vulnerable to competitors) and 9–10 are promoters; NPS is the net result:

$$NPS = Total\ Promoters - Total\ Detractors$$

Because the detractor range (0–6) is wider than promoters (9–10), NPS is highly sensitive to stakeholders' sentiment, serving as an early-warning system for test perception issues. Negative NPS provides a red flag that testing may be perceived as a hurdle rather than a value-added activity. This metric allows the TME to proactively address perception issues before they impact cross-functional collaboration.

Report the average score with NPS for context.

In large organizations, a high NPS often justifies the internal cost of centralized test centers.

7.2 Effectiveness, Efficiency and Satisfaction Metrics for the Test Policy

Test policy metrics quantitatively monitor how test activities support strategic test objectives, aligning test results with business goals. Their scope is organizational and strategic, targeting senior management, QA/QC leadership, stakeholders, and they are reviewed periodically. The TME is responsible for embedding these metrics into organizational test practices.

Metrics should measure effectiveness, efficiency, and satisfaction, and may extend existing project metrics to quantify process quality and risk. They can also be aggregated across related projects within the same organization or product line. As test objectives vary, suitable metrics differ and may cover testing, the SDLC, and project management.

© International Software Testing Qualifications Board

Test policies effectiveness metrics

Requirements stability index (RSI) measures the rate of change in baselined requirements. This index measures how volatile the test scope is that the test team must validate. It is a key predictor of test stability.

$$RSI = \left(1 - \frac{Total\ Number\ of\ Changed\ Requirements\ (Added,\ Deleted,\ or\ Modified)}{Total\ Number\ of\ Initial\ Requirements}\right) \cdot 100\%.$$

A low RSI (e.g., below 80%) indicates wasted test design, as requirement changes make test cases obsolete. An effective test policy should require requirement reviews as entry criteria to stabilize RSI before test design begins.

Test objective stability index ($TOSI$) measures whether the strategic test objectives (TO) as defined in the test policy remain consistent throughout the test project lifecycle.

$$TOSI = \left(1 - \frac{Total\ Number\ of\ Changed\ TO\ (Changed\ \ or\ Abandoned)}{Total\ Number\ of\ Initial\ TO}\right) \cdot 100\%.$$

A low $TOSI$ indicates a mid-stream shift in the test mission (e.g., from risk mitigation to date-driven release). While high stability can indicate a mature organization with a robust test policy, $TOSI$ must be interpreted with caution. Because not all test objective changes carry equal weight, a low score should trigger a qualitative review to distinguish between reactive management and necessary strategic adaptation. Rather than aiming for a perfect 100%, $TOSI$ should be used to provide visibility into the impact of shifting priorities on the overall test strategy.

Test policy compliance rate ($TPCR$) measures how well test projects or test teams comply with mandatory test policy rules, such as peer reviews, coverage, and tool usage.

$$TPCR = \frac{Total\ Number\ of\ Test\ Policy\ Compliant\ Activities}{Total\ Number\ of\ Test\ Activities} \cdot 100\%.$$

$TPCR$ can be calculated per activity, project, or test team. A low $TPCR$ may indicate an overly bureaucratic or misaligned test policy (e.g., forcing sequential development models on Agile teams). The TME can use $TPCR$ to identify test teams needing coaching rather than auditing. A high $TPCR$ should correlate with lower EDR; if not, the test policy is flawed.

Assessment or audit pass rate ($AAPR$) measures the results of formal evaluations, such as TMMi (TMMi, 2022), ISO 29119, or regulatory audits, against defined standards.

$$AAPR = \frac{Total\ Number\ of\ Passed\ Audit\ Points}{Total\ Number\ of\ Audit\ Points\ Evaluated} \cdot 100\%.$$

While $TPCR$ measures rule adherence, $AAPR$ measures their effectiveness. In regulated sectors, a high $AAPR$ reduces the cost of non-compliance, such as fines, product recalls, and legal actions.

© International Software Testing Qualifications Board

Test policy efficiency metrics

Reduction rate of test cycle time ($RRTC$) measures how efficiently a test policy changes, (e.g., CI/CD pipeline integration, automated regression testing, or shift left), reduces test cycle time (TCT), from build readiness to final test report.

$$RRTC = \frac{Baseline\ TCT - Current\ TCT}{Baseline\ TCT} \cdot 100\%,$$

Where $Baseline\ TCT$ is the average test time before a test policy change, $Current\ TCT$ is after improvements are made. The TME considers more than test cycle times, which can be inflated by delays (e.g., for test environments or test data). If the test policy ignores SLAs for test environments, $RRTC$ stays low despite improvements like increased test automation. Reductions in time are only efficient if DER remains stable, as faster testing at the expense of quality is risky.

Reuse rate of testware ($RRTW$) measures how effectively a test policy enforces modularity and standardization, such as using a central test data repository and keyword-driven test automation.

$$RRTW = \frac{Total\ Number\ of\ Reused\ Testware}{Total\ Number\ of\ Testware} \cdot 100\%.$$

$RRTW$ can be calculated at the test, project, or test team level and measures reuse of testware (e.g., test cases, test data, test scripts and test tools) used in new contexts with minimal changes (e.g., >80% identical). The TME distinguishes cloning from true reuse (i.e., referencing a single source) which reduces maintenance. An effective test policy should provide a common testware library. Low $RRTW$ (e.g., below 30-40%) indicates repeated reinvention.

Defect delay index (DDI) measures defect decay, (i.e., the period from defect introduction to defect removal) since costs and risks rise exponentially the longer a defect remains unresolved.

$$DDI = \frac{\sum_D (Phase\ of\ Defect\ Removal - Phase\ of\ Defect\ Introduction)}{Total\ Number\ of\ Defects}$$

Phases should be numbered, e.g., requirements = 1, design = 2, coding = 3, component testing = 4, component integration testing = 5, system testing = 6, acceptance testing = 7, production = 8.

DDI = 0 means defects are contained in the phase where they originated. DDI > 0 signals higher risk and cost as defects escaped quality gates. A high DDI reflects technical debt, leading to costly hotfixes and downtime.

Test overhead ratio (TOR): measures a test policy's "bureaucracy" by comparing effort spent on test process management versus test analysis, design, and execution.

$$TOR = \frac{Total\ Management\ Effort\ of\ Testing}{Total\ Technical\ Effort\ of\ Testing}$$

Management efforts (numerator) include meetings, reports, and audits. Technical efforts (denominator) include test activities of the given test process, such as defining entry and exit criteria, designing the physical test environment infrastructure, executing regression tests, and archiving testware. A healthy TOR is 0.1-0.2. Above 0.25, the test policy is too heavy, with testers spending more time on test process overhead rather than technical efforts (i.e., actual testing), common in highly regulated industries.

© International Software Testing Qualifications Board

Test policy satisfaction metrics

Feedback-to-test policy ratio ($FTPR$): measures test policy engagement and evolution, showing whether it is a living document or static shelfware.

$$FTPR = \frac{Total\ Number\ of\ Test\ Policy\ Feedbacks + Total\ Number\ of\ Change\ Requests}{Total\ Number\ of\ Test\ Policy\ Clauses}$$

Feedback on the test policy includes improvement suggestions, exception requests, and issues, often submitted as change requests. A zero $FTPR$ usually signals the policy is ignored, while a high $FTPR$ suggests it is too vague or restrictive. The TME uses $FTPR$ to trigger test policy reviews and updates, such as adjusting test automation mandates for CI/CD pipelines.

Test governance maturity score ($TGMS$). This composite score, from internal assessment or test frameworks like TMMi, measures the organization's test policy maturity.

$$TGMS = \frac{\sum_{i=1}^{n}(Rating_i \cdot Weight_i)}{\sum_{i=1}^{n}(Maximum\ Rating_i \cdot Weight_i)} \cdot 100\%,$$

where $Rating_i$ is a rating for i-th governance domain (e.g., test policy and test strategy, test organization, infrastructure, operational processes, and people) and $Weight_i$ denotes its importance. $TGMS$ is calculated using a structured rating system and weighted priorities, based on institutionalization and evidence-based performance in each governance domain. For example, $Rating_i$ scale can be 1–5, as in TMMi (1 = initial, 5 = optimization).

Weights can be adjusted to align with organizational goals (e.g., shifting from infrastructure to operational processes to prioritize DevOps). No domain should be weighted 0%, as gaps, especially in infrastructure or people domain, can cause failures. $TGMS$ highlights maturity gaps to justify test process improvements to executives.

Summary

To design test policy metrics, the TME can use the GQM approach (Basili, 1984). Metrics should be tied to follow-up actions like improving communication, test policy, or feedback loops. Metrics must be evaluated contextually, not in isolation, using three comparative lenses:

- Industry benchmarks such as (PractiTest, 2025), (Sogeti, 2024). (Jones, 2011) or (van Veenendaal, 2022) – e.g., to have EDR <10%, DDP of 85–95%, test policy compliance rate >90% in regulated sectors
- Similar projects to compare metrics before and after test policy implementation or across projects with similar size, stack, or test team composition
- Product families to make comparisons across shared architectures, product variants, or test teams using the same test policy to assess reuse and defect detection

The TME should normalize metrics to account for test team size, release cycle, product complexity, and test automation maturity. The TME should track trends via dashboards or time series charts to monitor adoption, fatigue, and correlations between effectiveness and satisfaction, and assess test policy maturity using models like TMMi or TPI NEXT® (Sogeti, 2009).

© International Software Testing Qualifications Board

7.3 Test Policy Improvement

A test policy is a living document that must evolve with the organization's changing business environment, risk profile, technology stack, and SDLC methodologies. As SDLC models continually shift and new challenges arise, (e.g., AI, security, and data privacy), the test policy provides a structured framework to establish, review, and continuously improve the organizational test strategy. This reflects a core principle of test management: the test process itself requires periodic review and refinement. The TME's role is to consolidate insights from retrospectives to guide updates to the test policy, so that changes are reflected in the test strategy and in the (master) test plans (see Section 1.3.1).

The formal mechanism for periodic test policy refinement is the managerial review. This structured, recurring process confirms the test policy remains suitable, adequate, and effective under current conditions. Such proactive monitoring is essential for advancing test process maturity, guiding the organization from ad hoc practices to a defined, continuously optimized approach. Managerial reviews provide a formal opportunity to assess the current state of testing, identify weaknesses, and create a roadmap for higher maturity, so that the test policy drives organizational quality rather than becoming stagnant documentation.

A sustainable test policy improvement program aims to maximize the value of testing by considering not just test results but also the broader costs of the test process. This includes financial and social costs, (e.g., human-centered impacts of how testing is organized, executed, and communicated). Factors that are impacted include cognitive load, stress, communication overhead, conflict, or resistance. Using a structured model like TMMi, organizations can assess current maturity, identify weaknesses, and create a roadmap for continuous improvement. TMMi guides the progression from ad hoc testing to defined, optimized processes, so that the test policy evolves as a driver of organizational quality and competitive advantage.

7.3.1 Coordinating a Managerial Review for Test Policy Assessment

The managerial review for test policy assessment is a formal process for evaluating the performance and ongoing relevance of the organizational test strategy. It is a structured meeting with a predefined agenda, rather than an informal discussion. The TME, in their role as a leader who directs test activities, is typically responsible for coordinating this review. This process is similar to a quality management review and is designed to assess whether the test policy meets its intended purpose.

A successful review relies on open communication and objective, discussion, focusing on systemic and process issues rather than individual blame. This approach uncovers root causes, encourages participation, and reduces resistance to change. A structured agenda keeps the review focused on improving cost-effectiveness and test management design, with the TME available for mediation if conflicts arise.

A thorough managerial review relies on a deliberate set of quantitative and qualitative inputs to give a holistic view of test policy performance. The TME organizes and presents these inputs to enable informed discussion, which can be categorized as follows:

- Test metrics: data on effectiveness, efficiency, and satisfaction
- Assessment, audit and compliance reports: findings from both internal and external audits, including assessments of internal controls and adherence to industry standards
- Feedback: qualitative input from stakeholders, customer satisfaction reports, and formal or informal feedback from the test team and other functional areas
- Change management data: details of major changes to the quality management system, the SDLC, or regulatory requirements affecting testing

© International Software Testing Qualifications Board

- Previous reviews: a summary of the results and a status update on follow-up actions from the previous managerial review

These inputs create a traceability chain linking daily testing to strategic outcomes. For example, a high EDR (effectiveness metric) paired with many customer support tickets (satisfaction metric) signals test policy issues. The TME presents this data as a narrative to highlight the need for test policy changes.

Metrics provide the essential data for assessing the test policy, but their true value lies in their interpretation. The analysis of these metrics must move beyond simple numbers to understand what they reveal about the underlying effectiveness, efficiency, and satisfaction delivered by the test policy.

Metrics reveal what is happening, but RCA (Söylemez, 2018) is the crucial next step for uncovering why a particular trend or problem exists. A successful RCA moves the discussion away from superficial symptoms to the fundamental, underlying causes that need to be addressed. Two powerful techniques for this purpose are the 5 Whys method (Gangidi, 2019) and the cause-and-effect diagram (Grady, 1996).

The 5 Whys method is an iterative technique where the test team repeatedly asks "Why?" to trace a defect from its symptom to its root cause, such as identifying reasons for a high EDR. For example:

- "Why did a critical defect escape to production?" – "Because a test case didn't cover it."
- "Why?" – "Because our test plans don't adequately address non-functional requirements."
- "Why?" – "Because the test policy doesn't mandate non-functional testing for this type of project."
- "Why?" – "Because the test policy hasn't been updated to reflect current product risks and system complexity."
- "Why?" – "Because there is no established governance or review cycle for maintaining and revising the test policy."

This structured approach can quickly reveal a test policy-level deficiency that requires correction.

Using the cause-and-effect diagram (see Section 6.3.1) supports that the analysis is comprehensive and considers all potential contributing factors, preventing the test team from focusing on a single, isolated cause. Using this diagram fosters a collaborative environment where test team members feel ownership over the findings, which in turn reduces resistance to the changes that will be implemented as a result.

7.3.2 Formulating Improvement Plans

Once the root causes have been identified, the TME must translate these findings into a concrete, actionable improvement plan for the test policy.

The test policy improvement plan can leverage TMMi or the TPI NEXT® models, which provide a strategic roadmap for elevating test maturity. An organization's TMMi assessment reveals its current level of maturity and the specific key practices required to reach the next level. The GQM approach (Basili, 1984) can then be used to define the specific, measurable goals for achieving those key practices. For example, a TMMi-identified need for a more defined process at Level 3 can be operationalized with a GQM plan to measure and improve the formalization of testware.

For implementing changes, the PDCA cycle (Deming, 1950) is a valuable iterative model. This approach involves planning a change on a small scale, executing the plan, studying the results, and then acting on the findings by either adopting, adapting, or abandoning the change. The PDCA cycle minimizes risk and fosters a culture of learning, allowing the organization to test and refine improvements before a broader implementation.

© International Software Testing Qualifications Board

Finally, all improvement goals must be articulated using the SMART framework to confirm they are clear and actionable. SMART goals are Specific, Measurable, Attainable, Relevant, and Timely. For instance, a GQM-derived metric on customer-reported defects could be transformed into the SMART goal: “Reduce the number of customer-reported, critical-severity defects by 50% within the next six months by implementing an enhanced approach to risk-based testing and updating the test level’s exit criteria.”

Communicating results to executive leadership: the language of business. Presenting test policy assessment results to senior leadership requires a deliberate shift from technical details to strategic business value. Executives, particularly at the C-level, are not interested in a long list of raw data or technical metrics. They want concise, high-level reports that focus on actionable items and translate directly into business outcomes. The TME’s role is to act as a bridge, translating data into the language of cost, revenue, and risk.

To effectively communicate, it is crucial to tell a compelling story with data. This means connecting test metrics to what matters most to the business, such as time-to-market delivery predictability, or defect reduction. Instead of test reporting an EDR of 10%, a more effective approach is to frame this as the cost of poor quality. This can include direct costs like lost revenue due to system downtime and indirect costs such as damage to brand reputation from a critical outage. For example, a C-level report might state, “An 8-minute reduction in build time from our CI/CD pipeline modernization saves an estimated $3 million in recovered engineering time annually”.

When presenting, use clear and easily understandable visuals. Management dashboards are powerful tools for condensing massive amounts of data into a single, intuitive view. These dashboards should highlight KPIs and avoid drowning the audience in data. A presentation can start with high-level, attention-grabbing statistics before drilling into empirical measurements. For added assurance and trust, metrics can be linked to external frameworks like the TMMi or a cyber security framework like those defined by NIST (NIST, 2024) or ISO/IEC standards (ISO/IEC 27000, 2026). Any detailed, granular data should be kept in an appendix and presented only if specifically requested.

Fostering a culture of continuous improvement. A test policy, as a strategic instrument, requires a commitment to continual improvement to remain relevant and effective. This is not a one-time project but a cultural discipline that must be embedded in the core of an organization. Continuous improvement should be a responsibility shared by all employees, not just the QA or the QC team, and training on the necessary tools and methodologies should be provided at all levels. Equally vital is a balanced approach that acknowledges current proficiencies alongside future goals, fostering a culture of positive reinforcement rather than one focused solely on corrective action.

Several frameworks can be used to guide this process systematically. The IDEAL model, (i.e., Initiating, Diagnosing, Establishing, Acting, and Learning), (Emam, 1999) serves as a robust roadmap for planning and implementing improvement actions. This structured approach helps an organization understand its current state and provides a framework for creating a solution, piloting it on a small scale, and refining it before broader implementation.

© International Software Testing Qualifications Board

8 Ethical Considerations - 105 minutes

Keywords

(none)

Domain specific keywords

(none)

Learning Objectives for Chapter 8:

8.1 Ethical Principles in Test Management

LO 8.1.1 (K2) Explain the fundamental ethical principles that guide test management decisions and describe their importance in maintaining professional integrity and mutual stakeholder trust

8.2 Identifying and Addressing Ethical Issues

LO 8.2.1 (K5) Evaluate a given set of behaviors under a given set of circumstances to identify possible ethical issues and determine appropriate corrective actions

© International Software Testing Qualifications Board

Introduction to Ethical Considerations

Introduction

Ethical considerations form the foundation of professional test management practice. The TME operates in complex organizational environments where they must balance competing interests, manage sensitive information, and make decisions that can significantly impact project outcomes, test team members, and organizational reputation. This chapter addresses the critical ethical dimensions of test management, providing frameworks for recognizing ethical dilemmas and guidance for maintaining professional integrity while achieving business objectives.

8.1 Ethical Principles in Test Management

The TME must adhere to fundamental ethical principles that guide professional conduct and decision-making. The ISTQB® code of ethics establishes core principles including public interest, client and employer interests, product integrity, professional judgment, responsibilities toward colleagues, management responsibilities, and continuous professional development. These principles establish the foundation for building mutual trust with stakeholders, maintaining professional credibility, and providing sustainable organizational development through ethical leadership practices.

Professionalism, supported by transparency and honesty, is a key aspect of ethical test management. In line with the ISTQB® code of ethics, the TME must provide accurate, objective test reporting of test results regardless of organizational pressure or, where possible, despite potential personal consequences, recognizing that real-world constraints may require escalation or use of formal conflict resolution channels. This includes honest communication about project risks, quality assessments, and test resource requirements. Transparency extends to stakeholders' communications, so that all parties have access to information defined in the test plan as necessary for stakeholders to make informed decisions. Objectivity and independence of testing require the TME to maintain professional judgment free from conflicts of interest or undue influence. This means resisting pressure to modify test results, suppress critical findings, or compromise quality standards to meet artificial deadlines. The TME has a mandatory professional obligation to advocate appropriate test practices, even when facing organizational resistance.

In line with ISO/IEC/IEEE 29119-3 standard, ethical principles relevant to test management, including professionalism, transparency, objectivity, independence of testing, confidentiality, and respect for colleagues, should be explicitly incorporated into the organization's test policy. This enables ethical considerations to be systematically embedded into formal test processes, regularly reviewed, and enforced alongside other organizational standards.

Confidentiality and professional competence. Confidentiality and data protection obligations, in the ethical context, focus on respecting privacy and safeguarding sensitive personal information, (e.g., not providing private details about team members), alongside information necessary for mutual stakeholder's trust. Broader confidentiality and data protection requirements, such as handling proprietary system details, customer data, and internal organizational information, may be governed by other organization standards, but the TME should confirm that ethical compliance aligns with these policies. The TME must establish clear protocols for information sharing, provide secure handling of test data, and respect intellectual property rights.

Professional competence demands that the TME maintain current knowledge, acknowledge limitations, and seek appropriate expertise when needed. This includes an honest assessment of test team capabilities and refusing assignments beyond professional competence without adequate support.

© International Software Testing Qualifications Board

Fair treatment and professional standards. Fair treatment requires equitable management of test team members, contractors, and other stakeholders without discrimination or favoritism. This encompasses fair performance evaluation, test resource allocation, and opportunity distribution. Professional standards must be consistently applied across all test team members and situations, so that ethical practices become embedded in organizational culture.

8.2 Identifying and Addressing Ethical Issues

The TME frequently encounters situations requiring ethical evaluation and decisive action. Common ethical dilemmas include pressure to falsify test results, requests to bypass test activities, conflicts between organizational interests and professional obligations, and challenges in test resource allocation or performance management. Recognition of these patterns helps TMEs prepare appropriate responses and maintain ethical standards under pressure.

Stakeholder pressure and confidentiality issues. Stakeholder pressure scenarios often involve demands to report favorable test results despite evidence of significant defects, or requests to reduce test scope to meet unrealistic deadlines. The TME must evaluate whether compliance would compromise professional integrity, public safety, or long-term organizational interests. Appropriate responses include documenting concerns, proposing alternative solutions, escalating to senior management, and maintaining detailed records of decisions and rationale.

Confidentiality dilemmas arise when the TME discovers information that could impact other projects, departments, or external parties. Acceptance criteria include assessing potential harm, determining disclosure obligations, and identifying appropriate channels for communication. The TME must balance confidentiality commitments with professional responsibilities to prevent harm or support organizational learning.

Test team management and external relationship ethics. Test team management ethics encompasses fair treatment of test team members, honest performance feedback, and appropriate handling of disciplinary issues. When evaluating behaviors, the TME should consider whether actions demonstrate respect for individual dignity, provide equal opportunities, and maintain professional boundaries. Corrective actions might include additional training, performance improvement plans, or formal disciplinary procedures following organizational policies.

Vendor and external relationship ethics require careful evaluation of potential conflicts of interest, fair procurement practices, and appropriate boundary management. The TME must assess whether relationships compromise objectivity, create unfair advantages, or violate organizational policies or contracts. Corrective measures include declaring conflicts, implementing oversight mechanisms, or removing individuals from decision-making processes.

Test resource allocation ethics involves promoting fair distribution of opportunities, equipment, and support among test team members and projects. Evaluation considerations include assessing whether decisions are based on legitimate business criteria, whether processes are transparent and consistent, and whether outcomes demonstrate equitable treatment.

Ethical AI use requires considering how AI systems affect people, organizations, and society. For the TME, this includes assessing: Does the AI treat all user groups equitably? Are AI decisions understandable and explainable? Is personal data being handled responsibly? Who is responsible if the AI fails or causes harm? Is the AI protected from misuse or malicious inputs? These principles set the baseline for evaluating behaviors.

© International Software Testing Qualifications Board

For example, a test manager instructs a test team to use an AI tool that predicts software defects. The AI's dataset is sourced mostly from legacy systems of a certain type, and some test team members notice it underreports defects in newer components. The possible ethical issues are as follows. The AI may systematically ignore certain kinds of defects, leading to unsafe or low-quality releases. Sole reliance on AI removes human judgment, which could miss context-specific risks. If the AI causes a defect in production, it is unclear who is responsible. The possible corrective actions include confirming that AI recommendations are reviewed by qualified test team members, who identify biases and gaps, retrain the AI with more representative data, and define who is accountable for AI-assisted decisions.

AI is an area of continual change and the TME must monitor industry trends to determine if AI is being used responsibly and for the best results.

Cultural influence on ethical norms. Different cultures have different ideas about what is considered ethical or acceptable behavior. In some cultures, preserving harmony and avoiding conflict may be prioritized over bluntly reporting defects. A tester might underreport defects or risks to maintain test team morale. In contrast, other cultures often value directness and transparency, expecting full disclosure of defects. In collectivist cultures, responsibility may be diffused across the test team. A test manager might avoid blaming individuals for missed defects, whereas in individualist cultures, accountability is often tied directly to specific roles. Cultural attitudes toward authority and rules can affect ethical decisions. In high power-distance cultures, a test manager might follow orders, even if it means cutting corners, while in low power-distance cultures, questioning management or advocating ethical standards might be encouraged.

The TME must be aware of the culture in which they work as well as other cultures with which they interact to determine the most acceptable and ethical approach to providing directions, resolving conflicts, and building a culture that respects and values feedback

Systematic approach to ethical problem-solving. When addressing identified ethical issues, the TME should follow systematic strategies including fact-gathering, stakeholders' consultation, test policy review, and documentation of decisions. Corrective actions must be proportionate, consistent with organizational values, and designed to prevent recurrence while maintaining test team morale and stakeholders' confidence. This systematic approach provides consistent application of ethical principles across diverse situations and organizational contexts.

- Identify the situation and stakeholders. What is happening? Who is involved? Who is affected?
- Gather relevant information. What are the potential risks and benefits? Are there applicable laws, regulations, or standards? What organization policies or professional codes apply?
- Identify ethical issues. Common principles to consider include:
    - Integrity/truthfulness: are you honest about outcomes?
    - Fairness: are any stakeholders treated unfairly or discriminated against?
    - Accountability: who is responsible for the outcomes?
    - Transparency: are decisions understandable and explainable?
    - Privacy and safety: are people's data or their safety at risk?
- Evaluate options and consequences. What are the short-term and long-term consequences? Who benefits or suffers from each option? Are there alternative actions that reduce harm?
- Make a decision. Choose the option that best aligns with ethical principles and minimizes harm. Ethical decisions often involve trade-offs, but should prioritize integrity, fairness, and safety.

Regardless of the environment, project pressures, and potential for political issues, the TME must always act in the most ethical manner possible. Objective evaluation of the situation will help to promote that all options are considered, and the best choice is made.

© International Software Testing Qualifications Board

9 List of Abbreviations

Abbreviation	Definition
AAPR	assessment or audit pass rate
ALM	application lifecycle management
CI/CD	continuous integration/continuous delivery
CL	centerline (on a control chart)
CoE	center of excellence
CoQ	cost of quality
CSAT	customer satisfaction score
DD	defect density
DDI	defect delay index
DDP	defect detection percentage
DRR	defect rejection rate
EDR	escaped defect rate
FTPR	feedback-to-test policy ratio
GQM	goal-question-metric
ITIL	Information Technology Infrastructure Library
KLOC	one thousand lines of code
KPI	key performance indicator
LCL	lower control limit (of a control chart)
MTTR	mean time to repair
NIST	National Institute of Standards and Technology
NPS	net promoter score
PDCA cycle	plan-do-check-act cycle
PMO	project management office
QA	quality assurance
QC	quality control
RCA	root cause analysis
ROI	return on investment
RRTC	reduction rate of test cycle time
RRTW	reuse rate of testware

© International Software Testing Qualifications Board

Abbreviation	Definition
RSI	requirements stability index
SDLC	software development lifecycle
SLA	service level agreement
SMART	S.M.A.R.T. goal methodology
TAEff	Test automation efficiency
TCE	test case effectiveness
TEE	test execution efficiency
TGMS	test governance maturity score
TME	test management expert
TMMi	Test Maturity Model integration
TOR	test overhead ratio
TOSI	test objective stability index
TPCR	test policy compliance rate
TRA	test result accuracy
UCL	upper control limit (of a control chart)
WBS	work breakdown structure

© International Software Testing Qualifications Board

10 References

10.1 Standards

ISO/IEC 25010 (2023-11), Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models

ISO/IEC 27000 (2026), Information security, cybersecurity and privacy protection – Information security management systems – Overview.

ISO/IEC/IEEE 29119-3 (2021), Software and systems engineering – Software testing – Part 3: Test documentation

ISO/IEC/IEEE 14764 (2022) Software engineering – Software life cycle processes – Maintenance

ISO/IEC 20246 (2017), Software and systems engineering – Work product reviews

ISO/IEC/IEEE 29119-1 (2022), Software and systems engineering – Software testing – Part 1: General Concepts

ISO/IEC/IEEE 29119-2 (2021), Software and systems engineering – Software testing – Part 2: Test processes

ISO/IEC/IEEE 29119-4 (2021), Software and systems engineering-Software testing – Part 4: Test techniques

ISO/IEC/IEEE 29119-6 (2020), Software and systems engineering-Software testing – Part 6: Guideline for the use of agile projects

ISO 31000 (2018), Risk management – Principles and guideline

10.2 ISTQB® Documentation

ISTQB-ATT ISTQB Certified Tester Agile Technical Tester

ISTQB-ATLaS ISTQB Certified Tester Agile Test Leadership at Scale

ISTQB-FL ISTQB Certified Tester Foundation Level Syllabus v4.0.1

ISTQB-ITP ISTQB Certified Tester Expert Level Improving the Test Process v1.0

ISTQB-TM ISTQB Certified Tester Advanced Level Test Management Syllabus v3.0

ISTQB-TTA ISTQB Certified Tester Advanced Level Technical Test Analyst v4.0

ISTQB-TA ISTQB Certified Tester Advanced Level Test Analyst v4.0

The authors Ina Schieferdecker, Arnika Hryszko, Adam Roman, Armin Born, Anja Kribernegg, Erhardt Wunderlich, Jaroslaw Hryszko, Renzo Cerquozzi.

Working Group team members: Benjamin Timmermans, Anoop Rayudu, Erik van Veenendaal, Reifa Tangon, Samuel Ouko, Webstar Ongaki Okerio.

© International Software Testing Qualifications Board

10.3 Glossary References

ISTQB Glossary, https://glossary.istqb.org/

Agile Alliance, https://www.agilealliance.org/agile-practice-guide/

IEEE/Pascal for software engineering, https://pascal.computer.org/

IREB-CPRE for Requirements Engineering, https://cpre.ireb.org/en/downloads-and-resources/downloads#cpre-glossary

10.4 Books, Articles and Web Pages

Khalid Alemerien, Magel Kenneth, Examining the effectiveness of testing coverage tools: An empirical study, International Journal of Software Engineering and its Applications 8(5), pp. 139-162, 2014, doi 10.14257/ijseia.2014.8.5.12.

Victor R. Basili, David M. Weiss, A Methodology for Collecting Valid Software Engineering Data, IEEE Transactions on Software Engineering, vol. 10(6), pp. 728-738, 1984, doi: 10.1109/TSE.1984.5010301.

Rex Black, Managing the Testing Process. Wiley, 2009.

Philip Crosby, Quality Is Free: The Art of Making Quality Certain, McGraw-Hill Education – Europe, 1978.

Edwards W. Deming, Elementary Principles of the Statistical Control of Quality. JUSE, 1950.

Khaled El Emam, Nazim H. Madhavji (eds), Elements of Software Process Assessment & Improvement. IEEE Computer Society Press, 1999.

Isabel Evans, Achieving Software Quality through Teamwork. Artech House, 2004.

Norman Fenton, James Bieman, Software Metrics: A Rigorous and Practical Approach, Third Edition, CRC Press, 2020.

Prashant Gangidi, A systematic approach to root cause analysis using 3×5 why's technique, International Journal of Lean Six Sigma 10(1), 2019, pp. 295-310.

Robert B. Grady, Software failure analysis for high-return process improvement decisions, Hewlett Packard Journal 47, 1996, pp. 15-24.

Frederick Herzberg, Bernard Mausner, Barbara B. Snyderman, The Motivation to Work, 2nd ed. John Wiley, 1959. ISBN 0471373893.

Ferenc Horváth et al., Code coverage differences of Java bytecode and source code instrumentation tools. Software Quality Journal 27, 2019, pp. 79-123, https://doi.org/10.1007/s11219-017-9389-z.

Watts S. Humphrey, Managing the Software Process. Addison-Wesley, 1989.

Kaoru Ishikawa, What Is Total Quality Control? The Japanese Way. Prentice Hall Business Classics, 1985.

ISTQB webpage, www.istqb.org, last access: 2026.

Capers Jones, Olivier Bonsignour, The Economics of Software Quality. Addison-Wesley, 2011.

Yasutaka Kamei et al., A large-scale empirical study of just-in-time quality assurance, IEEE Transactions on Software Engineering 39(6), pp. 757-773, 2013, doi: 10.1109/TSE.2012.70.

© International Software Testing Qualifications Board

Stephen Kan, Metrics and Models in Software Quality Engineering, 2nd ed. Addison-Wesley, 2002.

Rensis Likert, A technique for the measurement of attitudes. Archives of Psychology, 22 140, 55, 1932.

Erin Meyer, The Culture Map, PublicAffairs 2016.

NIST, The NIST Cybersecurity Framework (CSF) 2.0, https://doi.org/10.6028/NIST.CSWP.29, 2024.

PractiTest. The 2025 State of Testing™ Report (annual), https://www.practitest.com/state-of-testing/, 2025.

Sogeti, World Quality Report 2024-25 (annual), https://www.sogeti.com/research-and-insight/world-quality-report-2024-25/, 2024.

Sogeti. TPI NEXT® – Business Driven Test Process Improvement (1st ed.). 2009.

Mehmet Söylemez, Tarhan Ayca, Challenges of software process and product quality improvement: catalyzing defect root-cause investigation by process enactment data analysis, Software Quality Journal 26(2), 2018, pp. 779-807.

Bruce W. Tuckman, Developmental sequence in small groups, Psychological Bulletin 63(6), 384-399, 1965.

Erik van Veenendaal, Jan J. Cannegieter, Test Maturity Model integration (TMMi). Results of the first TMMi benchmark – where are we today? 2022.

TMMi Foundation. TMMi model specification (reference model), release 1.3. TMMi Foundation. https://www.tmmi.org/. 2022.

Web1, https://en.wikipedia.org/wiki/Poisson_distribution

Linda Westfall, The Certified Software Quality Engineer Handbook. ASQ Press, 2009

10.4.1 Books, Articles and Web Pages (not directly referenced in this Syllabus)

Rex Black, Critical Testing Processes: Plan, Prepare, Perform, Perfect. Addison-Wesley, 2003

Barry Boehm, Richard Turner, Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley, 2003.

Fred Brooks, The Mythical Man-Month: Essays on Software Engineering, 2nd ed. Addison-Wesley, 1995.

Lee Copeland, When helping doesn't help. SQE Magazine, 2001.

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

Tom DeMarco, Timothy Lister, Peopleware: Productive Projects and Teams, 2nd ed. Dorset House, 1999.

Tom DeMarco, Timothy Lister, Waltzing with Bears: Managing Risk on Software Projects. Dorset House, 2003.

Esther Derby, Diana Larsen, Agile Retrospectives-Making Good Teams Great. The Pragmatic Bookshelf, 2006.

Peter Drucker, The Practice of Management. Harper, 2006.

Patricia Ensworth, The Accidental Project Manager: Surviving the Transition from Techie to Manager. Wiley, 2001.

Robert Galen, Software Endgames: Eliminating Defects, Controlling Change, And the Countdown to On-time Delivery. Dorset House, 2004.

© International Software Testing Qualifications Board

Robert Glass, Facts and Fallacies of Software Engineering. Addison-Wesley, 2002.

Dorothy Graham, Mark Fewster, Software Test Automation. Addison-Wesley, 1999.

ITIL, Information Technology Infrastructure Library, https://www.itil.com, last access: 2026.

Capers Jones, Assessment and Control of Software Risks. Prentice Hall, 1993.

Capers Jones, Estimating Software Costs: Bringing Realism to Estimating. McGraw-Hill, 2007.

Capers Jones, Applied Software Measurement: Global Analysis of Productivity and Quality, 3rd ed. McGraw-Hill Osborne Media, 2008.

Nancy Leveson, Safeware: System Safety and Computers. Addison-Wesley, 1995.

Steve McConnell, Software Project Survival Guide. Microsoft Press, 1997.

Steve McConnell, After the Gold Rush: Creating a True Profession of Software Engineering. Microsoft Press, 1999.

Judy McKay, Managing the Test People. Rocky Nook, 2007.

Isabel B. Myers, Understanding Personality Type. Davies-Black Publishing, 1995.

PMI (Project Management Institute), A Guide to the Project Management Body of Knowledge (PMBOK® Guide), 4th ed. Project Management Institute, 2008.

Tim Riley, Adam Goucher, Beautiful Testing. O'Reilly, 2009.

Johanna Rothman, Hiring the Best Knowledge Workers, Techies and Nerds: The Secrets & Science of Hiring Technical People. Dorset House, 2004.

Johanna Rothman, Manage It!: Your Guide to Modern, Pragmatic Project Management. Pragmatic Bookshelf 2007.

Yogesh Singh, Metrics and Models in Software Testing. In: Software Testing. Cambridge University Press, 2011, pp. 420-452.

Mark Utting, Bruno Legeard, Practical Model-Based Testing: A Tools Approach. Morgan Kaufmann, 2006.

Erik van Veenendaal, Test maturity model integration (TMMi): Guideline for test process improvement, release 1.2. TMMi Foundation, 2018.

James Whittaker, How to Break Software: A Practical Guide to Testing. Addison Wesley, 2002.

Karl Wiegers, Creating a Software Engineering Culture. Dorset House, 1996.

Edward Yourdon, Death March, 2nd ed. Prentice Hall, 2003.

The previous references point to information available on the Internet and elsewhere. Even though those references were checked at the time of publication of this syllabus, the ISTQB® cannot be held responsible if the references are not available anymore.

© International Software Testing Qualifications Board

11 Trademarks

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

TMMi® is a registered trademark of TMMi Foundation.

TPI NEXT® is a registered trademarks of Sogeti, The Netherlands.

© International Software Testing Qualifications Board

12 Appendix A – Learning Objectives/Cognitive Level of Knowledge

Use the Bloom's Taxonomy when writing Learning Objectives. Refer to https://cft.vanderbilt.edu/guides-sub-pages/blooms-taxonomy/

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

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

In general, Learning Objective and the K levels and their time allocated for the section should match:

- K1: 15 minutes or less
- K2: 15 minutes
- K3: 60 minutes
- K4: 75 minutes
- K5: 90 minutes

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.	

© International Software Testing Qualifications Board

Examples	Notes
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 / or process .
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. Typical application is to analyze a document, software or project situation and propose appropriate actions to solve a problem or 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 test 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.

© International Software Testing Qualifications Board

Level 5: Evaluate (K5)

The candidate may make judgments based on criteria and standards. He detects inconsistencies or fallacies within a process or product, determines whether a process or product has internal consistency and detects the effectiveness of a procedure as it is being implemented (e.g., determine if a scientist's conclusions follow from observed data.)

Action verbs: Assess, critique, evaluate, recommend

Examples	Notes
Assess a test organization using either TPI NEXT® or TMMi.	
Critique the appropriateness of the test activities in an organization for the given context.	
Evaluate an organization to determine the options for proper placement of the test team.	Should be restricted to the evaluation, not including the resulting solution (which should be K6)
Recommend measures to create acceptance of the changes by the people involved.	

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.

© International Software Testing Qualifications Board

13 Appendix B – Business Outcomes traceability matrix with Learning Objectives

This section lists the traceability between Business Outcomes and the Learning Objectives of CTEL-TM

Business Outcomes: CTEL-TM			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10
ELTM-BO1	On organizational/strategic level, promote and position testing for the production of quality software		5									
ELTM-BO2	Define, implement, evaluate, and improve the test policy and test strategy suitable for the business objectives and support commitments within the organization			4								
ELTM-BO3	Align test management with other roles or management areas across the organization and outside the organization				6							
ELTM-BO4	Lead the test management within an organization, project or program to identify and manage critical success factors and risks with management commitment at strategic level					4						
ELTM-BO5	Understand and manage the human themes associated with test-project management and implement necessary changes.						8					
ELTM-BO6	Set up a metrics tracking and test reporting for test monitoring and test control reporting process for test monitoring and control, interpret the test results, make business decisions, and communicate to internal and external stakeholders.							11				

© International Software Testing Qualifications Board

Business Outcomes: CTEL-TM			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10
ELTM-BO7	Assess the current status of testing, implement improvements, and manage the introduction of change.								4			
ELTM-B08	Adapt test management to address different SDLC models or project types									1		
ELTM-B09	Maximize productivity by selecting the appropriate mix of personnel considering skill levels, staffing options, location of test resources and management structure										3	
ELTM-B10	Introduce innovations to test management within organizations, which among others include roles, skills, test techniques, technologies, tools and organizational structure											3

Unique LO	Learning Objective	K-Level										
1	Test Policies, Test Strategies and Test Objectives											
1.1	Quality Goals and Test Policy											
1.1.1	Analyze the organizational quality goals to determine a test policy consistency with these goals	K4	x									
1.1.2	Evaluate test objectives that might have long term effects on the software quality and on organizational, economic, and technical dependencies	K5		x								
1.2	Organizational-level Test Strategies											
1.2.1	Analyze project context to determine which individual strategies, or	K4		x								

© International Software Testing Qualifications Board

Unique LO	Learning Objective	K-Level										
	blended strategies, are most likely to succeed											
1.2.2	Analyze test strategy failures to identify causes and contributing factors	K4		x								
1.2.3	Recommend comprehensive test strategy documents that enable success factors and mitigate failure risks	K5		x								
1.3	Alignment of Test Policy and Test Strategy with the Organization											
1.3.1	Evaluate test policies and strategies for organizational alignment, considering short-term and long-term perspectives across processes, people, tools, systems, and techniques	K5		x								
2	**Managing the Test Team**											
2.1	Building the Test Team											
2.1.1	Evaluate candidate suitability for strategic testing roles, considering emerging technologies, DEI principles, and adaptability to DevOps/test automation .	K5									x	
2.1.2	Outline the major points of a training program for new people	K4									x	
2.1.3	Explain considerations that must be made when terminating an employee or contractor	K2									x	
2.2	Developing the Test Team											
2.2.1	Evaluate leadership and communication approaches using strategic stakeholder management and metrics aligned with organizational goals	K5					x				x	

© International Software Testing Qualifications Board

Unique LO	Learning Objective	K-Level										
2.2.2	Explain the importance of an individual understanding their role and responsibilities within a test team	K2					x					
2.2.3	Analyze the tester skills to identify areas for skill development and learning including training and mentoring	K4					x					
2.2.4	Apply performance reviews that address both high performers and poor performers, setting appropriate goals and development plans for each	K3					x					
2.3	Leading the Test Team											
2.3.1	Evaluate the information and guidance needed by the test team, adjust management and leadership approach based on circumstances to determine the most effective method of communication	K5					x					
2.3.2	Analyze team dynamics and organizational structure to design team-building activities that foster trust, loyalty, and communication across distributed teams	K4					x					
2.3.3	Explain the challenges of motivating a test team, both short term and long term	K2					x					
3	**Managing Across the Organization**											
3.1	Advocating the Test Team											
3.1.1	Apply strategies to promote, advocate for, and defend the test team	K3	x									
3.1.2	Analyze the quantitative and qualitative benefits of the test team using defined metrics and proxies to	K4	x									

© International Software Testing Qualifications Board

Unique LO	Learning Objective	K-Level										
	build a business case for a given stakeholder group											
3.2	Placement of the Test Team											
3.2.1	Evaluate a test team's placement within the organization against independence of testing, information flow, test resource availability, and organizational culture to recommend a placement based on the evaluation	K5	x									
3.2.2	Analyze the implications of different test team placement options with respect to test effectiveness, stakeholder collaboration, and organizational quality maturity	K4	x									
3.3	Building Strategic Relationships											
3.3.1	Analyze complex project environments to identify key relationships and develop strategies for creating and building effective partnerships with managers and teams	K4			x							
3.4	Stakeholder Communication											
3.4.1	Use effective communication with testing stakeholders, including non-test team, about critical issues related to testing	K3			x							
3.5	Integrating Tools Across the Organization											
3.5.1	Explain the factors to consider when dealing with multi-use tools	K2										x
3.5.2	Analyze a proposed change to a multi-use tool and assess the impact on the test organization	K4										x
4	**Managing External Relationships**											

© International Software Testing Qualifications Board

Unique LO	Learning Objective	K-Level										
4.1	Types of External Relationships											
4.1.1	Give examples of the third-party relationships regarding testing responsibilities	K2			x							
4.2	Contractual Issues											
4.2.1	Prepare a suitable SLA for a given project with an external vendor	K3			x							
4.3	Communication Strategies for Third-Parties											
4.3.1	Select appropriate communication strategies for a given third-party engagement	K4			x							
4.4	Integrating from External Sources											
4.4.1	Analyze a given project to determine the appropriate level of internal test involvement in combination with a third party	K4		x								
4.5	Merging Third-Party Test Strategies											
4.5.1	Summarize the items that must be considered when merging test strategies with a third-party organization	K2		x								
4.6	Verifying Third-Party Quality											
4.6.1	Assess the quality of external relationship	K5			x							
5	**Project Management Essentials**											
5.1	Test Estimation and Defect Prediction											
5.1.1	Use standard deviation to interpret the test estimation	K3				x						
5.1.2	Select a defect prediction model	K4				x						

© International Software Testing Qualifications Board

Unique LO	Learning Objective	K-Level										
5.2	Project Risk Management											
5.2.1	Analyze trade-offs between quality, test schedule, budget, and test scope of a test project	K4				x						
5.2.2	Apply project risk analysis	K3				x						
5.3	Change Management											
5.3.1	Summarize the role of TME in the change management process	K2							x			
5.3.2	Illustrate impact analysis for project-wide changes on testing	K4							x			
5.4	Test Management Considerations for Project Types											
5.4.1	Apply a given test management approach to a specific project type	K3								x		
6	**Test Monitoring and Test Control**											
6.1	Metrics and Measurement											
6.1.1	Prepare a set of the most effective metrics to be used to monitor the testing	K3						x				
6.1.2	Evaluate a test measurement and test reporting process	K5						x				
6.2	Test Reporting											
6.2.1	Distinguish between internal and external reporting	K2						x				
6.2.2	Analyze, effectiveness, efficiency, and satisfaction metrics for the test process, identifying opportunities for improvement	K3						x				
6.2.3	Select the proper level and media for providing test progress reports	K4						x				
6.3	Interpreting the Test Results											

© International Software Testing Qualifications Board

Unique LO	Learning Objective	K-Level										
6.3.1	Apply basic quality control techniques	K3						x				
6.3.2	Analyze a given test condition based on available data	K4						x				
6.3.3	Apply test control actions to resolve deviations from the test plan	K3						x				
7	**Evaluating Effectiveness, Efficiency and Satisfaction and Improving Test Policy**											
7.1	Effectiveness, Efficiency and Satisfaction Metrics for the Test Process											
7.1.1	Explain the purpose of tracking effectiveness, efficiency and satisfaction metrics	K2						x				
7.1.2	Analyze, effectiveness, efficiency, and satisfaction metrics for the test process, identifying opportunities for improvement	K4						x				
7.2	Effectiveness, Efficiency and Satisfaction Metrics for the Test Policy											
7.2.1	Evaluate the effectiveness, efficiency, and satisfaction metrics for the test policy as means to monitor its quality	K5						x				
7.3	Test Policy Improvement											
7.3.1	Use managerial reviews for policy assessment	K3							x			x
7.3.2	Analyze the results of a managerial review to formulate plans for test policy improvement	K4							x			
8	**Ethical Considerations**											

© International Software Testing Qualifications Board

Unique LO	Learning Objective	K-Level										
8.1.	Ethical Principles in Test Management											
8.1.1	Explain the fundamental ethical principles that guide test management decisions and describe their importance in maintaining professional integrity and stakeholder trust	K2					x					
8.2	Identifying and Addressing Ethical Issues											
8.2.1	Evaluate a given set of behaviors under a given set of circumstances to identify possible ethical issues and determine appropriate corrective actions	K5					x					

© International Software Testing Qualifications Board

14 Appendix C – Release Notes

ISTQB® Expert Level Test management is a major update and rewrite of release 2011. For this reason, it is not possible to give detailed changes per chapter and section. However, a summary of principal changes is provided here.

In this version all learning objectives have been edited to make them atomic, and to create one-to-one traceability from learning objectives to content and not having content without also having a learning objective.

A lot of definitions have been removed because by the time they have been taken by Foundation level, advanced Test manager and other syllabi.

The goal is to make this version easier to read, give added values with respect to the Foundation level, advanced levels, and specialized Syllabi.

The Syllabus has been completely rewritten, based on existing ELTM v1.0 and checking the definitions on (ISTQB-FL), (ISTQB-TM), (ISTQB-TTA), (ISTQB-ATLaS), and others.

Additional changes made to the 2018 Foundation Syllabus are:

1. ISO/IEC/IEEE 29119 is now used for reference instead of IEEE Standard 829.
2. ISO/IEC 25010 is now used for reference instead of ISO 9126.
3. ISO/IEC 20246 is now used for reference instead of IEEE 1028.

The Code of Ethics has been moved from chapter one to the ISTQB.ORG web page.

© International Software Testing Qualifications Board

15 Appendix D – Information for training providers

15.1 Training Time

Each chapter in the syllabus is assigned an allocated time in minutes. The purpose of this is to give guidance on the relative proportion of time to be allocated to each section of an accredited course and to give an approximate minimum time for teaching each section.

Training providers may spend more time than is indicated and candidates may spend more time again in reading and research. A course curriculum does not have to follow the same order as the syllabus. It is not required to conduct the course in one continuous block of time.

The table below provides a guideline for class instruction and exercise timing for each chapter and shows the timing for exercises which may be conducted in the workplace. All times are shown in minutes (Hours). Note that exercises in the workplace may also be conducted as part of the class given by the training provider (see below).

The duration of class instruction, Exercise have been calculated assuming a net training day of 6.5 hours

Chapter	Chapter title	Class instruction and exercises (min.)	Exercises in the workplace (min.)	Total (min.)
1	Test Policies, Test Strategies and Test Objectives	105	390	495
2	Managing the Test Team	210	390	600
3	Managing Across the Organization	525	0	510
4	Managing External Relationships	200	130	330
5	Project Management Essentials	420	0	420
6	Test Monitoring and Control Test Monitoring and Test Control	365	130	495
7	Evaluating Effectiveness, Efficiency and Satisfaction and Improving Test Policy	170	145	315
8	Ethical Considerations	50	55	105
	Total minutes (hours) recommendation	**2045** (**34h05'**)	**1240 (20h40')**	**3270** (**54h45'**)

The following table shows the total course times in days, based on an average of seven hours per workday.

Course element	Days
Class instruction and exercises	5
Exercises in the workplace/homework	3
Total:	8

© International Software Testing Qualifications Board

15.2 Guideline for Practical Exercises in the Workplace

Practical work should be included for all aspects where candidates are expected to apply their knowledge (i.e., learning objectives of K-Levels). The lectures and exercises should be based on the learning objectives and the description of the topics in the content of the syllabus.

Certain learning objectives may be achieved by conducting practical exercises in the workplace or at home. The ISTQB® web site (ISTQB, 2026), Expert Level section, could help to support the Training Providers for creating exercises for the trainees.

© International Software Testing Qualifications Board

16 Index

analytical test strategy 15

cause-and-effect diagram 72, 80, 93

change management 14, 21, 50, 52, 62, 69, 70, 115

consultative test strategy 15

continuous integration/continuous delivery 40

control chart 72, 79, 99, 100

cost of quality 17, 36, 62, 99

cost-benefit analysis 17, 42, 62, 66

dashboard 72

defect density 80, 84, 99

defect prediction 14, 62, 64, 65, 114

DEI principles 24, 25, 27, 111

entry criteria 54, 56, 59, 60, 61, 89

escaped defect rate 37, 84, 99

exit criteria 46, 54, 56, 59, 60, 61, 70, 81, 82, 90, 94

goal-question-metric 18, 72, 99

impact analysis, 62

measurement 14, 18, 20, 22, 23, 42, 63, 72, 73, 74, 75, 76, 103, 115

methodical test strategy 15

metric 54, 72, 84

model-based test strategy 15

mutation analysis 62

organizational quality goal 15

outlier 62, 63, 64

Pareto chart 72, 80

process-compliant test strategy 15

project risk 14, 62, 67, 115

psychological safety 24, 25, 34, 36, 38, 39, 40, 45, 52

quality assurance 12, 99, 102

quality control 14, 72, 99, 116

quality enablement 36, 37, 38, 49, 50

reactive test strategy 15, 19

regression-averse test strategy 15, 19

risk analysis 14, 62, 67, 68, 115

S.M.A.R.T. goal methodology 24, 100

satisfaction 14, 22, 31, 34, 37, 38, 39, 40, 41, 49, 66, 74, 84, 85, 88, 91, 92, 93, 99, 115, 116

scatter plot 72, 80

service level agreement 54, 100

standard deviation 14, 62, 63, 64, 75, 79, 80, 114

test control 10, 70, 72, 80, 81, 83, 109, 116

test estimation 62, 63, 64, 114

test framework 84

test monitoring 10, 14, 20, 21, 23, 42, 59, 70, 72, 85, 109

test objective 15, 89, 100

test policy 10, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 25, 27, 28, 33, 84, 85, 86, 89, 90, 91, 92, 93, 94, 96, 98, 99, 100, 109, 110, 116

test progress report 72, 78

test reporting 10, 14, 22, 29, 30, 32, 33, 42, 43, 55, 59, 70, 72, 73, 74, 75, 76, 77, 78, 79, 81, 85, 94, 96, 109, 115

test strategy 10, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 42, 54, 56, 58, 59, 60, 69, 71, 76, 82, 89, 91, 92, 107, 109, 111

© International Software Testing Qualifications Board

test team advocacy 36
third-party 14, 47, 54, 55, 56, 57, 59, 71, 114
time-to-learn 36, 38, 40, 52
value-based prioritization 62, 66, 67

© International Software Testing Qualifications Board