TMMi® FOUNDATION

Test Maturity Model integration (TMMi®)

Guidelines for Test Process Improvement

Version 2.0

#SuccessWithTMMi

AUTHOR: ERIK VAN VEENENDAAL

www.tmmi.org

The Test Maturity Model integration (TMMi) has been developed by the TMMi Foundation as a guideline and reference framework for test process improvement. Using TMMi, organizations can have their test processes objectively evaluated by accredited assessors, improve their testing practices, and achieve formal certification when they comply with the prescribed requirements.

This document provides the complete and detailed description of TMMi version 2.0. The document is organized into three main parts.

Part One – Overview and Framework Structure

Part One consists of three chapters:

- Chapter 1 – "Test Maturity Model integration"
  Describes the background and history of TMMi, the primary sources used during its development, and the scope of the model.
- Chapter 2 – "TMMi Maturity Levels"
  Provides an overview of the five maturity levels, including descriptions of the associated process areas.
- Chapter 3 – "Structure of the TMMi"
  Explains the components of the TMMi process areas and presents an overview of the relationship between TMMi process areas and CMMI practice areas.

Part Two – The Process Areas

Part Two, entitled "The Process Areas", consists of sixteen sections, one for each TMMi process area. The sections are organized according to maturity level. Each section includes descriptions of the goals, practices, subpractices, and supporting examples.

Part Three – Informational Resources

Part Three contains six informational resources:

- Annex A – Artificial Intelligence and TMMi
- Annex B – Quality Engineering Mapping
- Annex C – The Improvement Process
- Annex D – TMMi Level 1 Practices
- Glossary
- References

An overview of the major changes introduced in TMMi version 2.0, together with the rationale for these changes compared to earlier versions, is provided in a separate document [TMMiReleaseV2].

Contributors

Rogier Ammerlaan (The Netherlands)
Katalin Balla (Hungary)
Suresh Chandra Bose (USA)
Clive Bates (UK)
Adrian Howes (UK)
Mattijs Kemmink (The Netherlands)
Poonam Jain (India)
Qin Liu (China)
Alfonsina Morgavi (Argentina)
Matthias Rasking (Germany)
Eric Riou du Cosquer (France)
Ella Shang (China)
Miaomiao Tang (China)
Çiçek Tuna (Germany)
Erik van Veenendaal (The Netherlands)
Kelly Ye (Australia)

Revisions

This section summarizes the key revisions between releases of this document.

This section is provided for information only.

Release	Revision Notes
V2.0	Initial TMMi version 2.0 document, released June 2026

Table of Contents

1 Test Maturity Model integration (TMMi®)

1.1 Introduction

Organizations face tough business challenges on a daily basis, e.g., decrease time-to-market, requirements for higher levels of quality and reliability, and reduce costs. Systems, in which most often software is a dominant factor, have become highly complex and therefore challenging to build. New approaches, methods, techniques, and tools have become available to support software development and maintenance tasks. Because systems play such an important role in today's society, both economically and socially, there is pressure for the software engineering discipline to focus on quality issues. Inadequate levels of software quality are no longer acceptable, because software failures can result in catastrophic consequences. In this context the importance of the testing discipline, as one of the quality measures that can be taken, has grown significantly. Today, testing is a key activity that directly influences not only the product quality but also the 'performance' of the entire development and manufacturing process.

For decades, the software industry has invested substantial effort to improve the quality of its products. This has been a difficult job since the size and complexity of software has increased rapidly, while in parallel customers and users have become more and more demanding. Despite encouraging results from various quality improvement approaches, the software industry is still experiencing defects in released products. To improve product quality, the software industry has often focused on improving its development processes. A guideline that has been used repeatedly to improve the development processes is the Capability Maturity Model Integration (CMMI). Despite the fact that testing regularly accounts for up to 30-40% of the total project costs, only limited attention is given to testing in the various software process improvement models such as the CMMI. As an answer, the testing community has created its own process improvement models. This document describes version 2.0 of the Test Maturity Model Integration (TMMi). The TMMi is a detailed model for test process improvement, initially released by the TMMi Foundation in 2012, and today the world-leading model for test process improvement [Garousi].

1.2 Background

The TMMi framework has been developed by the TMMi Foundation as a guideline and reference framework for test process improvement, addressing those issues important to test managers, test engineers, developers and software quality professionals. With TMMi, testing is defined in its broadest sense to encompass all software product quality related activities. TMMi uses the concept of maturity levels for process evaluation and improvement. Furthermore, process areas, goals and practices are identified. Applying the TMMi maturity criteria will improve the test process and has shown to have a positive impact on product quality, test engineering productivity, and cycle-time effort. TMMi has been developed to support organizations in evaluating and improving their test processes. Within the TMMi, testing evolves from a chaotic, ill-defined process with a lack of resources, tools and well-trained testers, to a mature and controlled process that has defect prevention as its main objective.

Practical experiences are very positive and show that TMMi supports the process of establishing a more effective and efficient test process. The world-wide TMMi user survey revealed that 94% of TMMi users are experiencing benefits related to product quality e.g., a reduction of product risks, and 78% of them in test efficiency, e.g., increased testing productivity [Survey23]. With TMMi, testing becomes a profession and a fully integrated part of the development process. As stated, the focus of testing ultimately changes from defect detection to defect prevention.

1.3 Sources

1.3.1 TMMi development

The initial development of the TMMi used the TMM framework, developed by the Illinois Institute of Technology, as one of its major sources [Burnstein]. In addition to the TMM, the work was largely guided by the Capability Maturity Model Integration (CMMI). Whereas early versions of the CMMI [CMMI] had both a staged and continuous representation, the TMMi has been developed as a staged model. A staged model uses predefined sets of process areas to define an improvement path for an organization. This improvement path is described by a model component called maturity level. A maturity level is a well-defined evolutionary plateau towards achieving improved organizational processes.

Another source used in the TMMi development are international testing standards, especially those from ISO. Achieving TMMi maturity level 3 also ensures full coverage for the processes outlined in ISO/IEC 29119-2 Software Testing and ISO/IEC 20246 Reviews [Veenendaal16]. The testing terminology used in the TMMi is derived from the ISTQB Standard Glossary of terms used in Software Testing [ISTQB]. Furthermore, various industry best practices and testing surveys contributed to the TMMi development, providing it with its necessary empirical foundation and required level of practicality. These illustrate the current best practices in the IT industry, and have allowed the developers of the TMMi framework to extract realistic benchmarks against which to evaluate and improve testing practice

1.3.2 TMMi version 2

For TMMi version 2, practical experiences using the TMMi have been a major source of input. TMMi practitioners, e.g., test process improvers and TMMi (lead) assessors, and the TMMi Local Chapters, representing the local TMMi training providers and TMMi assessment service providers, have all been asked to provide input and review the plans for TMMi version 2.

In addition, new developments surrounding the CMMI model have been studied for consideration and possible uptake. Examples are more focused on business value and the removal of generic goals and generic practices from process areas. Also, since the initial release of TMMi, other lifecycle models have become much more popular. In TMMi version 2, Agile and DevOps have been taken into account, which resulted among others in adding and removing (sub)practices, and adding specific examples. In general, lean as a mindset and approach has also been a driver during the development of TMMi version 2. An overview of the major changes for TMMi version 2 and their rationale is provided in a separate document [TMMiReleaseV2].

1.4 Scope

1.4.1 Test Levels and Test Types

Whereas some models for test process improvement focus mainly on higher test levels, the TMMi has both lower test levels (e.g., component test, integration test) and higher test levels (e.g., system test, acceptance test) within scope. The TMMi also has the various test types in scope. It covers functional testing, regression testing, non-functional testing, e.g., performance testing and security testing, and maintenance testing. TMMi also covers both dynamic testing (testing by running test cases and executing the software) and static testing, e.g., reviews and static analysis. The world-wide TMMi user survey [Survey23] shows that TMMi is widely used across all test levels and test types.

1.4.2 Agile and DevOps

TMMi is intended to be lifecycle-independent and can therefore be used with sequential, Agile and DevOps lifecycle models. To support those who are applying TMMi in an Agile or DevOps context, the TMMi Foundation has published highly detailed supporting guidelines [TMMiAgile], [TMMiDevOps]. These guidelines explain how TMMi can be used and applied beneficially in an Agile or DevOps context. Alternatives are provided to traditional testing approaches that implement TMMi practices while maintaining and perhaps even increasing agility. Each TMMi process area, along with its goals, is discussed in turn. It is explained how these relate to Agile or DevOps and what the practices will typically look like. If a practice has no added value in an Agile or DevOps context and is not expected to be performed, this is also explicitly stated.

1.4.3 Test Automation

TMMi is a framework to support test *process* improvement and although test automation and test tools are referenced in the model at several points, it is not a framework for the implementation of test automation. In this context the TMMi Foundation has adopted the principles of a highly popular framework in the business domain: the People, Process and Technology framework (also known as the PPT framework) [Leavitt]. It refers to, and exhibits, how the balance of people, processes, and technologies drives successful organizational change, improvements, and re-engineering. Technology provides the tools that people can use to implement and perform the process. Technology is therefore an essential element to ensure the success of an improvement process. Translating the technology aspect to testing naturally points towards Artificial Intelligence (see section 1.4.4), test automation and test tools (see figure 1.1). In addition to test-execution tools that support test automation, there are many other test tools that fall under the technology aspect, such as test-design tools, test-management tools, performance-testing tools, and static-testing tools.

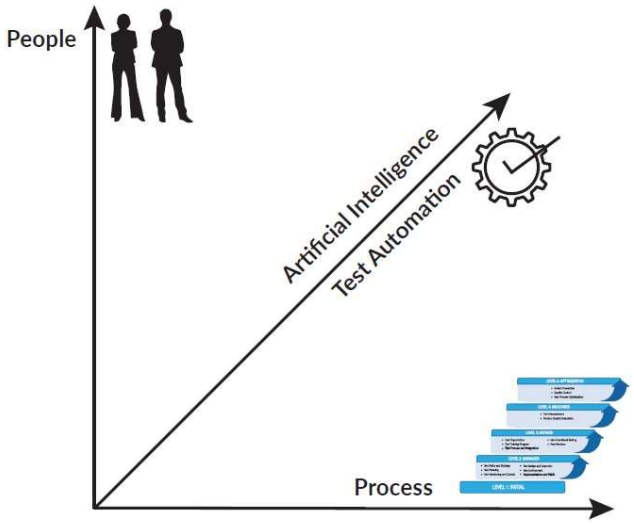

Figure 1.1 The PPT model for Testing

The PPT framework is all about how the three elements interact and must balance each other. Although the three elements exist independently, the actions of one component will affect the others. For example, if technology changes, changes will also be seen in people and processes. The testing community often debates what has more impact: ISTQB, TMMi or Artificial Intelligence/Test Automation. The focus of the discussion should not be how they compare and contrast, but how they can be used together successfully. More on the PPT framework and how to apply the framework in a testing and TMMi context can be found in “ISTQB, TMMi or Test Automation, Three Pillars for Success: The PPT Framework” [Veenendaal22].

1.4.4 Artificial Intelligence

Artificial Intelligence (AI) is transforming the way modern software is built, deployed, and maintained. In the field of software testing, AI enhances traditional testing approaches by automating repetitive tasks, generating intelligent test cases from requirements, detecting patterns in defects, and predicting potential failures before they occur. By leveraging techniques such as machine learning and natural language processing, AI-driven test

tools can adapt to changes in applications, improve test coverage, and accelerate release cycles. As software systems grow more complex, AI is rapidly becoming an essential component in delivering faster, more reliable, and higher-quality software. AI is not only enhancing test activities such as test design and test execution, but is increasingly acting as a decision-support mechanism within the test lifecycle. AI can assist in identifying high-risk areas, recommend optimal test coverage, and prioritize test execution based on historical defect patterns and usage data. In certain controlled scenarios, AI can also enable partial decision-making, such as automated test case selection for regression suites or intelligent defect classification, while still operating under human oversight.

However, as noted in the previous paragraph, TMMi is a framework designed to support test *process* improvement. Artificial Intelligence (AI) is a powerful emerging technology that will influence how testers operate, offering opportunities to make testing more efficient and, when applied wisely, more effective. Perhaps AI will help testing finally gain the time needed to perform all essential testing and better manage the growing complexity and rising expectations for product quality. However, like test automation and tools, AI is "just" a technology. In the context of TMMi, the People, Process and Technology (PTT) framework is again referred to (refer to section 1.4.3 for more on the PPT framework and also figure 1.1). To apply AI (technology) successfully, one needs to know and understand what one is doing (process) and be able to evaluate and use it wisely (people). The process part in this case refers to the test (process) maturity of the organization. It makes sense to apply AI only to test process areas that have already been mastered. In other words, the benefits that can be achieved with AI, will only be achieved when there is a thorough understanding of the (defined) process, and those involved have the knowledge and skills to evaluate and validate the AI-generated results.

There are many testing areas where AI can play a supporting role and make testing both more effective and efficient. Annex A provides an expanded discussion on AI and software testing, including a detailed overview of where and how AI can support the various TMMi process areas.

1.4.5 Quality Engineering

Quality engineering is an engineering discipline focused on ensuring that products or services are designed, developed and produced to meet or exceed customer expectations and industry standards. It encompasses a variety of activities and methodologies aimed at maintaining and improving quality throughout the entire lifecycle of a product or service, from initial design through production and beyond. Key aspects of quality engineering include quality planning, design for quality, quality assurance, quality control (including testing), continuous improvement and customer feedback.

Quality engineering has gained increased recognition in the IT and testing community, especially in the context of DevOps, wherein quality engineering is defined as: "team members and their stakeholders taking joint responsibility to continuously deliver IT systems with the right quality at the right moment to the business and their customers" [Marcelis]. Quality engineering is a principle of software engineering concerned with applying quality measures to assure the quality of IT systems. In their book, Marcelis et al. distinguish several related activities, grouped under the headings of organizing topics and performing topics, for quality engineering. In Annex B this set of quality engineering topics is mapped to the TMMi process areas and improvement goals, thereby showing that TMMi is also highly applicable in the context of quality engineering. This demonstrates that within TMMi, the concept of testing is not just about executing test cases, but rather it is more broadly defined and used. Of course, each TMMi process area, goal and practice always need to be interpreted in the context in which they are being applied, in this case, in the context of quality engineering.

1.4.6 Software and Systems Engineering

The TMMi is intended to support testing activities and test process improvement in both the systems engineering and software engineering discipline. Systems engineering is an interdisciplinary approach that focuses on designing, integrating, and managing complex systems throughout their entire lifecycle. Systems engineering covers the development of complex multidisciplinary systems, which may or may not include software, but typically involves a combination of hardware and software. Software engineering covers the design, development, testing, deployment and maintenance of software systems.

1.4.7 TMMi and CMMI

TMMi is an independent test improvement model, but can also be used as a complementary model to the CMMI. CMMI is a leading process improvement framework that focuses on improving processes for product and service development [CMMIV3]. In many cases, a given TMMi process area benefits from mature processes outside of testing. For example, the TMMi process area Test Design and Execution benefits from a mature requirements engineering and management process. Paragraph 3.3 “Related CMMI Practice Areas” provides more information on the relationship between TMMi process areas and CMMI practice areas.

1.5 TMMi Assessments

Many organizations find value in benchmarking their progress in test process improvement for both internal purposes and external customers. Test process assessments focus on identifying improvement opportunities and understanding the organization’s position relative to the selected model or standard. The TMMi provides an excellent reference model to be used during such assessments. Assessment teams use TMMi to guide their identification and prioritization of findings. These findings, along with the guidance of TMMi practices, are used to plan improvements for the organization or project. The assessment framework itself is not part of the TMMi. Requirements for TMMi assessments are described by the TMMi Foundation in the document “TMMi Assessment Method Application Requirements” [TAMAR]. These requirements are based upon the ISO/IEC 33002 standard [ISO 33002]. The achievement of a specific maturity level shall imply the same thing for different assessed organizations. Rules for ensuring this required level of consistency are contained in the TMMi assessment method requirements. The TMMi assessment method requirements contain guidelines for various classes of assessments, e.g., informal and formal assessments. More details on TMMi assessments and certification are provided in [TAMAR].

1.6 Improvement Approach

1.6.1 IDEAL

The TMMi provides a test improvement framework to be used as a reference model during test process improvement. However, it does not provide an approach on how to perform test process improvement such as the IDEAL model [IDEAL]. IDEAL, which is based on similar ideas as with the well-known plan-do-check-act (PDCA) cycle, is an acronym that stands for Initiating, Diagnosing, Establishing, Acting and Learning.

Initiating (I) the improvement process. At the start of the improvement process, the objectives and scope of the process improvements are agreed upon by the stakeholders. Practical experiences have shown that the most powerful initial step to test process improvement is to build strong organizational sponsorship before investing in test process improvements and assessments.

Diagnosing (D) the current situation. The current test process is assessed to identify possible improvements.

Establishing (E) a test process improvement plan. The test improvement plan lists all of the detailed actions that must be performed to achieve improvements. Depending on the context, the plan can be highly informal and very lightweight.

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

Learning (L) from the improvement program. Having fully deployed the process improvements, it is essential to verify which benefits were achieved. Having learned what worked and what did not work, we must act on this information and only thereafter the next improvement cycle can start.

More details on the IDEAL improvement approach are provided in Annex C. Additional recommendations and guidelines regarding an approach for test process improvement can be found in "The Little TMMi" [Veenendaal and Cannegieter].

1.6.2 TMMi is not a Checklist

Sometimes TMMi is perceived and used as checklist where all (sub)practices need to be ticked-off. This is of course totally against the idea of using TMMi where interpretation of the intent of a goal and practice in context is key. One should always study a goal and practice and discuss how it will be best implemented to deliver business and added value for the organization.

Also, not every part of the TMMi model needs to be implemented 100% to be considered Fully Achieved. In order to score "F" (Fully Achieved) in relation to a particular process component, there should be consistent convincing evidence found of compliance with the TMMi. For Fully Achieved the practice achievement is expected to be in the range of >85% to 100%. To achieve certification a score of "L" (Largely Achieved) is sufficient. For Largely Achieved the practice achievement is expected to be in the range of >50% to ≤85% [TAMAR].

As explained in chapter 3, the various TMMi components are grouped into three categories: required, expected and informative. Only goals are a required component! They describe what an organization must achieve to satisfy a process area. Practices are a so-called expected component and describe what an organization will typically implement to achieve a goal. Subpractices are no more than informative components that provide details to help organizations get started in thinking about how to approach and implement practices to achieve goals.

In summary, the (level of) implementation of TMMi goals and practices should be driven by business needs and added value. They should always be interpreted and prioritized based on business context. Using TMMi goals, practices and subpractices as a checklist of activities that must be implemented one-by-one, and maybe even focusing simply on achieving certification, without considering business value and context, is not the approach to take. This will lead to a highly theoretical attempt in implementing TMMi, and would most likely not be successful. TMMi is intended to be practical, and should be applied as a business-driven and objective-driven test process improvement model.

2 TMMi Maturity Levels

2.1 Overview

The TMMi has a staged architecture for process improvement. It contains stages or levels through which an organization passes as its testing process evolves from one that is ad hoc and unmanaged, to one that is managed, defined, measured, and finally in a state of continuous improvement, referred to as optimization. Achieving each level ensures that an adequate improvement has been established as a foundation for the next. The internal structure of the TMMi is rich in testing practices that can be applied in a systematic way to support a quality testing process that improves in incremental steps. The TMMi consists of five levels that define a maturity hierarchy and outline an evolutionary path for improving test processes. Each TMMi level, except TMMi level 1, has a set of process areas that an organization must implement to achieve maturity at that level. Experience has shown that organizations perform best when they focus their test process improvement efforts on a manageable number of process areas at a time, and that those areas require increasing sophistication as the organization improves. Because each maturity level forms a necessary foundation for the next level, trying to skip a maturity level is usually counter-productive. However, test process improvement efforts should always focus on the needs of the organization in the context of its business environment. Also, sometimes process areas at higher maturity levels may address the current needs of an organization or project. For example, organizations seeking to move from maturity level 1 to maturity level 2 are frequently encouraged to establish a test group, which is addressed by the Test Organization process area which is part of maturity level 3. Although the test group is not a necessary characteristic of a TMMi level 2 organization, it can nevertheless be a useful part of the organization's approach to achieve TMMi level 2.

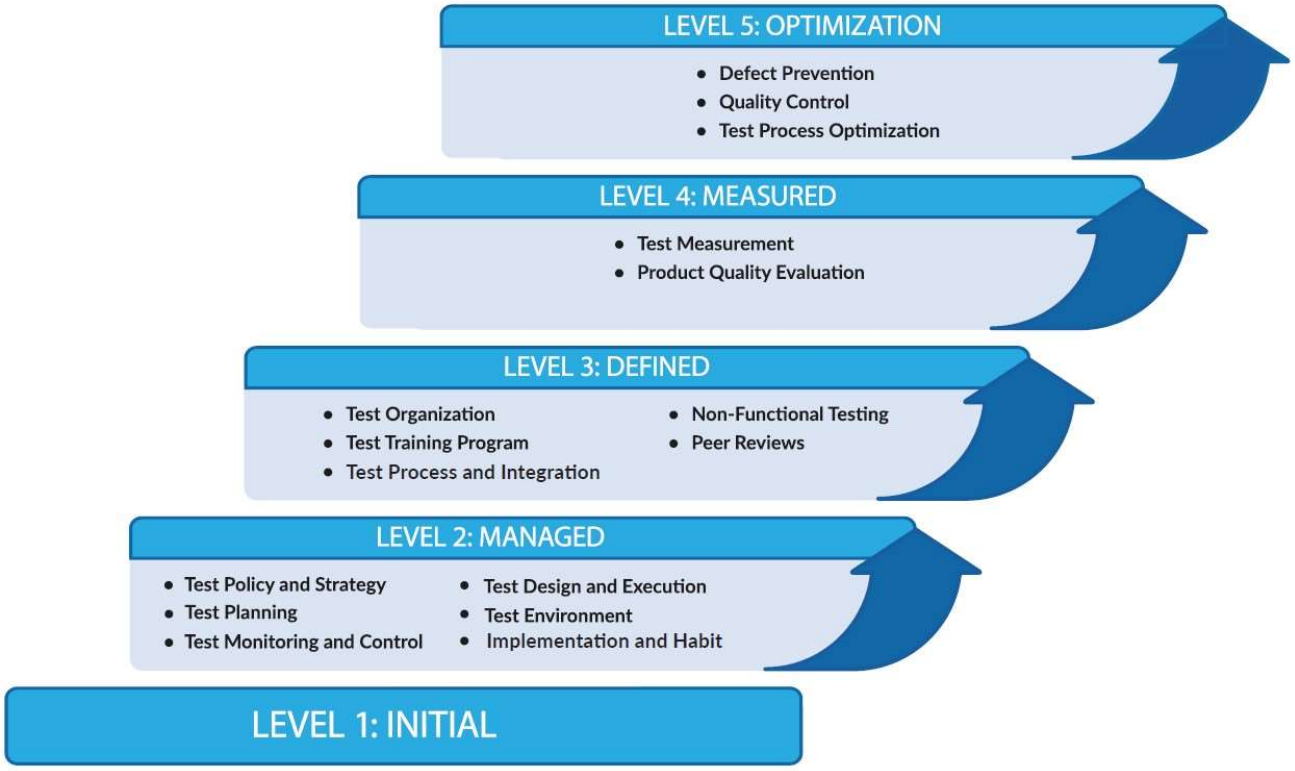

Figure 2.1: TMMi maturity levels and process areas

The process areas for each TMMi maturity level are shown in figure 2.1. The process areas are fully described in the various chapters hereafter in this document. In this chapter, each TMMi level is briefly explained along with a description of the characteristics of organizations operating at each TMMi level. The descriptions introduce the reader to the evolutionary path of TMMi-based test process improvement.

2.2 TMMi Level 1 Initial

TMMi level 1 applies to a wide-range of organizations. At this level there are no process areas and maturity goals with which to comply. This means that every organization is at least at TMMi level 1. TMMi level 1 is basically a “free” level. In some TMMi level 1 organizations, testing is still a highly chaotic, undefined process largely performed by development, while others may already be well on their way to becoming a TMMi level 2 organization. At TMMi level 1, testing activities are typically performed without adequate process descriptions and process management. Success in level 1 organizations depends on the competence and heroics of the people working within them. Test cases are most often developed ad hoc, without a test design step and not using test design techniques. They are also developed relatively late in the lifecycle, and not used to find defects in requirements or user stories. Developers conduct component testing, but a definition of what component testing means, including its objective, is missing.

The objective of testing at this level is to show that the software runs without major failures. Products are released without adequate visibility regarding product quality and risks. Within testing there is a lack of resources, tools and well-trained staff. At TMMi maturity level 1, organizations are characterized by a tendency to overcommit, abandonment of processes in times of crisis, and an inability to repeat their successes. In addition, products tend not to be released on time, budgets are overrun and delivered quality is not according to expectations.

As already stated, TMMi level 1 applies to a wide range of organizations with respect to test maturity. There are those that are just starting to focus on testing as a dedicated practice and thereby, in parallel, commence their test process improvement journey. While some processes may exist, these organizations are very people-dependent and focus on testing as an execution-based activity. Planning, monitoring and managing dependencies like test environments are done rather ad hoc, when the project lead or other dependent roles ask for it.

Annex D lists and briefly describes some testing practices that are typically performed by TMMi level 1 organizations that are already “on their way”. This sample set of TMMi level 1 testing practices can also be used as an initial checklist by organizations that have yet to start on their test improvement journey.

2.3 TMMi Level 2 Managed

At TMMi level 2, testing becomes a managed process. The process discipline reflected by TMMi maturity level 2 helps to ensure that proven practices are retained during times of stress. An infrastructure is put in place to implement and perform the set of processes and for them to become habitual. The infrastructure includes process descriptions, resource availability and training. As part of governance, management commitment is established, ensuring the way testing is performed is relevant and important to the business and the organization. An organizational test policy and a company-wide or program-wide test strategy are established. At TMMi level 2, multiple test levels, such as component testing, integration testing, system testing and user acceptance testing, are identified, defined and performed. For each identified test level, specific testing objectives are defined in the test strategy with the aim of aligning the various test levels.

At TMMi Level 1, testing is largely informal and ad hoc. In contrast, at TMMi Level 2 a set of basic, essential, and structured test practices is implemented at project or team level, including test planning, test monitoring and control, test design and execution, and a supporting test environment. Test planning is practiced as a means of up-front thinking, whereby a test approach is defined based on the result of a product risk assessment. Risk assessment techniques are used to identify product risks based on documented requirements, such as user stories. Commitments to the test approach, exit criteria and resources needed are established with stakeholders and revised as necessary. Testing is monitored and controlled to ensure it is going according to plan and expectations, and control actions are taken if deviations occur. The status of test work products, test execution, and the quality of the test object are visible and reported to stakeholders. During test design, test conditions are identified based on requirements and subsequently implemented as automated test scripts, manual test procedures or test charters as a basis for test execution. During test execution, test logs are written and incidents found are reported and managed to closure. Testing may start relatively late in the development lifecycle, e.g., during the design or even during the coding phase, and may still be perceived by some stakeholders as a lifecycle phase that follows coding.

The main objective of testing in a TMMi level 2 organization is to verify that the product satisfies the specified requirements. However, quality problems at this TMMi level could still occur because although early involvement of testing is encouraged, it may not be consistently achieved. Post coding, execution-based testing is considered the primary testing activity.

The process areas at TMMi level 2 are:

- 2.1 Test Policy and Strategy
- 2.2 Test Planning
- 2.3 Test Monitoring and Control
- 2.4 Test Design and Execution
- 2.5 Test Environment
- 2.6 Implementation and Habit

2.4 TMMi Level 3 Defined

At TMMi level 3 testing becomes a defined process. Organizational standard test processes and test process assets are developed, maintained and used by all projects and teams. A fully defined process has enough detail that it can consistently be performed by trained and skilled people and is both persistent and habitual. Key terms that describe the objectives at TMMi level 3 are institutionalization and early involvement. Institutionalization, building on the practices of TMMi level 2, is now further enhanced by practices such as test organization, test career paths, test training program and a standard organizational test process. Early involvement and early defect detection are covered by practices centered on the integrated lifecycle model and peer reviews.

At TMMi level 3, testing is no longer confined to a phase that follows coding. It is fully integrated into the development lifecycle and transformed into proactive, early-stage analysis, which, in an Agile context, typically starts during iteration planning. The organization's set of standard test processes, which is the basis for TMMi maturity level 3, is established and improved over time. A test organization is implemented, e.g., in the form of a test competence center. Test process improvement is fully institutionalized as part of the test organization's set of practices. A dedicated test training program exists, and testing is perceived by all as being a profession.

Organizations at TMMi level 3 understand the importance of early defect detection and the role of reviews in quality control. Peer reviews take place throughout the development lifecycle and their approach is coordinated with dynamic testing. Test professionals are involved in requirements reviews, e.g., backlog refinement sessions. Whereas testing at TMMi level 2 mainly focuses on functional testing, at TMMi level 3, driven by business objectives, test improvements also include establishing a capability for non-functional testing, e.g., security, performance and usability.

A critical distinction between TMMi maturity level 2 and 3 is the scope of the standards, test process descriptions, and test process assets. At TMMi maturity level 2, these may still be different per project and team. At TMMi level 3 the way of working is aligned across projects and teams, which also allows for easier transfer of staff between projects or teams. Differences in the way of working that exist because of specific needs are now managed through a set of tailoring guidelines. Another distinction of TMMi level 3 is that process descriptions are typically described in slightly more detail than at TMMi level 2 because the target audience is wider. Consequently, at TMMi level 3, the organization may need to revisit not only the test policy and test strategy, but also the process descriptions of the TMMi level 2 processes.

The process areas at TMMi level 3 are:

3.1 Test Organization

3.2 Test Training Program

3.3 Test Process and Integration

3.4 Non-Functional Testing

3.5 Peer Reviews

2.5 TMMi Level 4 Measured

Achieving the goals of TMMi level 2 and 3 has the benefits of putting in place a technical, managerial, and staffing infrastructure capable of thorough testing and providing support for test process improvement. Testing has become a defined process applied consistently throughout the organization, which is a prerequisite for a successful and meaningful measurement program. With all of this in place, testing can become a measured process to encourage further growth and accomplishment.

An organization-wide test measurement program will be implemented whereby test measurement objectives, test measures and procedures are defined based on identified information needs and business objectives to optimize resource usage. Information needs typically focus on improved understanding of test performance or results of test improvement actions. Test measurement is the process of identifying, collecting, and analyzing data from the test process activities and the products being developed, to understand the effectiveness and efficiency of the test processes. Test measurement will also provide support to individual projects and teams by offering historical data and metrics, e.g., to support accurate estimation and planning. An organizational test measurement repository is established to enable sharing of historical data across projects and teams.

With respect to product quality, the presence of a measurement program allows an organization to implement a product quality evaluation process at project or team level. By starting with identifying product quality needs, it is ensured that only the necessary product quality characteristics are included in the scope of measurement. (Work) products are evaluated using quantitative criteria for quality characteristics such as functional suitability, reliability, usability and maintainability. Product quality is understood in quantitative terms and is managed to the defined objectives throughout the lifecycle. Peer review as a defect detection

technique is transformed, using sampling, into a technique to measure product quality. The overall objective is to contribute to satisfying the needs of the business, users and customers by delivering quality products.

The process areas at TMMi level 4 are:

4.1 Test Measurement

4.2 Product Quality Evaluation

2.6 TMMi Level 5 Optimization

The achievement of all previous test improvement goals at levels 2 through 4 of TMMi has created an organizational infrastructure for testing that supports a completely defined and measured process. At TMMi maturity level 5, an organization is capable of continuously improving its processes based on a quantitative understanding of statistically controlled processes. Improving test process performance is carried out through incremental and innovative process and technological improvements. The testing methods and techniques are constantly being optimized and there is a continuous focus on fine tuning and process improvement. To support the continuous improvement of the test process infrastructure, and to identify, plan and implement test improvements, a permanent test process improvement group (also known as a Test Process Group) is formally established and is staffed by members specifically trained in the knowledge and skills required. Support for a Test Process Group formally begins at TMMi level 3 when the test organization is introduced. At TMMi level 4 and 5, its responsibilities grow as more high-level practices are introduced.

At TMMi level 5, testing completes its journey from being a detection-focused process to being one centered on defect prevention. Defect prevention is a mechanism used to evaluate the development and testing process and to identify the most effective improvements relating to product quality. It involves analyzing selected defects, identifying their root and common causes across teams, products and value streams, and defining specific actions to prevent the occurrence of similar defects in the future.

Quality control consists of the procedures and practices employed to ensure that a work product or deliverable conforms to requirements and standards. The principles of quality control can also be applied to the process creating the product, thereby creating a feedback loop in line with the prevention-oriented and optimizing approach of TMMi level 5. At TMMi level 5, organizations use quality control to drive the testing process. The process area Quality Control addresses the practices for establishing a statistically controlled test process, and for testing being performed, based on operational profiles and statistical techniques.

At the highest level of the TMMi, the test process is subject to continuous improvement across the entire organization. The test process is quantified, improved and fine-tuned, and capability growth is an on-going process. Test Process Optimization introduces a mechanism to continuously improve testing. The process area Test Process Optimization addresses the practices for continuously identifying test process improvements, evaluating and selecting new testing technologies and deploying them in the organization’s standard test process.

The process areas at TMMi level 5 are:

5.1 Defect Prevention

5.2 Quality Control

5.3 Test Process Optimization

3 Structure of the TMMi

In this chapter, the components and structure of the TMMi are described. The TMMi structure distinguishes between required components (goals), expected components (practices) and informative components (e.g., subpractices and example work products). In addition, this chapter describes how CMMI-DEV practice areas can support a TMMi implementation.

3.1 Required, Expected and Informative Components

The various TMMi components are grouped into three categories: required, expected and informative.

3.1.1 Required Components

Required components describe what an organization shall achieve to satisfy a process area. These achievements must be clearly implemented in an organization's processes. The only required components in TMMi are the goals. Goal satisfaction is used in assessments as the basis for deciding if a process area has been achieved and satisfied.

3.1.2 Expected Components

Expected components describe what an organization will typically need to implement to achieve a required component. Expected components guide those who implement improvements or perform assessments. The expected components in TMMi are the practices. Goals can only be considered achieved when the organization's processes include either the practices as described, or acceptable alternatives. For example, in the TMMi and Agile [TMMiAgile], and TMMi and DevOps [TMMiDevOps] documents acceptable alternatives are provided in detail for those implementing TMMi in an Agile or DevOps context.

3.1.3 Informative Components

Informative components provide details that help organizations start considering how to approach the required and expected components. The TMMi informative components are the subpractices, example work products, notes, examples, and references.

3.2 Components of the TMMi

The required and expected components of the TMMi model can be summarized to illustrate their relationship as in figure 3.1 "TMMi Structure and Components". The following sections provide a description of the components. Note that the TMMi also provides a specific glossary of terms. The definitions of the terms used in the glossary are largely based on international test terminology standard developed by the International Software Testing Qualifications Board (ISTQB): Standard glossary of terms used in Software Testing [ISTQB].

3.2.1 Maturity Levels

In the TMMi, a maturity level can be regarded as a degree of organizational test process quality. It is defined as an evolutionary plateau of test process improvement. Each level progressively develops an important part of the organization's test processes. The TMMi has five maturity levels, each defining what to implement in order to achieve the particular level. As an organization achieves higher maturity levels, its test processes

become more mature. To reach a maturity level, an organization must satisfy all goals of the process areas at the specific level and also those at lower maturity levels. Note that all organizations possess a minimum of TMMi level 1, as this level does not have any goals that must be satisfied.

3.2.2 Process Areas

As stated, with the exception of level 1, each maturity level consists of several process areas indicating on what an organization should focus to improve its test process. Each process area identifies a cluster of test related activities. When the practices are all performed correctly, significant improvements to the activities related to that area will be made. In the TMMi, only those process areas that are considered to be key determinants of test process capability are identified. All process areas of the maturity level and the lower maturity levels must be satisfied to consider a maturity level to be achieved. For example, if an organization is at TMMi level 3, it has satisfied all of the process areas at both TMMi level 2 and TMMi level 3.

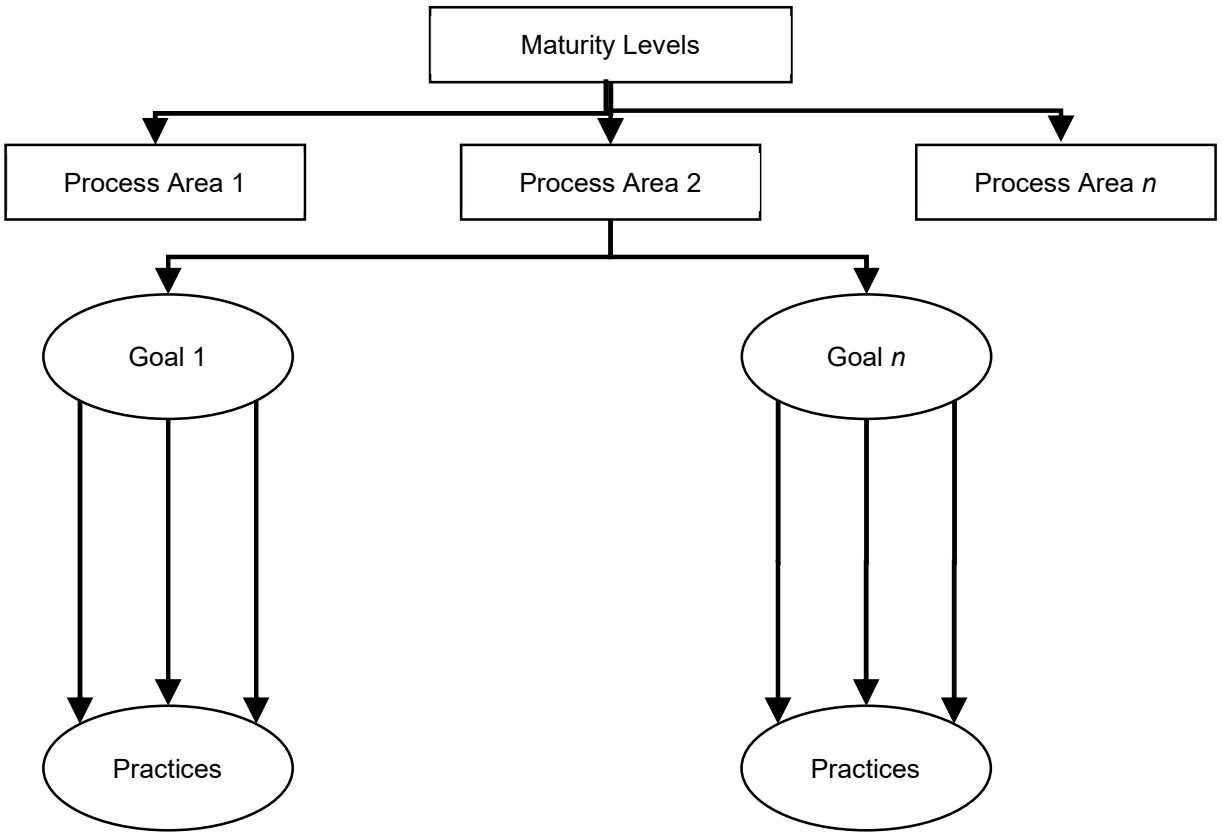

Figure 3.1: TMMi structure and components

Purpose

The purpose statement describes the intent of the process area and is an informative component. For example, the purpose statement of the Test Planning process area is “to define a test approach based on the identified risks and the defined test strategy, and to establish and maintain well-founded plans for performing and managing the testing activities”.

Value

The value statement describes the business benefits and added value of the process area that are likely to be achieved from implementing the process area.

Introductory Notes
The introductory notes section of the process area describes the major concepts covered in the process area and is an informative component.

Scope
The scope section of the process area specifically identifies the test practices that are addressed by the process area, and, if necessary, test practices that are explicitly outside the scope of the process area.

Goal
A goal describes the required characteristic that must be present to satisfy the process area. Each goal also contains a description of the added value of the goal. A goal is a required model component and is used in assessments to determine whether a process area is satisfied.

Practice
A practice is the description of an activity that is considered important in achieving the associated goal. The practice describes the activities expected to result in achievement of the goals of a process area. A practice is an expected model component.

Example Work Products
The example work products section lists sample outputs from a practice. These examples are called 'example work products' because there are often alternative work products that are just as effective but are not listed. An example work product is an informative model component.

Subpractice
A subpractice is a detailed description providing guidance for interpreting and implementing a practice. Although sometimes worded prescriptively, subpractices are an informative component and only intended to provide ideas that may be useful for test process improvement.

Supporting Informative Components
There are many times when additional information is needed to describe a concept. This informative information is provided by way of the following components:

Notes
A note is text that can accompany any other model component. It may provide detail, background, or rationale. A note is an informative model component.

Examples
An example is a component comprising text or a list of items, usually in a box, that can accompany almost any other component and provides one or more examples to clarify a concept or activity. An example is an informative model component.

References
A reference is a pointer to additional or more detailed information in related process areas and can accompany almost any other model component. A reference is an informative model component.

3.3 Related CMMI Practice Areas

Although TMMi is most often used in isolation, it can also be used as a complementary model to the CMMI. In many cases a given TMMi level can benefit from partly- or fully-deployed CMMI-DEV practice areas [CMMIV3].

Tables 3.1 to 3.4 summarize the CMMI-DEV practice areas that relate to the TMMi process areas. An overview of the CMMI-DEV practice areas related to TMMi level 2 achievement is shown in table 3.1. Tables 3.2, 3.3, and 3.4 provide an overview of the related CMMI-DEV practice areas for TMMi levels 3, 4 and 5 respectively.

TMMi Level 2 Process Area	CMMI Practice Area
2.1 Test Policy and Strategy	*Governance;* senior management provide their information needs and sets organizational directives for testing and thus provide support for G1 Establish a Test Policy. *Managing Performance and Management* provides support in the implementation of G3 Establish Test Performance Indicators.
2.2 Test Planning	*Estimation* provides supporting practices for test estimation. *Planning*; project management practices can be re-used for test management. *Risk and Opportunity Management*; practices for identifying product and project risks.
2.3 Test Monitoring and Control	*Monitor and Control*; practices can be re-used for monitoring and controlling test progress. *Risk and Opportunity Management*; practices for controlling product and project risks.
2.4 Test Design and Execution	*Requirements Development and Management*; requirements management practices can be built upon for establishing and maintaining horizontal traceability. Requirements development aims to deliver high-quality requirements which is crucial for test design and testing because they provide clear, complete, and unambiguous specifications that determine what to test.
2.5 Test Environment	*Requirements Development and Management*; requirements development practices to be re-used with G1 Develop Test Environment Requirements.
2.6 Implementation and Habit	*Configuration Management* provides support for G3 Manage configurations. *Governance*; provide support for G1 Establish Governance. *Implementation Infrastructure* provides support for G2 Implement infrastructure.

Table 3.1: Related CMMI-DEV practice areas for TMMi level 2 process areas

TMMi Level 3 Process Area	CMMI Practice Area
3.1 Test Organization	*Process Management* provides support in the implementation of G4 Determine, Plan and Implement Test Process Improvements.
3.2 Test Training Program	*Organizational Training* provides supporting practices for Test Training Program.
3.3 Test Process and Integration	*Process Asset Development* provides support for G1 Develop the Organizational Test Process and Test Process Assets. *Process Management* provides support for G3 Deploy the Organizational Test Process and Test Process Assets.
3.4 Non-Functional Testing	*Risk and Opportunity Management;* practices for identifying product risks. *Requirements Development & Management*; requirements management practices can be built upon for establishing and maintaining horizontal traceability.

TMMi Level 3 Process Area	CMMI Practice Area
3.5 Peer Reviews	*Peer Reviews* provides supporting practices for Peer Reviews.

Table 3.2: Related CMMI-DEV practice areas for TMMi level 3 process areas

TMMi Level 4 Process Area	CMMI Practice Area
4.1 Test Measurement	*Managing Performance and Measurement* provides support for both goals of the Test Measurement process area.
4.2 Product Quality Evaluation	*Managing Performance and Measurement* provides some support for the measurement practices within Product Quality Evaluation.

Table 3.3: Related CMMI-DEV practice areas for TMMi level 4 process areas

TMMi Level 5 Process Area	CMMI Practice Area
5.1 Defect Prevention	*Causal Analysis and Resolution* provides support for both goals of the Defect Prevention process area.
5.2 Quality Control	*Managing Performance and Measurement* provides support for Quality Control, especially G1 Establish a Statistically Controlled Test Process
5.3 Test Process Optimization	*Process management* manages and implements continuous improvement of processes and as such provides support throughout for the Test Process Optimization process area.

Table 3.4: Related CMMI-DEV practice areas for TMMi level 5 process areas

Verification and Validation

Note that the CMMI-DEV test-specific practice area Verification and Validation is not listed as a supporting practice area for TMMi. For this CMMI-DEV practice area the relationship is different. For the practice area Verification and Validation, the TMMi process areas provide support and a more detailed specification of what is required to establish a defined verification and validation process. Here, the test-specific TMMi improvement model complements the CMMI improvement model.

TMMi Level 2 Managed

At TMMi level 2, testing becomes a managed process. The process discipline reflected by TMMi maturity level 2 helps to ensure that proven practices are retained during times of stress. An infrastructure is put in place to implement and perform the set of processes and for them to become habitual. The infrastructure includes process descriptions, resource availability and training. As part of governance, management commitment is established, ensuring the way testing is performed is relevant and important to the business and the organization. An organizational test policy and a company-wide or program-wide test strategy are established. At TMMi level 2, multiple test levels, such as component testing, integration testing, system testing and user acceptance testing, are identified, defined and performed. For each identified test level, specific testing objectives are defined in the test strategy with the aim of aligning the various test levels.

At TMMi Level 1, testing is largely informal and ad hoc. In contrast, at TMMi Level 2 a set of basic, essential, and structured test practices is implemented at project or team level, including test planning, test monitoring and control, test design and execution, and a supporting test environment. Test planning is practiced as a means of up-front thinking, whereby a test approach is defined based on the result of a product risk assessment. Risk assessment techniques are used to identify product risks based on documented requirements, such as user stories. Commitments to the test approach, exit criteria and resources needed are established with stakeholders and revised as necessary. Testing is monitored and controlled to ensure it is going according to plan and expectations, and control actions are taken if deviations occur. The status of test work products, test execution, and the quality of the test object are visible and reported to stakeholders. During test design, test conditions are identified based on requirements and subsequently implemented as automated test scripts, manual test procedures or test charters as a basis for test execution. During test execution, test logs are written and incidents found are reported and managed to closure. Testing may start relatively late in the development lifecycle, e.g., during the design or even during the coding phase, and may still be perceived by some stakeholders as a lifecycle phase that follows coding.

The main objective of testing in a TMMi level 2 organization is to verify that the product satisfies the specified requirements. However, quality problems at this TMMi level could still occur because although early involvement of testing is encouraged, it may not be consistently achieved. Post coding, execution-based testing is considered the primary testing activity.

The process areas at TMMi level 2 are:

- 2.1 Test Policy and Strategy
- 2.2 Test Planning
- 2.3 Test Monitoring and Control
- 2.4 Test Design and Execution
- 2.5 Test Environment
- 2.6 Implementation and Habit

Each of these process areas is described in more detail in the sections hereafter.

Process Area 2.1 Test Policy and Strategy (TPS)

Purpose

The purpose of the Test Policy and Strategy process area is to develop and establish a test policy, and an organization-wide or program-wide test strategy in which the test activities, e.g., test types and test levels, are defined. To measure test performance, the value of test activities and expose areas for improvement, test performance indicators are introduced.

Value

The test policy creates a common view of testing within an organization. By involving management, it also creates management commitment to testing and test process improvement. The test process performance indicators aim to show management the results being achieved through the test process improvement activities thereby ensuring continuation of management commitment. The organizational test strategy guides projects and teams, and improves their alignment. An implemented test strategy leads to a more efficient and effective test process.

Introductory Notes

When an organization wants to improve its test process, it should first clearly define a test policy. The test policy defines the organization's overall test objectives, goals and strategic views regarding testing. It is important for the test policy to be aligned with the overall business (quality) policy of the organization. A test policy is necessary to attain a common view of testing and its objectives between all stakeholders within an organization. It prevents endless testing-based discussions at project or team level. This common view is required to align test (process improvement) activities throughout the organization. Within many organizations that apply an Agile or DevOps lifecycle, there is much discussion on the changing role of testing, independence of testing, test automation and professional testers. These topics are typically issues that should be addressed in a discussion with management and other stakeholders and documented in a test policy. Within the test policy the objectives for test process improvement should be stated. These objectives will subsequently be translated into a set of test performance indicators. The test policy and the accompanying test performance indicators provide a clear direction, and a means to communicate expected and achieved levels of test performance. The test performance indicators must show the value of testing and test process improvement to the stakeholders. The test performance indicators will provide a quantitative indication of whether the organization is improving and achieving the defined set of test (improvement) goals.

Based upon the test policy a test strategy will be defined. The test strategy addresses the generic product risks and presents a process for mitigating those risks in accordance with the test policy. The test strategy serves as a starting point for the testing activities within projects and teams. A test strategy ensures less overlap between the various test levels and test types, leading to a more efficient test process. Also, since the test objectives and approach of the various levels and types are aligned, fewer gaps are likely to remain, leading to a more effective test process. In an Agile context the testing quadrants model [Crispin] is often used as a starting framework on which to build a test strategy. It will define the relationship from Agile teams to those test levels that are performed outside their scope, e.g., hardware/software integration testing, system integration testing or beta testing. In a DevOps context the test strategy is built around the CI/CD pipeline to represent the flow of testing activities within the pipeline. A test strategy ensures that all those involved in testing understand the bigger testing picture.

Note, that a test policy and test strategy update is usually required as an organization evolves and moves up the levels of the TMMi.

Scope

The process area Test Policy and Strategy involves the definition and deployment of a test policy and test strategy at an organizational level. Note that a test policy can also be incorporated within a quality policy. Within the test strategy, test levels and test types are identified. For each test level and type, objectives, responsibilities, main tasks and entry/exit criteria are defined. To measure test performance and the accomplishment of test (improvement) objectives, test performance indicators are defined and implemented.

Goal and Practice Summary

G1 Establish a Test Policy
- P1.1 Define test goals
- P1.2 Define test policy
- P1.3 Distribute the test policy to stakeholders

G2 Establish a Test Strategy
- P2.1 Define test strategy
- P2.2 Distribute the test strategy to stakeholders

G3 Establish Test Performance Indicators
- P3.1 Define test performance indicators
- P3.2 Deploy test performance indicators

Practices by Goal

G1 Establish a Test Policy

A test policy, aligned with the business (quality) policy, is established and agreed upon by the stakeholders to create a common view of testing within the organization.

P1.1 Define test goals

Define and maintain test goals based upon test directives defined by management, business needs and objectives.

Example work products

1. Test goals

Subpractices

1. Study test directives, business needs and objectives

Examples of business needs and objectives to be studied include:
- Mission statement
- Business drivers
- Business (quality) policy
- Business and user needs regarding the products
- Main goals of a quality program
- Type of business, e.g., risk level of products being developed

2. Provide feedback for clarifying test directives, business needs and objectives

3. Study the test performance improvement directives defined by management
4. Define test goals traceable to test directives, business needs, objectives and test performance improvement directives

Examples of test goals include:

- Validate products for 'fit-for-use'
- Prevent defects from occurring in operation
- Improve product quality
- Verify compliance with external standards
- Provide visibility regarding product quality and risks

5. Review the test goals with stakeholders
6. Revisit and revise the test goals as necessary, e.g., on a yearly basis

Refer to process area 2.6 Implementation and Habit for more on organizational test directives and test performance improvement directives defined by management.

P1.2 Define test policy

A test policy, aligned with the business (quality) policy, is defined based on the test goals and agreed upon by the stakeholders.

Example work products

1. Test policy

Subpractices

1. Define the test policy based on the defined test goals

Examples of typical topics to be part of a test policy include [after ISO 29119-3]:

- Definition of testing
- Objectives and added value of testing
- Quality levels to be achieved
- Basic views regarding testing and the testing profession
- Independence of testing
- Required training and certifications
- Key responsibilities of testing
- Tester ethics
- High level test process definition
- Applicable standards

2. Review the test policy with stakeholders, e.g., senior management
3. Revisit and revise the test policy as necessary, e.g., on a yearly basis

P1.3 Distribute the test policy to stakeholders

The test policy and test goals are distributed to all relevant stakeholders, both within and outside testing (e.g., Agile or DevOps teams), and are presented and discussed to ensure understanding.

Example work products

1. Test policy presentation

Examples of distribution mechanisms include:

- Presenting in project and/or departmental meetings

- Referencing it via posters on the wall
- Making it part of the departmental onboarding program
- Providing access to it on a central web portal

G2 Establish a Test Strategy

An organization-wide or program-wide test strategy that identifies and defines the test levels and test types to be performed is established and deployed to guide projects and teams in their testing, to improve their alignment and prevent projects and teams from re-inventing the wheel.

The test strategy will serve as a starting point for testing to be performed in a project or team. Each project or team can tailor the overall organizational strategy to its specific needs. Any area of non-compliance with the test strategy shall be clearly documented, e.g., in their test plan

P2.1 Define test strategy

The test strategy is defined to identify and define the test levels and test types. For each level and type, the objectives, responsibilities, main tasks, entry/exit criteria and so forth are defined.

Example work products

1. Test strategy
2. Generic product risk list, with priority assigned to each risk

Subpractices

1. Study test policy and goals
2. Identify and analyze the generic product risks for the type of products and applications being developed
3. Discuss and prioritize the generic product risks with stakeholders
4. Define the test strategy based on the generic product risks and defined test policy

Examples of topics to be addressed as part of a test strategy include [after ISO 29119-3]:

- Generics risks of the products being developed
- Overall test model (V-model, Agile or DevOps lifecycle) to be employed as a way to mitigate the risks
- Test levels (e.g., component, integration, system and acceptance test)
- Test Types (e.g., functional testing, regression testing and non-functional testing such as performance and security)
- Objectives, responsibilities and main tasks at each test level and test type
- Test case design techniques to be used at each test level and test type
- Entry and exit criteria for each test level and test type, including, for DevOps, the quality gates embedded in each test job within the CI/CD pipeline
- Which test types are carried out at which test level
- Standards that must be adhered to
- Organization, including test roles, responsibilities and communication structure
- Level of independence of testing
- Environment in which the tests will be executed
- Approach to confirmation and regression testing
- Approach to automation at each test level and test type
- Configuration and incident management

5. Review the test strategy with stakeholders
6. Revisit and revise the test strategy as necessary, e.g., on a yearly basis

P2.2 Distribute the test strategy to the stakeholder

The test strategy is distributed to all relevant stakeholders, both within and outside testing (e.g., Agile or DevOps teams), and is presented and discussed to ensure understanding.

Example work products

1. Test strategy presentation

Examples of distribution mechanisms include:

- Presenting in project and/or departmental meetings
- Referencing it via posters on the wall
- Making it part of the departmental introduction program
- Providing access to it on a central web portal

G3 Establish Test Performance Indicators

A set of goal-oriented test process performance indicators measuring the quality of the test process is established and deployed to be able to show to stakeholders the results being achieved through the test process improvement activities.

P3.1 Define test performance indicators

The test performance indicators are defined based upon the test policy, test goals and management information needs, including a procedure for data collection, storage and analysis.

Example work products

1. Test performance indicators
2. Data collection, storage, analysis and reporting procedure

Subpractices

1. Study test policy, test goals, e.g., the objectives for test process improvement, and management information needs
2. Define test performance indicators traceable to the test policy, test goals and management information needs

Examples of test performance indicators include:

- Test effort and cost
- Defect detection percentage
- Escaped defects
- Customer satisfaction rating
- Change failure rate, the percentage of deployments that cause a failure in production requiring remediation, e.g., hotfix or rollback
- Test coverage
- Test maturity level

In general, the defined test performance indicators should relate to the business value of testing. In a DevOps context, they are also used to support value stream management and optimization, e.g., test execution lead time, to understand how this contributes to overall lead time.

3. Review the test performance indicators with stakeholders, e.g., senior management
4. Specify how the test performance indicators will be obtained and stored
5. Specify how the test performance indicators will be analyzed and reported

Refer to process area 2.6 Implementation and Habit for more information on management information needs.

Note, with Agile and DevOps the focus will typically be more team-based and systems thinking. This may result in a corresponding broadening of the indicators to the team and overall system rather than being confined solely to the specifics of testing itself.

P3.2 Deploy test performance indicators

Deploy the test performance indicators and provide their measurement results to stakeholders, e.g., senior management.

Example work products

1. Test performance indicator data
2. Measurement reports regarding the test performance indicators

Subpractices

1. Obtain specified test performance indicator data

   In a DevOps context this should be an integral part of the CI/CD activities, whereby the data should be available in almost real-time.
2. Analyze and interpret test performance indicator data
3. Manage and store test performance indicator data and analysis results
4. Report the test performance indicator data to stakeholders on a periodic basis
5. Assist stakeholders in understanding the results

Examples of actions to assist in understanding the results include:

- Discussing the results with relevant stakeholders
- Provide contextual information that provides background and explanation

Process Area 2.2 Test Planning (TPL)

Purpose

The purpose of Test Planning is to define a test approach based on the identified risks and the defined test strategy, and to establish and maintain well-founded plans for performing and managing the testing activities.

Value

Test planning balances test effort with functionality and quality delivered, and increases the likelihood of meeting the test objectives.

Introductory Notes

As part of Test Planning, the test approach is defined based on the outcome of a product risk assessment and the defined organization-wide or program-wide test strategy. The test approach will comply, or explain non-compliances, with the test strategy. Depending on the priority and category of the risks, it is decided which requirements (e.g., user stories) of the product will be tested, to what degree, how and when. The objective is to provide the most feasible coverage of the various parts of the system considering the identified product risks.

Based on the test approach, the work to be done and test products to be delivered are identified and estimated and, as a result, the proposed test approach is provided with clear effort, cost and/or size information. The product risks, test approach and estimates are defined in close co-operation with all stakeholders and the team, rather than just by testing. Within test planning standards to be adhered to, the required resources, stakeholders and their roles, and aspects relating to infrastructure are addressed.. In addition, test project risks are identified. As a result, the test plan will define what testing is required, when, how and by whom.

Finally, the test plan document is developed and agreed to by the stakeholders. The test plan provides the basis for performing and controlling the testing activities. Beware that the key to successful test planning is in upfront thinking (the activity), and much less in defining the associated test plan (the document).

Scope

The process area Test Planning involves performing a product risk assessment on the test object and defining a differentiated test approach based on the risks identified. It also involves developing estimates for the testing to be performed, establishing necessary commitments, and defining and maintaining the plan to guide and manage the testing. Test plans are developed as required and needed by the project or value stream. A test plan may be developed for a specific test level or test type, or as a master test plan defining a coherent test approach across multiple test levels and test types, possibly also involving external parties.

Within Agile and DevOps lifecycles, two types of (test) planning typically occur: release planning and iteration planning. However, in DevOps, release planning typically evolves into a more dynamic, continuous process that supports fast, reliable delivery. At TMMi level 3, with the process area Non-Functional Testing, test plans will be introduced for the non-functional quality characteristics, e.g., performance or security.

Goal and Practice Summary

G1 Perform a Product Risk Assessment

- P1.1 Identify product risks
- P1.2 Analyze product risks

G2 Establish a Test Approach
- P2.1 Identify items and features to be tested
- P2.2 Define the test approach
- P2.3 Define entry criteria
- P2.4 Define exit criteria

G3 Establish Test Estimates
- P3.1 Identify test work products to be developed and test tasks to be performed
- P3.2 Determine estimates for test effort and cost

G4 Develop a Test Plan
- P4.1 Establish the test schedule
- P4.2 Plan for test staffing
- P4.3 Plan stakeholder involvement
- P4.4 Identify test project risks
- P4.5 Establish the test plan

G5 Obtain Commitment to the Test Plan
- P5.1 Review test plan and obtain commitments
- P5.2 Reconcile work and resource levels

Practices by Goal

G1 Perform a Product Risk Assessment

A product risk assessment is performed to identify the critical areas for testing.

P1.1 Identify product risks

Product risks are identified and documented.

Example work products

1. Identified product risks

Subpractices

1. Identify and select stakeholders that need to contribute to the product risk assessment
2. Identify product risks using input from stakeholders and requirements documents

Examples of product risk identification techniques:
- Risk workshops
- Risk poker
- Brainstorming
- Expert interviews
- Checklists
- Lessons learned
- ROAM (Resolved, Owned, Accepted, Mitigated) [Baah]

3. Document the background and potential consequences of the risk
4. Identify the relevant stakeholders associated for each risk

P1.2 Analyze product risks

Product risks are evaluated, categorized and prioritized.

Example work products

1. Product risk list, with a category and priority assigned to each risk

Subpractices

1. Analyze the identified product risks for likelihood and impact
2. Categorize and group product risks

Examples of product risk categories include:

- Functional risks
- Architectural risks
- Non-functional risks, e.g., security, usability, efficiency, portability, maintainability, reliability
- Change related risks, e.g., regression

3. Prioritize the product risks for mitigation
4. Establish a horizontal traceability between product risks and requirements to ensure that the source of product risks is documented
5. Review and obtain agreement with stakeholders on the completeness, category and priority level of the product risks
6. Revise the product risks as necessary

Examples of when product risks need to be revised include:

- New or changing requirements (user stories)
- Change of the software development approach
- Lessons learned on quality issues in the project or iteration

G2 Establish a Test Approach

A test approach, based on identified product risks, is established and agreed upon to provide all stakeholders with a clear understanding of the approach, their specific tasks and objectives.

P2.1 Identify items and features to be tested

The items and features to be tested, and not to be tested, are identified based on the product risks.

Example work products

1. List of items and features to be tested and not to be tested

Subpractices

1. Breakdown the prioritized product risks into items and features to be tested and not to be tested
2. Document the risk level and source documentation (requirements), e.g., epic or user story, for each identified item and feature to be tested
3. Review with stakeholders the list of items and features to be tested and not to be tested

P2.2 Define the test approach

The test approach is defined to mitigate the identified and prioritized product risks.

Example work products

1. The approach, e.g., selected set of test design techniques, should be described in sufficient detail to support identification of major test tasks and estimation of the effort required to do each one.

Subpractices

1. Select the test design techniques and approach for developing test cases regarding the items and features to be tested. Multiple test design techniques are selected to provide adequate test coverage based on the identified product risks.

Criteria for selecting a test design technique include:

- Type of system
- Regulatory standards
- Customer or contractual requirements
- Level of risk
- Type of risk
- Documentation available
- Knowledge of the testers
- Time and budget
- Development lifecycle
- Previous experience with types of defects found

In a DevOps context, techniques like Behavior-Driven Development (BDD) and Acceptance Test-Driven Development (ATDD) are often used to define scenarios and identify test conditions.

2. Define the approach to test automation
3. Define the approach for reviewing test work products
4. Define the approach for confirmation testing

Examples of approaches for confirmation testing include:

- For all high-risk test items, a full confirmation test will be performed, re-executing the full test procedure
- For all low-risk test items the incidents are confirmation tested in isolation

5. Define the approach for regression testing

Examples of elements of a regression test approach include:

- Focus of regression testing, e.g., which items and/or features
- Methods to select the test cases to be part of the regression test set
- Using the testing pyramid [Cohn] as one of the starting principles for regression testing
- Types of testing to be performed
- Manual testing or using test automation

6. Define suspension and resumption criteria

Examples of suspension criteria related to the test process include:

- Number of critical defects
- Number of non-reproducible defects
- Issues with test execution due to the test environments

7. Identify the supporting test tools to be used

8. Identify significant constraints regarding the test approach

Examples constraints regarding the test approach include:
- Test resource availability
- Test environment features and availability
- Quality level of requirements
- Project deadlines

9. Align the test approach with the defined organization-wide or program-wide test strategy
10. Identify any non-compliance with the test strategy and its rationale
11. Review the test approach with stakeholders
12. Revise the test approach as necessary

Examples of when the test approach may need to be revised include:
- New or changed priority level of product risks
- Lessons learned applying the test approach

P2.3 Define entry criteria

The entry criteria, also called definition-of-ready, for testing are defined to prevent testing from starting under conditions that do not allow for a thorough test process.

Example work products

1. Entry criteria (definition-of-ready) per identified test level

Subpractices

1. Define a set of entry criteria related to the test process

Examples of entry criteria related to the test process include:
- The availability of a test summary report from the previous test level
- The availability of a test environment according to requirements
- The availability of documentation, e.g., test release notes, user manual, installation manual

2. Define a set of entry criteria related to product quality

Examples of entry criteria related to product quality include:
- User stories comply with the INVEST criteria [Wake]
- Acceptance criteria are available in Given/When/Then format and cover happy path, edge cases, and error scenarios
- A successful intake test
- No outstanding defects (of priority level X) from previous test level
- All outstanding defects from previous test level have been analyzed

3. Review the entry criteria with stakeholders, especially those stakeholders responsible for meeting the entry criteria

P2.4 Define exit criteria

The exit criteria, also called definition-of-done, for testing are defined to determine when testing is complete.

Example work products

1. Exit criteria (definition-of-done) per identified test level

Subpractices

1. Define a set of exit criteria related to the test process

Examples of exit criteria related to the test process include:

- Percentage of tests prepared that have been executed (successfully)
- Percentage of test cases executed
- Percentage of coverage for each test item, e.g., code coverage or requirements coverage
- The availability of an approved test summary report

2. Define a set of exit criteria related to product quality

Examples of exit criteria related to product quality include:

- All high priority product risks mitigated
- Defect detection rate falls below a defined threshold
- Code coverage criteria achieved
- All test cases have passed
- Number of outstanding defects (by priority level)
- Percentage of software modules supported by a reviewed design
- Level of customer satisfaction

3. Review the exit criteria with stakeholders, especially those responsible for meeting the exit criteria

Note that the exit criteria of a test level should be aligned with the entry criteria of the subsequent test level.

G3 Establish Test Estimates

Well-founded test estimates are established and maintained for use in discussing the test approach with stakeholders to reduce uncertainty, and to provide a basis for making test commitments and planning of testing activities.

P3.1 Identify test work products to be developed and test tasks to be performed

Establish an overview of the test work products to be developed and test tasks to be performed, to clearly define the scope of the testing to be performed and, thereby, the scope for the test estimate.

Example work products

1. Test work products list
2. Test tasks to be performed

Subpractices

1. Identify test work products to be developed based on the defined test approach
2. Identify test work products that will be acquired externally
3. Identify test work products that will be re-used
4. Identify test tasks to be performed related to the test work products, e.g., test design, test implementation and test execution.
5. Identify indirect test tasks to be performed such as test management (reporting), test completion, meetings, configuration management, etc.

Note, also take into account tasks for implementing test environment requirements. Refer to the process area 2.5 Test Environment for more information on implementing the test environment requirements.

P3.2 Determine estimates for test effort and cost

Estimate the test size, effort and/or cost for the test work products to be created and test tasks to be performed based on the estimation rationale.

Example work products

1. Attribute estimates of test work products and test tasks
2. Test size estimates
3. Test effort estimates
4. Test cost estimates

Subpractices

1. Determine attributes of the test work products and test tasks

Examples of attributes used to estimate test work products and test tasks include:

- Size, e.g., expected number of test cases, number of pages, number of requirements
- Complexity of related test item, e.g., cyclomatic number
- Level of re-use
- Product risk level

2. Identify (technical) factors that influence the test estimate

Examples of factors that influence the test estimate include:

- Usage of test tools
- Quality of earlier test levels
- Quality of requirements (user stories)
- Knowledge and skill level of testers

3. Use an estimation technique to transform the attributes of the test work products and test tasks into estimates of size (e.g., story points), effort and/or cost

Examples of estimation techniques include:

- Three-point estimate
- Wide Band Delphi [Veenendaal02]
- Planning poker
- T-shirt sizing
- Swim lane sizing

4. Estimate effort and/or cost of the supporting test infrastructure needs

Examples of supporting infrastructure needs include:

- Test environment
- Critical computer resources
- Office environment
- Test tools

5. Document assumptions made in deriving the estimates
6. Record the test estimation data, including the associated information needed to reconstruct the estimates

G4 Develop a Test Plan

A test plan is established and maintained to provide a basis for managing testing and to ensure effective and efficient communication with stakeholders.

P4.1 Establish the test schedule

The test schedule is established and maintained based on the developed test estimate and test lifecycle. In an Agile and DevOps context the schedule is often an ordering of user stories (backlog items) and (testing) tasks that reflect the release and iteration priorities.

Example work products

1. Test schedule

Subpractices

1. Identify test scheduling constraints resources and inputs needed
2. Identify test task dependencies
3. Define the test schedule (timing of testing activities)
4. Document assumptions made in defining the test schedule

P4.2 Plan for test staffing

A plan is created for the availability of the necessary test staff resources who have the required knowledge and skills to perform the testing. In an Agile and DevOps context, this practice is typically performed only once upon setting up the team.

Example work products

1. Staffing requirements
2. Inventory of skill needs
3. Staffing and new hire plan
4. Test training plan

Subpractices

1. Determine staffing requirements based on the list of test work products and tasks, test estimate and test schedule
2. Identify knowledge and skills needed to perform the test tasks
3. Assess the knowledge and skills available
4. Select mechanisms for providing needed knowledge and skills

   Examples of mechanisms include:
   - In-house training
   - External training
   - Coaching
   - External skill acquisition

5. Incorporate selected mechanisms into the test plan

P4.3 Plan stakeholder involvement

A plan is created for the involvement of the identified stakeholders. Stakeholders are identified from all phases of the test lifecycle by identifying the type of people and roles needed during the testing activities.

Stakeholders are also identified by their relevance to and the degree of interaction to the specific testing activities. In an Agile and DevOps context, the product owner will play an important part in representing the business stakeholders.

Example work products

1. List of stakeholders and their involvement

P4.4 Identify test project risks

The test project risks associated with testing are identified, analyzed and documented.

Example work products

1. Identified test project risks
2. Prioritized test project risk list (impediment log)
3. Test project risk mitigation plans

Subpractices

1. Identify test project risks

Examples of project risk identification techniques include:

- Brainstorming
- Expert interviews
- Checklists

2. Analyze the identified test project risks in terms of likelihood and impact
3. Prioritize the analyzed test project risks
4. Review and obtain agreement with stakeholders on the completeness and priority level of the documented test project risks
5. Define contingencies and mitigation actions for the (high priority) test project risks
6. Revise the test project risks as necessary

Examples of when test project risks may need to be revised include:

- When new test project risks are identified
- When the likelihood of a test project risk changes
- When test project risks are retired
- When testing circumstances change significantly

P4.5 Establish the test plan

A test plan is established and maintained as a basis for managing testing and guiding the communication with the stakeholders. The results of previous practices are documented in an overall test plan, tying together the information in a logical manner.

Example work products

1. Test Plan

Examples of elements of a test plan include [after ISO 29119-3]:

- An overall introduction, including the context of testing
- Non-compliances with the test strategy, including rationale
- Product risks
- Items to be tested (including priority level) and not to be tested
- Features to be tested (including priority level) and not to be tested
- Assumptions and constraints
- Test approach (e.g., test design techniques)
- Approach to re-testing and regression testing
- Entry and exit criteria
- Suspension and resumption criteria
- Test milestones and work products
- Test lifecycle and tasks
- Test environmental and test data needs
- Staffing and training needs (including roles and responsibilities)
- Stakeholder involvement
- Test reporting and communication
- Test estimate and test schedule
- Test project risks and contingencies

Refer to the process area 2.5 Test Environment for more information on test environmental needs and requirements.

Note, ISO 29113-3 also provides a specific template for a master test plan.

Examples of elements of a test plan in an Agile or DevOps context include:

- Product risks
- Epics / User stories to be tested (including priority level)
- Test approach
- Definition-of-done criteria

A test plan in an Agile or DevOps context could be displayed or reflected on the task board.

G5 Obtain Commitment to the Test Plan

Commitments to the test plan are established and maintained to achieve a consistent understanding by all stakeholders thereby reducing the likelihood of changes to the approach and plan at a later stage.

P5.1 Review test plan and obtain commitments

Review the test plan to achieve and obtain test commitments.

Example work products

1. Test plan review log
2. Documented commitments

Subpractices

1. Organize reviews with stakeholders to facilitate their understanding of the test commitments
2. Review (external) commitments with senior management as necessary

P5.2 Reconcile work and resource levels

Adjust the test plan to reconcile available and estimated resources.

Example work products

1. Revised test approach and corresponding estimation parameters
2. Renegotiated test budget

3. Revised test schedule

Subpractices

1. Discuss differences between estimates and available resources with stakeholders
2. Reconcile any differences between estimates and available resources

Note that reconciliation is typically accomplished by lowering or deferring technical performance, negotiating more resources, finding ways to increase productivity, changing the scope of the project or iteration such as removing features, outsourcing, adjusting staff skill mix, or revising the schedule.

Process Area 2.3 Test Monitoring and Control (TMC)

Purpose

The purpose of Test Monitoring and Control is to provide an understanding of test progress and product quality so that appropriate corrective actions can be taken when test progress or product quality deviate significantly from plan or expectations.

Value

Test Monitoring and Control increases the probability of meeting test and project objectives by taking early action to adjust for significant performance deviations.

Introductory Notes

The progress of testing and the quality of the products should both be monitored and controlled. Test progress is monitored by comparing the status of test (work) products, tasks, effort and cost to what is identified in the schedule and test plan. Product quality is monitored by means of indicators such as product risks mitigated, the number of defects found, number of open defects, and status against exit criteria (definition-of-done). In an Agile and DevOps context, monitoring is less plan driven, with progress continuously being reviewed using results from testing to adapt the plan and approach as necessary.

Monitoring involves gathering data, e.g., from test logs and test incidents reports, and deducing the test progress and product quality status. Test reports and dashboards are established to provide a common understanding of test progress and product quality. Appropriate corrective actions should be taken when the test progress or product quality deviate significantly from plan and/or expectations.

An essential part of test monitoring and control is test project risk management. Test project risk management is performed to identify, and solve as early as possible, major problems that undermine testing. Building on the test project risks already identified during test planning, test project risks are monitored and controlled, and corrective actions are initiated as needed.

Scope

The process area Test Monitoring and Control involves monitoring test progress and product quality against estimates, commitments, plans and expectations. It also includes reporting on test progress and product quality to the team and other stakeholders, performing milestone reviews, taking corrective actions when necessary, and managing the corrective actions to closure.

Goal and Practice Summary

G1 Monitor Test Progress

- P1.1 Monitor test planning parameters
- P1.2 Monitor test environment resources provided and used
- P1.3 Monitor test project risks
- P1.4 Report and discuss test progress
- P1.5 Conduct test progress milestone reviews

G2 Monitor Product Quality

- P2.1 Check against entry criteria
- P2.2 Monitor defects
- P2.3 Monitor product risks

P2.4 Monitor exit criteria
P2.5 Report and discuss product quality
P2.6 Conduct product quality milestone reviews

G3 Manage Corrective Actions to Closure
P3.1 Analyze issues
P3.2 Take corrective action
P3.3 Manage corrective action

Practices by Goal

G1 Monitor Test Progress

The actual progress and performance of testing is monitored and compared against the values in the test plan to identify significant deviations at an early stage so more effective corrective action can be taken.

P1.1 Monitor test planning parameters

Monitor the actual values of the test planning parameters against the test plan.

Example work products

1. Records of test performance
2. Updated task boards
3. Burndown charts
4. Records of significant deviations from plan

Subpractices

1. Monitor test progress against the test schedule

Examples of progress monitoring typically include:
- Periodically measuring the actual completion of test tasks, test (work) products and test milestones
- Comparing actual completion of test tasks, test (work) products and test milestones against the test schedule documented in the test plan
- Monitoring the progression of tasks and stories on the task board
- Using burndown charts to track progress within an iteration
- Identifying significant deviations from the test schedule estimates in the test plan

2. Monitor the test cost and expended test effort

Examples of cost and effort monitoring typically include:
- Periodically measuring the actual test costs, effort expended and staff assigned
- Comparing actual test cost, effort and staffing to the estimates documented in the test plan
- Identifying significant deviations from the test cost, effort and staffing in the test plan
- Comparing the actual attributes of the test work products and test tasks to the estimates documented in the test plan
- Identifying significant deviations from the estimates in the test plan

3. Document the significant deviations in the test planning parameters

P1.2 Monitor test environment resources provided and used

Monitor the test environment resources provided and used against those defined in the plan.

Example work products

1. Records of test environment resources provided and used
2. Burndown charts
3. Records of significant deviations from plan

Subpractices

1. Monitor test environment resources provided against plan
2. Monitor the actual usage of the provided test environment resources against plan, e.g., using specific burndown charts to monitor the test environment resources usage against those agreed during iteration planning
3. Identify and document significant deviations from the estimates in the plan

In a DevOps context, continuous monitoring with on-line dashboards will typically be applied to monitor the infrastructure, application and network in an automated way throughout the CI/CD pipeline.

P1.3 Monitor test project risks

Monitor test project risks against those identified in the test plan.

Example work products

1. Updated test project risk register
2. Records of project risk monitoring

Subpractices

1. Periodically review the test project risks in the context of the current status and circumstances, e.g., during daily stand-up meetings

Examples of test project risks to be reviewed include:

- Test commitments
- Stakeholder involvement
- Knowledge and skills of staff

2. Update the test project risk register as additional information becomes available, to incorporate any change
3. Communicate test project risk status to relevant stakeholders

P1.4 Report and discuss test progress

Periodically report, review and discuss test progress, performance and issues to keep stakeholders informed. Progress reviews are often held both internally with team members and externally with other stakeholders. These informal progress reviews are typically held regularly, e.g., daily (in an Agile and DevOps context), weekly or bi-weekly.

Example work products

1. Test progress report
2. Progress dashboard
3. Documented test progress review results, e.g. minutes of the progress meetings

Subpractices

1. Collect and analyze test progress monitoring measures

2. Regularly communicate status on test progress and performance to stakeholders, e.g., during daily stand-up meetings

Examples of stakeholders typically include:

- Project management
- Business management, e.g., represented by a product owner
- Team members

3. Regularly organize test progress review meetings with stakeholders
4. Discuss significant issues and deviations from the test plan
5. Document the results of the reviews, e.g., decisions made and corrective actions defined

P1.5 Conduct test progress milestone reviews

Review the accomplishments and progress of testing at selected test milestones, e.g., at the end of a test lifecycle phase or at the end of an iteration. Milestone reviews are typically formal reviews.

Example work products

1. Test milestone report
2. End of iteration progress dashboard
3. Documented milestone review results, e.g., minutes of the review meeting

Subpractices

1. Conduct test progress reviews with relevant stakeholders at meaningful points in the test schedule, such as the completion of selected stages or at the end of an iteration
2. Communicate accomplishments, test progress and performance status to stakeholders, e.g., by means of a formal report or by means of charts of a dashboard
3. Review the commitments, plan, status, and project risks of testing
4. Review the test environment resources
5. Identify, document and discuss significant test progress issues and their impacts
6. Document the results of the reviews, actions items, decisions and lessons learned
7. Update the test plan to reflect accomplishments and latest status

G2 Monitor Product Quality

Actual product quality is monitored against the quality measurements defined in the plan and the quality expectations, e.g., of the customer/user, to identify significant product quality issues at an early stage so more effective corrective action can be taken. In a DevOps context, this is a continuous process in which product quality is also monitored in production.

P2.1 Check against entry criteria

At the start of the test execution phase check the status against the entry criteria, also called definition-of-ready, identified in the test plan.

Example work products

1. Records of entry check (definition-of-ready)

Subpractices

1. Check the status against the entry criteria identified in the test plan
2. Identify and document significant deviations in compliance with entry criteria and initiate corrective action

P2.2 Monitor defects

Monitor measures of defects found during testing against expectations.

Example work products

1. Records of defect monitoring

Subpractices

1. Monitor measures of defects found and status against expectations

Examples of useful defect measures include [after Burnstein]:

- Total number of defects (for a component, subsystem or the system) outstanding per priority level
- Total number of defects found during the most recent test run per priority level
- Number of defects resolved/unresolved (for all levels of testing)
- Number of defects found for each given type
- Number of defects causing failures of severity level greater than "X"
- Number of defects/KLOC ("incident volume")
- Test pass/fail rate
- Defect discovery rate
- Actual number versus estimated number of defects (based on historical data)

2. Identify and document significant deviations from expectations for measures regarding defects found

P2.3 Monitor product risks

Monitor product risks against those identified during test planning.

Example work products

1. Updated test product risk register
2. Records of product risk monitoring

Subpractices

1. Periodically review the product risks in the context of the current status and circumstances with a selected set of stakeholders
2. Monitor changes and additions to the requirements to identify new or changed product risks
3. Revise the documentation of the product risks as additional information becomes available, e.g., as a result of exploratory testing, to incorporate a change of likelihood, impact and/or priority status
4. Monitor the (number of) product risks mitigated by testing
5. Communicate product risk status to relevant stakeholders

P2.4 Monitor exit criteria

Monitor the status of the exit criteria, also called definition-of-done, against those identified in the test plan.

Example work products

1. Records of exit criteria (definition-of-done) monitoring

Subpractices

1. Monitor the test process related exit criteria, e.g., test coverage, against plan
2. Perform customer satisfaction surveys to receive feedback on whether the product meets customer expectations, e.g., when validation is part of the set of exit criteria
3. Monitor the product quality related exit criteria against plan
4. Identify and document significant deviations in exit criteria status from plan

P2.5 Report and discuss product quality

Periodically report, review and discuss product quality to keep stakeholders informed. Product quality reviews are often held both internally with team members and externally with other stakeholders. These informal product quality reviews are typically held regularly, e.g., daily (in an Agile context), weekly or bi-weekly.

Example work products

1. Product quality report
2. Product quality dashboard
3. Documented product quality review results, e.g., minutes of the product quality meetings

Subpractices

1. Collect and analyze product quality monitoring measures
2. Regularly communicate status on product quality to stakeholders, e.g., during daily stand-up meetings

Examples of stakeholders typically include:

- Project management
- Business management, e.g., represented by a product owner
- Team members

3. Regularly organize product quality review meetings with stakeholders
4. Discuss significant product quality issues and deviations from expectations and plan
5. Document the results of the reviews, e.g., decisions made and corrective actions defined

P2.6 Conduct product quality milestone reviews

Review product quality status at selected test milestones, e.g., at the end of a test lifecycle phase or at the end of an iteration. Milestone reviews are typically formal reviews.

Example work products

1. Test milestone report
2. End of iteration product quality dashboard
3. Documented milestone review results, e.g., minutes of the review meeting

Subpractices

1. Conduct product quality reviews, e.g., sprint reviews, with relevant stakeholders at meaningful points in the test schedule, such as the completion of selected stages or at the end of an iteration
2. Communicate product quality status to stakeholders in a formal product quality report, through charts on a dashboard and / or product demonstrations.

Examples elements of a product quality test report include [after ISO 29119-3]:

- Management summary (including reporting period)

- Comprehensive assessment
- Summary of results and activities
- Testing completed against plan, including factors blocking completion
- Test measures (e.g., against exit criteria)
- New, changed and residual product risks

3. Review the status regarding defects, product risks and exit criteria
4. Identify, document and discuss significant product quality issues and their impacts
5. Document the results of the reviews, actions items, decisions and lessons learned
6. Update the test plan to reflect accomplishments and the latest status

G3 Manage Corrective Actions to Closure

Corrective actions are managed to closure when test progress or product quality deviate significantly from plan and/or expectations, to increase the probability that corrective actions are successfully executed and objectives will be met.

P3.1 Analyze issues

Collect and analyze the issues and determine corrective actions necessary to address them.

Example work products

1. List of issues needing corrective actions

Subpractices

1. Collect issues for analysis, e.g., during daily stand-up meetings

Examples of issues typically include:

- Significant deviations in actual test planning parameters from estimates in the test plan
- Commitments that have not been satisfied
- Significant changes in test project risk status
- Stakeholder representation or involvement issues
- Significant deviations in test environment implementation progress from plan
- Number, severity and priority level of defects found
- Status regarding exit criteria (definition-of-done)
- Significant changes in product risks

2. Analyze issues to determine need for corrective action

Note that corrective action is required when the issue, if left unresolved, may prevent testing, or even the project or team, from meeting its objectives.

P3.2 Take corrective action

Take corrective action as appropriate for the identified issues.

Example work products

1. Corrective action plan

Subpractices

1. Determine and document the appropriate actions needed to address the identified issues

Examples of potential actions include:

- Re-negotiating commitments
- Adding resources
- Changing the test approach. e.g., based on revised product risks
- Re-visiting the exit criteria (definition-of-done)
- Deferring release date
- Changing the scope of the project, e.g., delivering less functionality
- Removal of user stories from the iteration backlog

Note that many of the potential actions listed above will lead to a revised test plan.

2. Review and obtain agreement with team and relevant stakeholders on the actions to be taken
3. Re-negotiate commitments with stakeholders (both internally and externally)

P3.3 Manage corrective action

Manage the corrective action to closure.

Example work products

1. Corrective action results

Subpractices

1. Monitor corrective actions for completion, e.g., as backlog items via a task board and progress discussed at daily stand-up meetings
2. Analyze results of corrective actions to determine their effectiveness
3. Report progress on status of corrective actions

Process Area 2.4 Test Design and Execution (TDE)

Purpose

The purpose of Test Design and Execution is to improve test capability during test design and execution by identifying test conditions as a basis for developing test procedures, test charters and automated test scripts, performing a structured test execution process and managing test incidents to closure.

Value

Test Design and Execution aims to ensure adequate verification and validation of the quality of the software product, based on the identified product risks.

Introductory Notes

The test design and execution activities are based on product risks and the test approach as defined in the test plan. This process area covers the requirements for test analysis, test design, test implementation and test execution as defined in the fundamental test process [Veenendaal24]. In a sequential lifecycle these test phases are also performed largely sequentially. By contrast, when using Agile methodologies, they are not subsequent test phases, but rather performed in parallel, overlapping and iteratively.

High-quality requirements are crucial for test design and testing because they provide clear, complete, and unambiguous specifications that determine what to test. Test analysis involves studying and reviewing the test basis to determine its testability and understand what needs to be tested. During test design, test design techniques are used to derive and select test conditions and design test cases from requirements (e.g., user stories) and design specifications. During test implementation, the test conditions and test cases are translated into manual test procedures, test charters for exploratory testing and/or automated test scripts. Specific test data required to run the tests is created. During the test execution stage, tests are executed using test procedures, exploratory testing and/or automated test scripts. For incidents found, incident reports are created and logged using an incident management system. A process is established for the incident lifecycle and to manage each incident to closure.

Scope

The process area Test Design and Execution addresses the analysis of the test basis and the test design activity to derive and select test conditions. It also deals with test implementation, where manual test procedures, test charters and/or automated test scripts are developed and specific test data is created as well as subsequent test execution activities and incident management.

Goal and Practice Summary

G1 Perform Test Analysis and Design
- P1.1 Study and analyze the test basis
- P1.2 Identify and prioritize test conditions
- P1.3 Identify and prioritize test cases
- P1.4 Identify specific test data
- P1.5 Maintain horizontal traceability with requirements

G2 Perform Test Implementation
- P2.1 Implement test cases
- P2.2 Create specific test data
- P2.3 Specify smoke test

P2.4 Develop test execution schedule

G3 Perform Test Execution

P3.1 Perform smoke test
P3.2 Execute test cases
P3.3 Report test incidents
P3.4 Establish test log

G4 Manage Test Incidents to Closure

P4.1 Decide on disposition of test incidents
P4.2 Perform appropriate action to close test incidents
P4.3 Monitor and communicate the status of test incidents

Practices by Goal

G1 Perform Test Analysis and Design

During test analysis and design, the test approach is translated into test conditions and test cases, using test design techniques, to ensure adequate coverage, in line with the identified product risks.

P1.1 Study and analyze the test basis

The test basis is studied, analyzed and reviewed, e.g., during backlog refinement sessions, to identify issues and determine its testability, e.g., completeness, unambiguousness, and consistency.

Example work products

1. Test basis issue log

Subpractices

1. Identify the relevant test basis, e.g., requirements, user stories, acceptance criteria, design and interface specifications
2. Study and analyze the test basis
3. Review the test basis, possibly using a checklist
4. Discuss issues regarding the test basis with the document owner
5. Document test basis issues in a review log

P1.2 Identify and prioritize test conditions

Test conditions are identified and prioritized using the principles of test design techniques.

Example work products

1. Test conditions
2. Test design specification

Subpractices

1. Derive the test conditions from the test basis using the principles of test design techniques in line with the documented test approach

Examples of test design techniques [Veenendaal24] include:

- Equivalence Partitioning
- Boundary Value Analysis
- Decision Table Testing

- Statement Testing
- Branch Testing

Note, in an Agile and DevOps context, acceptance criteria defined for the user stories typically take over the role of traditional test conditions.

2. Prioritize the test conditions based on identified product risks
3. Document the test conditions in a test design specification or as acceptance criteria in an Agile or DevOps context

Examples of elements of a test design specification include [after ISO 29119-3]:

- Test design specification identifier
- Items and/or features to be tested
- Objective
- Approach refinements
- Test conditions
- Priority
- Traceability

4. Review the test conditions with stakeholders
5. Revise the test conditions as appropriate, e.g., whenever the requirements change

P1.3 Identify and prioritize test cases

Test cases are identified and prioritized using test design techniques or approaches such as Acceptance Test-Driven Development (ATDD) or Behavior-Driven Development (BDD).

Example work products

1. Test cases

Subpractices

1. Derive test cases from test conditions / acceptance criteria to provide their coverage using test design techniques or approaches such as ATDD or BDD

   A test case consists of a set of input values, execution preconditions, actions (where applicable), expected results and execution post conditions.
2. Prioritize test cases based on identified product risks and business value of the requirements / user stories they are covering
3. Document test cases as appropriate, e.g., by means of a test case specification or in Given/When/Then format (Gherkin language) when using BDD or ATDD
4. Review test cases with stakeholders
5. Revise test cases as necessary

P1.4 Identify specific test data

Specific test data necessary to support the execution of test cases is identified.

Example work products

1. Test data specification

Subpractices

1. Identify the specific test data required to implement and execute test cases

2. Document the specific test data

P1.5 Maintain horizontal traceability with requirements

Traceability between the requirements, e.g., user stories, and the test conditions / test cases is established and maintained.

Example work products

1. Requirements vs. test conditions / test cases traceability

Subpractices

1. Maintain requirements traceability to ensure that the source of test conditions / test cases is documented
2. Establish requirements vs. test conditions / test cases traceability
3. Set up the traceability such that monitoring requirements coverage is facilitated

G2 Perform Test Implementation

During test implementation, the test procedures, test charters and/or automated test scripts, including the smoke test, are developed to allow for an effective and efficient execution of the tests. Test data is created and a test execution schedule defined.

P2.1 Implement test cases

Test conditions and test cases are implemented for execution, including those for regression testing, in the format of test procedures, test charters and/or automated test scripts, in line with the defined test approach.

Example work products

1. Test procedure specifications
2. Test charters
3. Automated test scripts

Subpractices

1. Develop test procedures based on the identified test cases, ordering them for execution and including other information needed for test execution

   Examples of elements of a test procedure specification include [after ISO 29119-3]:
   - Test procedure specification identifier
   - Test sets (unique identifier, objective, priority, test cases)
   - Test procedures (unique identifier, objective, priority, start up, test cases to be executed, relationship to other procedures, stop and wrap up)
2. Develop test charters based on the identified test conditions as a basis for performing exploratory testing
3. Develop automated test scripts, e.g., for regression testing, based on the identified test conditions and test cases
4. Prioritize test procedures, test charters and test scripts based on identified product risks
5. Review test procedures, test charters and test scripts with stakeholders
6. Revise test procedures, test charters and test scripts as necessary

P2.2 Create specific test data

Specific test data is created.

Example work products

1. Specific test data

Subpractices

1. Create specific test data (as specified during the test design activity) required to perform the automated and manual tests
2. Archive the set of specific test data to allow the initial situation to be restored when required

Refer to the practice P3.2 “Perform generic test data management” from the Test Environment process area for managing the created test data.

P2.3 Specify smoke test

The smoke test is specified. A smoke test is used to decide, at the beginning of test execution, whether the test object is ready for detailed and further testing. It is also a common practice with continuous integration to check critical functionalities after each new build.

Example work products

1. Smoke test checklist
2. Smoke test procedure or script

Subpractices

1. Define a list of checks to be executed during the smoke test using the entry criteria as defined in the test plan as an input

Examples of checks to be part of a smoke test include:

- All required major functionalities are available
- Representative functionalities are available and pass a set of basic scenario test cases
- Interfaces are available with other components or systems that will be tested
- The documentation is complete for the available functionality including a test release note

2. Develop a manual smoke test procedure or automated smoke test script based on the identified checks
3. Review the smoke test procedure or automated test script with stakeholders
4. Revise the smoke test procedure or automated test script as necessary.

P2.4 Develop test execution schedule

A test execution schedule is developed that describes the sequence in which the tests, especially the manual tests, will be executed.

Example work products

1. Test execution schedule, possibly using a task board

Subpractices

1. Investigate dependencies between the tests to be executed
2. Schedule the test procedures and exploratory tests using their priority level as a main driver
3. Assign test resources to perform the execution of the test procedures and exploratory tests
4. Review the test execution schedule with stakeholders
5. Revise the test execution schedule as necessary

G3 Perform Test Execution

Automated test scripts and manual tests are executed according to the test execution schedule to verify and validate the quality of the software product. Test incidents are reported and test logs are created.

P3.1 Perform smoke test

Perform the smoke test (intake test) to decide whether the test object is ready for comprehensive testing and/or to check critical functionalities after a new build.

Example work products

1. Test log
2. Incident reports

Subpractices

1. Perform the smoke test as developed during test implementation
2. Document the results of the smoke test by means of a test log, based on the test log standard
3. Log incidents when a discrepancy is observed

Note, this practice is highly related to the practice P2.4 “Perform test environment smoke test” from the process area 2.5 Test Environment. The smoke test on the test object and test environment can possibly be combined.

P3.2 Execute test cases

The test cases are run (either manually using documented test procedures and/or exploratory testing, or via test automation using pre-defined test scripts) according to the defined test execution schedule. This practice includes running test cases for confirmation and regression testing.

Example work products

1. Test results

Subpractices

1. Execute test cases using documented test procedures, exploratory testing and/or automated test scripts
2. Record actual results
3. Compare actual results with expected results
4. Repeat test activities after the receipt of a fix by performing confirmation testing
5. Perform regression testing as appropriate to ensure that defects have not been introduced or uncovered as a result of the changes made.

P3.3 Report test incidents

Discrepancies between the actual and expected results are reported as test incidents.

Example work products

1. Test incident reports
2. Criteria for logging test incidents

Subpractices

1. Log a test incident when a discrepancy is observed
2. Analyze the test incident for further information on the issue
3. Determine the suspected cause of the test incident

4. Assign an initial priority and severity level to the test incident
5. Record the test incident using an incident report template

Examples of elements of a test incident report include [after ISO 29119-3]:

- Identifier
- Summary
- Date and time
- Originator
- Description of the test incident (input, expected results, actual results, anomalies, test case, environment, reproducibility)
- Suspected cause
- Priority
- Severity

Note, if not all test incidents found are logged, e.g., in an Agile or DevOps context, criteria must be developed that define which test incidents should be logged and which ones should not.

6. Review the test incident report with stakeholders. Refer to [Black] for a set of best practices for test incident reporting also to be used during reviewing
7. Store the test incident in a central repository

P3.4 Establish test log

Test logs are established (written or generated) to provide a record of relevant details about the execution of the tests.

Example work products

1. Test logs

Subpractices

1. Collect test execution data
2. Document the test execution data by means of a test log, based on a test log standard

Examples of elements of a test log include [after ISO 29119-3]:

- Test log identifier
- Date and Time
- Items being tested
- Environment in which the testing has been executed
- Execution description (reference to test cases being executed, test results, anomalous events, incident report identifiers)
- Impact

3. Review the test log with stakeholders

G4 Manage Test Incidents to Closure

Test incidents are managed and resolved as appropriate to minimize the impact of disruptions and meet product quality objectives and customer commitments more effectively.

P4.1 Decide on disposition of incidents

Appropriate actions on test incidents are decided upon.

Example work products

1. Decision log regarding test incidents
2. Updated test incident report

Subpractices

1. Review and analyze test incidents
2. Revisit the priority and severity level of test incidents
3. Determine actions to be taken for test incidents

Examples of decisions that can be made include:

- Rejected, test incident is not a defect
- Deferred, test incident is declined for repair now but may be dealt with during a later stage
- Fix, test incident is accepted and shall be repaired

4. Record the decision including rationale and update (the status of) the test incident report
5. Assign the test incident to the appropriate group, e.g., development, to perform appropriate actions

P4.2 Perform appropriate actions to close test incidents

Appropriate actions are taken to fix, confirmation test and close test incidents or defer them to a future iteration or release.

Example work products

1. Test log (including test results)
2. Updated test incident report

Subpractices

1. Repair the test incident which typically involves updating software code and possibly documentation
2. Record information on the repair action and update (the status of) the test incident report
3. Perform confirmation testing, and possibly regression testing, to confirm the fix of the test incident
4. Record information on the confirmation testing in a test log and update (the status of) the test incident report
5. Formally close the test incident provided confirmation testing was successful and meets the criteria for closure

P4.3 Monitor and communicate the status of test incidents

The status of test incidents is monitored and communicated, and appropriate actions are taken.

Example work products

1. Status report test incidents
2. Updated test incident report

Subpractices

1. Monitor progress for test incidents throughout their life
2. Take appropriate action, such as escalate to appropriate level of management as needed, e.g., when a test incident that needs repair has the same status for longer than an agreed period of time
3. Provide and discuss status reports on test incidents with stakeholders and team

Process Area 2.5 Test Environment (TEV)

Purpose

The purpose of Test Environment is to establish and maintain an adequate environment, including test data, in which it is possible to execute the tests in a manageable and repeatable way.

Value

Test Environment aims to facilitate and support an effective and efficient execution of tests, by providing an adequate test environment and supporting generic test data.

Introductory Notes

A managed and controlled test environment is indispensable for any testing. It is also needed to obtain test results under conditions which are as close as possible to the 'real-life' situation. This is especially true for higher level testing, e.g., at system and acceptance test level. Furthermore, at any test level the reproducibility of test results should not be endangered by undesired or unknown changes in the test environment.

Identification and specification of test environment requirements is performed early in the project. In Agile, a so-called initial iteration (iteration-0) is sometimes applied where the elicitation and specification of the test environment requirements is performed. The requirements specification is reviewed to ensure its correctness, suitability, feasibility and accurate representation of a 'real-life' operational environment. Early test environment requirements specification has the advantage of providing more time to acquire and/or develop the required test environment and its components.

Subsequently, the test environment is implemented in line with the test environment requirements. This often includes the implementation of service virtualization and development of scripts to automatically set up, configure, and tear down test environments. Once the implementation is done, a test environment smoke test is performed to determine whether the test environment is ready to be used for comprehensive testing.

Performing test environment management includes practices such as managing access to the test environment, providing technical support on progress-disturbing issues during test execution, monitoring resource usage, and after testing cycles, decommissioning and cleaning up test environments.

As part of the Test Environment process area, the requirements regarding generic test data, and the creation and management of these test data are covered. Whereas specific test data is defined during test design and analysis, generic test data is often defined and created as a separate activity. Generic test data is (re-)used by many, and provides overall background data that is needed to execute the functionalities of the system and perform non-functional testing. Generic test data can come from a variety of sources, such as real-world datasets, mock data, or data that is generated specifically for testing purposes. Properly defining, creating and managing generic test data is crucial for ensuring thorough and effective software testing.

Scope

The process area Test Environment addresses all activities for specifying test environment requirements, and implementing, managing and controlling the test environment. It also encompasses specifying the requirements for the generic test data, and creating (or obtaining) and managing the generic test data.

Goal and Practice Summary

G1 Develop Test Environment Requirements

P1.1 Elicit test environment and generic test data needs

P1.2 Develop test environment and generic test data requirements
P1.3 Analyze test environment and generic test data requirements

G2 Perform Test Environment Implementation
P2.1 Implement the test environment
P2.2 Create generic test data
P2.3 Specify test environment smoke test
P2.4 Perform test environment smoke test

G3 Manage and Control Test Environments
P3.1 Perform test environment management
P3.2 Perform generic test data management
P3.3 Coordinate the availability and usage of the test environments
P3.4 Report and manage test environment incidents

Practices by Goal

G1 Develop Test Environment Requirements

Test environment and test data needs, expectations and constraints are collected and translated into test environment and test data requirements to ensure a mutual understanding of the requirements, needs and expectations.

P1.1 Elicit test environment and generic test data needs

Elicit test environment needs, expectations and constraints, including those for generic test data.

Example work products

1. Test environment needs, expectations and constraints
2. Test data needs, expectations and constraints

Subpractices

1. Study the test approach for test environment and test data implications
2. Engage testing representatives for eliciting test environment and generic test data, needs, expectations and constraints
3. Document the test environment and generic test data, needs, expectations and constraints

P1.2 Develop test environment and generic test data requirements

Transform the test environment and test data needs into prioritized requirements.

Example work products

1. Prioritized test environment requirements
2. Prioritized generic test data requirements

Subpractices

1. Translate the test environment and generic test data needs, expectations and constraints, into test environment and generic test data requirements
2. Define when each test environment item and the generic test data are needed
3. Establish and maintain a prioritization of test environment and generic test data requirements to ensure critical test environment items and generic test data are available on time

P1.3 Analyze test environment and generic test data requirements

Analyze and review the requirements to ensure they are necessary, sufficient and feasible.

Example work products

1. Test environment requirements review log
2. Generic test data requirements review log

Subpractices

1. Analyze test environment and generic test data requirements to determine whether they fully support the test lifecycle and test approach

Examples of test environment requirements analysis include:

- Mapping of test environment requirements to test levels
- Mapping of test environment requirements to test types
- Mapping of test environment requirements to the CI/CD pipeline

2. Analyze test environment requirements to ensure that, together, they sufficiently represent the operational environment, especially for higher test levels
3. Review the test environment and generic test data requirements with stakeholders
4. Revise the test environment and generic test data requirements as necessary

G2 Perform Test Environment Implementation

The test environment and generic test data requirements are implemented to facilitate and support test execution.

P2.1 Implement the test environment

Implement the test environment in line with the test environment requirements.

Example work products

1. Operational test environment
2. Test results for test environment components

Subpractices

1. Implement the test environment as specified, e.g., setting up servers, workstations, and ensuring that the required software, tools, and libraries are installed. The test environment may also be implemented using Infrastructure-as-Code and Configuration-as-Code for defining environment components, variables and configurations
2. Ensure that the correct versions of software, operating systems, and third-party tools are installed, particularly when testing multiple versions or configurations
3. Implement service virtualization as specified, e.g., of machines, servers, stubs and drivers
4. Write scripts to automatically set up, configure, and tear down test environments, e.g., in a DevOps context Infrastructure-as-Code is often used to define and implement environment components and Configuration-as-Code to define and implement environment variables and configurations
5. Develop supporting documentation, e.g., installation, operation and maintenance documentation
6. Revise and maintain the test environment as necessary, thereby coordinating with stakeholders for any changes to the environment (e.g., infrastructure updates and software upgrades)

P2.2 Create generic test data

Generic test data is created in line with the generic test data requirements.

Example work products

1. Generic test data

Subpractices

1. Create or obtain generic test data required to support the execution of the tests as specified. Test data may possibly be created using AI techniques
2. Refresh the test data regularly to ensure they remain up-to-date and represent various real-world scenarios
3. Anonymize or mask sensitive data in line with the privacy policy when 'real-life' data is used as a source
4. Archive the set of generic test data to allow the initial situation to be restored when required
5. Ensure the integrity and correctness of test data, so that they are valid and can support accurate test results.
6. Verify that the generic test data cover all needs and requirements.
7. Develop management protocols for generic test data

P2.3 Specify test environment smoke test

The test environment smoke test, to decide whether the test environment is ready to be used for testing, is specified.

Example work products

1. Test environment smoke test checklist
2. Test environment smoke test procedure or script

Subpractices

1. Define a list of checks to be carried out during the smoke test of the test environment
2. Develop the manual test environment smoke test procedure or automated test script based on the identified set of checks
3. Review the test environment smoke test procedure or automated test script with stakeholders
4. Revise the test environment smoke test procedure as necessary

Note, this practice is highly related to the practice P2.3 "Specify smoke test" from the process area 2.4 Test Design and Execution.

P2.4 Perform test environment smoke test

The test environment smoke test is performed to determine whether the test environment is ready to be used for comprehensive testing.

Example work products

1. Test environment smoke test log
2. Incident reports

Subpractices

1. Perform the test environment smoke test using the manual test procedure or the automated test script to decide if the test environment is ready to be used for testing

2. Document the results of the test environment smoke test by means of a test log, based on the test log standard
3. Log incidents if a discrepancy is observed

Refer to the practice P3.3 “Report test incidents” from the process area 2.4 Test Design and Execution for more information on incident logging.

Note, this practice is highly related to the practice P3.1 “Perform smoke test” from the process area 2.4 Test Design and Execution; the smoke test on the test object and test environment can possibly be combined.

G3 Manage and Control Test Environments

Test environments and test data are managed and controlled to allow for uninterrupted test execution.

P3.1 Perform test environment management

Systems management is performed on the test environments to effectively and efficiently support the test execution process.

Example work products

1. System management log file
2. Test log

Subpractices

1. Install components needed, e.g., for a specific test session
2. Manage access to the test environment by providing log-in details and ensuring proper permissions are in place
3. Manage environment-specific settings like API end-points, feature flags, or other configuration variables
4. Provide technical support on progress blocking issues during test execution
5. Provide logging facilities, which can be used afterwards to analyze test results, and are used to regularly monitor for potential issues
6. Monitor resource usage (e.g., CPU, memory, disk space) to ensure the test environment remains operational and performant
7. Ensure test environments adhere to security policies and best practices, including patching vulnerabilities and applying updates regularly
8. Integrate test environment provisioning CI/CD pipelines to dynamically create test environments for different stages of the pipeline
9. Ensure proper backups of the test environments are prepared for recovery scenarios
10. After testing cycles, decommission and clean up test environments to free up resources

P3.2 Perform generic test data management

Test data is managed and controlled to effectively and efficiently support the test execution process.

Example work products

1. Archived generic test data
2. Generic test data management log file

Subpractices

1. Manage generic test data, e.g., with respect to storage resources needed in an organized and secure repository to ensure easy retrieval and management during testing cycles

2. Remove old or unused generic test data from environments to optimize storage and ensure that obsolete data does not interfere with new tests
3. Ensure that the generic test data complies with legal and regulatory requirements regarding privacy and data protection
4. Generate reports on the usage, coverage, and quality of generic test data to inform stakeholders and improve the generic test data management process
5. Ensure that only authorized testers and developers can access specific datasets to maintain security
6. Automatically provide the correct generic data sets to test environments to save time and effort, often integrated into CI/CD pipelines
7. Archive and restore generic test data and other files on a regular basis and as appropriate for a test session

P3.3 Coordinate the availability and usage of the test environments

The availability and usage of the test environment by multiple groups is coordinated to achieve maximum efficiency.

Example work products

1. Test environment reservation schedule

Subpractices

1. Make reservations of the test environments in the reservation schedule
2. Identify specific test environment components needed when making a reservation
3. Discuss conflicting reservations with involved groups and stakeholders
4. Define a test environment reservation schedule for the upcoming period
5. Decommission the test environment correctly after usage, e.g., by making sure it is in a known state and test files are removed

P3.4 Report and manage test environment incidents

Problems that occur while using the test environment and/or generic test data are formally reported as incidents and are managed to closure.

Example work products

1. Test environment / generic test data incident reports
2. Decision log regarding test environment / generic test data incidents

Subpractices

1. Log test environment incidents when a problem is observed
2. Record test environment incidents using an incident report template
3. Manage test environment incidents to closure

Refer to the process area 2.4 Test Design and Execution for practices and subpractices covering incident reporting and management.

Process Area 2.6 Implementation and Habit (IH)

Purpose

The purpose of Implementation and Habit is to establish and maintain an organizational environment in which performing the test processes and test process improvement are supported and can successfully be institutionalized at project, team or organizational unit level.

Value

Implementation and Habit aims to provide the practices that are essential for being able to execute the test processes and for improvements to last, and, over time, to become habitual.

Introductory Notes

This process area describes the implementation of practices that address the various aspects of institutionalization. Institutionalization is an important concept in process improvement. Institutionalization implies that the test process is ingrained in the way the work is performed, is habitual and there is commitment to and consistency in performing the process. An institutionalized process is more likely to be sustained during times of stress. This process area addresses the initial step for institutionalization, to be further enhanced at TMMi level 3. An institutionalized process is a process that accomplishes the work necessary to produce work products. It is planned and executed in accordance with policy, employs skilled people and has adequate resources to produce controlled outputs. At TMMi level 2, a process may be instantiated by a project, team, or organizational unit.

The Implementation and Habit process area specifically addresses governance, implementation of the infrastructure and configuration management.

Governance contains the practices (senior) management perform to promote the way testing work is accomplished that is relevant and important to the business and the organization. Visible and active management involvement is critical to the success of both test performance improvement and test process implementation.

The implementation of the infrastructure refers to everything that is needed to implement and perform the set of processes and for them to become habitual. The infrastructure includes process descriptions, resource availability and training. Process descriptions should be clear and concise, and typically not require a significant amount of administration. Without an infrastructure, processes may not be followed and will not become habitual. The infrastructure basically provides the ability to perform the processes.

Configuration management provides a discipline for identifying, controlling, and managing changes to test work products such as test plans, test scripts, test results, test logs, and test reports as configuration items.

Scope

The process area Implementation and Habit addresses the governance (the role and responsibilities for senior management), institutionalization (things that are needed to be able to perform the processes) and configuration management (supporting the ability to manage the integrity of the test work products).

Goal and Practice Summary

G1 Establish Governance

P1.1 Define organizational test directives

P1.2 Identify information needs and use collected information
P1.3 Assign authority for test process improvement
P1.4 Provide resources for test process improvement

G2 Implement Infrastructure
P2.1 Develop and maintain test processes
P2.2 Train people
P2.3 Provide resources to perform the test processes
P2.4 Monitor implementation

G3 Manage Configurations
P3.1 Develop a configuration and change management process
P3.2 Identify test items to be placed under configuration management
P3.3 Develop and use versions of test items under configuration management

Practices by Goal

G1 Establish Governance

Provides explicit guidance to senior management on their role in sponsorship and governance of test performance, test processes and related activities to increase the likelihood that implemented test processes support and contribute to the success of the business.

P1.1 Define organizational test directives

Senior management defines, keeps updated and communicates organizational test directives for test process implementation and test performance improvement based on organizational needs and objectives.

Example work products

1. Organizational test directives
2. Test performance improvement directives
3. Records of communication

Subpractices

1. Define and communicate organizational test directives to drive the organizational test policy
2. Define and communicate test performance improvement directives, e.g., which test process areas should be prioritized, to drive the test policy
3. Review test process implementation and test performance improvement objectives defined in the organizational test policy
4. Review and update organizational test directives and test performance improvement directives on a periodic or event-driven basis

Refer to process area 2.1 Test Policy and Strategy for more information on the test policy.

P1.2 Identify information needs and use collected information

Senior management identifies their information needs and uses the collected information to provide governance and oversight of effective test process implementation and test improvement.

Example work products

1. Management information needs

2. Review results

Subpractices

1. Identify information needs related to test process capability, test improvement and test performance objectives to drive the test performance indicators
2. Review the defined test performance indicators
3. Review accomplishments, status, and results of the test process implementation and test improvement activities using the measurements of test performance indicators

Refer to process area 2.1 Test Policy and Strategy for more information on test performance indicators.

P1.3 Assign authority for test process improvement

Senior management assigns authority and holds people accountable for adhering to organizational directives defined in the test policy and achieving test process implementation and test performance improvement objectives.

Example work products

1. Assignment of responsibility
2. Rewards, recognition and incentives

Subpractices

1. Identify and record authority, including responsibilities and decision making
2. Identify individuals who lead the test improvement process and individuals who perform test process roles
3. Provide rewards, recognition and incentives for test process improvement

P1.4 Provide resources for test process improvement

Senior management ensures resources are provided for developing, improving, and evaluating adherence to the expected test processes.

Example work products

1. Record of allocation of funding and resources
2. Records of reviews and communication

Subpractices

1. Provide and approve the funding and resources to develop, improve and monitor the test process
2. Review and communicate assignment of needed funding, personnel, and resources to develop, improve and monitor the test process
3. Review the allocation of budget and resources

G2 Implement infrastructure

The implementation of the infrastructure covers what is needed to implement, perform and sustain the set of test processes to increase the ability to achieve goals and objectives efficiently and effectively.

P2.1 Develop and maintain test processes

Develop and keep test processes up-to-date.

Example work products

1. Documented test processes

Note, test processes at TMMi level 2 are not necessarily organizational test processes. They can be defined at project, team or organizational unit level and as such may vary across the organization.

Subpractices

1. Study and understand the process
2. Document the process, which typically includes a purpose, inputs, a sequence of steps or activities, outputs, and roles including stakeholders and responsibilities
3. Review and update the documented processes with stakeholders
4. Communicate and make documented processes available

P2.2 Train people

Ensure that people have the necessary skills and expertise to perform or support the processes.

Example work products

1. Process training materials
2. Training records

Subpractices

1. Develop or obtain training material for the processes
2. Provide appropriate training to those that will perform the process
3. Provide process overview training to stakeholders who interact with those performing the process
4. Keep records for those who completed a training course or any other training activity

P2.3 Provide resources to perform the test processes

Adequate resources are provided for executing the test processes, developing work products, and providing the services defined in the processes.

Example work products

1. Budget for resources
2. Tools

Subpractices

1. Assign responsibility for performing the specific tasks of the test processes
2. Provide sufficient time to perform the test processes and develop the work products
3. Ensure that skilled people, either trained or experienced, are available to perform the test processes
4. Ensure that relevant stakeholders are available to provide support for the various testing activities throughout the test process
5. Acquire or build tools to support the test processes as needed

P2.4 Monitor implementation

The implementation of the test processes on projects and teams is monitored.

Example work products

1. Status report and results of monitoring test processes in projects and teams

Subpractices

1. Monitor projects and teams for their use of the test processes

2. Review test process artifacts created by a project or team to ensure compliance
3. Identify, document, and monitor to closure issues related to implementing the test process

G3 Manage Configurations

Manage the integrity of test work products using configuration identification, version control and change control to reduce the loss of work and increase the ability to deliver and use the correct version of test work products.

P3.1 Develop a configuration and change management system

Develop or re-use, e.g., from software development, a configuration and change management process.

Example work products

1. Configuration and change management process

Subpractices

1. Describe how test items and changes to them are controlled and managed throughout their lifecycle
2. Establish a method to manage multiple levels of control

Examples of levels of control include:

- Anyone can make changes
- Authors or owners control changes
- A designated authority authorizes and controls changes

3. Define naming conventions for test work product documents
4. Provide access control to ensure authorized access to the configuration management system
5. Provide functionalities to store and retrieve test configuration items in/from the configuration management system
6. Preserve the contents of the configuration management system, e.g., by means of back-up and restore, archive and recovery

P3.2 Identify test items to be placed under configuration management

Based on criteria defined in the process, configuration test items are identified that are to be controlled and managed.

Example work products

1. Identified test configuration items

Subpractices

1. Assign unique identifiers to each test configuration item
2. Describe the characteristics for each test configuration item

Examples of characteristics to describe a test configuration item include:

- Objective evaluations
- Owner or author
- Type of test work product
- Purpose of the item
- Relationship to other items

P3.3 Develop and use versions of test items under configuration management

Develop, keep updated and use records describing versions of test items under configuration management.

Example work products

1. Test configuration items (test work products) and their versions
2. Revision history or change log of test configuration items
3. Change request records

Subpractices

1. Perform version control by identifying the correct version of the test configuration item
2. Record configuration management actions in sufficient detail so the content and status of each test configuration item is known, and previous versions can be recovered
3. Ensure that stakeholders have access to, and knowledge of, the configuration status of test configuration items, e.g., by generating automatic alerts
4. Specify the differences between versions of test configuration items
5. Update the status and history, e.g., changes, of each test configuration item as necessary

TMMi Level 3 Defined

At TMMi level 3 testing becomes a defined process. Organizational standard test processes and test process assets are developed, maintained and used by all projects and teams. A fully defined process has enough detail that it can consistently be performed by trained and skilled people and is both persistent and habitual. Key terms that describe the objectives at TMMi level 3 are institutionalization and early involvement. Institutionalization, building on the practices of TMMi level 2, is now further enhanced by practices such as test organization, test career paths, test training program and a standard organizational test process. Early involvement and early defect detection are covered by practices centered on the integrated lifecycle model and peer reviews.

At TMMi level 3, testing is no longer confined to a phase that follows coding. It is fully integrated into the development lifecycle and transformed into proactive, early-stage analysis, which, in an Agile context, typically starts during iteration planning. The organization's set of standard test processes, which is the basis for TMMi maturity level 3, is established and improved over time. A test organization is implemented, e.g., in the form of a test competence center. Test process improvement is fully institutionalized as part of the test organization's set of practices. A dedicated test training program exists, and testing is perceived by all as being a profession.

Organizations at TMMi level 3 understand the importance of early defect detection and the role of reviews in quality control. Peer reviews take place throughout the development lifecycle and their approach is coordinated with dynamic testing. Test professionals are involved in requirements reviews, e.g., backlog refinement sessions. Whereas testing at TMMi level 2 mainly focuses on functional testing, at TMMi level 3, driven by business objectives, test improvements also include establishing a capability for non-functional testing, e.g., security, performance and usability.

A critical distinction between TMMi maturity level 2 and 3 is the scope of the standards, test process descriptions, and test process assets. At TMMi maturity level 2, these may still be different per project and team. At TMMi level 3 the way of working is aligned across projects and teams, which also allows for easier transfer of staff between projects or teams. Differences in the way of working that exist because of specific needs are now managed through a set of tailoring guidelines. Another distinction of TMMi level 3 is that process descriptions are typically described in slightly more detail than at TMMi level 2 because the target audience is wider. Consequently, at TMMi level 3, the organization may need to revisit not only the test policy and test strategy, but also the process descriptions of the TMMi level 2 processes.

The process areas at TMMi level 3 are:

3.1 Test Organization

3.2 Test Training Program

3.3 Test Process and Integration

3.4 Non-Functional Testing

3.5 Peer Reviews

Each of these process areas is described in more detail in the sections hereafter.

Process Area 3.1 Test Organization (TO)

Purpose

The purpose of the Test Organization process area is to identify and organize a group of skilled professionals responsible for building and maintaining a test competence across the organization. The test organization manages improvements to the organization's test process and test process assets based on a thorough understanding of their current strengths and weaknesses.

Value

Test Organization aims to ensure that the organization has the necessary testing resources, skills, and infrastructure to perform testing in projects and teams. It is the driver of test improvement and innovation towards achieving software product quality and testing excellence.

Introductory Notes

Establishing a test organization implies a commitment to better testing and higher-quality software. To initiate the formation, senior management, building on the governance established at TMMi level 2, shall support the decision to establish a test organization and commit resources to it. It also requires leadership in areas that relate to testing and quality issues. The staff members of such an organization are called test specialists. A test organization is the embodiment of effective relationships between test specialists, test facilities and project/team-related test activities to achieve a high standard in testing.

The Test Organization process area is often misunderstood. Many misconstrue this process area as TMMi requiring an independent test group or department that performs independent testing. Although this is one option, there are other organizational models that comply with the TMMi requirements, such as a test competence center, which is much preferred in an Agile context. Typical tasks and responsibilities of a test competence center are establishing, managing and improving test processes, defining a test automation strategy, and providing resources with testing experience, knowledge and skills to projects and teams. A test competence center commonly has ownership of test methodologies, strategies and processes. Note, that a tester is typically part of an Agile or DevOps team on a day-to-day basis, but it is common for a tester to also belong to a test organization, such as a test competence center or test guild [TMMiAgile], [TMMiDevOps].

Testing is regarded as a professional field, and the test organization is recognized as essential. The test specialists possess detailed and specialized knowledge and skills in test engineering, test management, and the application domain. Test functions and test career paths are defined, and supported by a test training program. For each test function the typical tasks, responsibilities, authorities, required knowledge, skills and test training are specified. As a result, the process areas "Test Organization" and "Test Training Program" are closely related and interdependent. One of the principal objectives of the training program is to support the test organization in training test specialists. The test specialists are assigned to a specific test function and are dedicated to establishing awareness of, and achieving, product quality goals. They have the responsibility to verify and validate the system. The test roles, responsibilities and tasks for other staff members (non-test specialists) are specified.

While test process improvement is typically a project at TMMi level 2, at TMMi level 3 it becomes a well-organized and structured process within the test organization. Typically, a test competence manager is assigned the responsibility for facilitating and managing the test process improvement activities. Sometimes a Test Process Group is established. Candidates for test process improvements are obtained from various sources, including lessons learned, retrospectives (both bottom-up) and assessment results (top-down).

Planning is required to ensure that resources are available and test process improvement efforts across the organization are managed. Planning for test process improvement results in a process improvement plan.

Scope

The process area Test Organization defines the functioning (tasks, responsibilities, reporting structure) and position of a test group in the overall organization. Test roles, functions, and career paths are defined to support testing as a professional discipline. Test process improvement is a key activity of the test organization. Test process improvement encompasses assessing the current test process, using lessons learned to identify possible test improvements, and implementing them.

Goal and Practice Summary

G1 Establish a Test Organization
- P1.1 Define the test organization
- P1.2 Obtain commitments for the test organization
- P1.3 Implement the test organization

G2 Establish Test Functions for Test Specialists
- P2.1 Identify test functions
- P2.2 Develop job descriptions
- P2.3 Assign staff members to test functions

G3 Establish Test Career Paths
- P3.1 Establish test career paths
- P3.2 Develop personal test career development plans

G4 Determine, Plan and Implement Test Process Improvements
- P4.1 Assess the organization's test process
- P4.2 Analyze lessons learned using the organizational test process
- P4.3 Identify the organization's test process improvements
- P4.4 Plan test process improvements
- P4.5 Implement test process improvements
- P4.6 Manage the deployment

Practices by Goal

G1 Establish a Test Organization

A test organization is defined and established which develops and manages testing knowledge, processes and technology to create a test competence at a higher level than that of individual projects or teams, enabling test specialists to collaborate and share best practices across the organization.

P1.1 Define the test organization

A test organization is defined and agreed upon by the stakeholders.

Example work products

1. Test organization description

Subpractices

1. Identify the most suitable type of test organization based on the test policy and test strategy

Examples of types of test organization include:

- Independent test organization
- Test service center
- Testing Center of Excellence (TCoE)
- Test competence center
- Test guild

2. Define the test organization

Examples of topics to be addressed when defining a test organization include:

- Formal position in the overall organization
- Tasks, competences and responsibilities of the test organization
- Level of independence
- Reporting structure
- Staffing and resources

Note, ideally, the test organization should be a separate organizational entity or function. However, this is not always possible or practical given the size of the organization, risk level of the type of systems being developed and/or level of resources available.

3. Review the test organization description with stakeholders

P1.2 Obtain commitments for the test organization

Commitments for implementing and supporting the test organization are established and maintained.

Example work products

1. Documented commitments

Subpractices

1. Review and obtain agreement on commitments regarding the test organization with senior management
2. Document all commitments

P1.3 Implement the test organization

The test organization is implemented in the organization, based on the committed test organization description.

Example work products

1. Test organization implementation status report

Subpractices

1. Monitor implementation progress and commitments
2. Identify, document, and monitor to closure issues with implementing the test organization
3. Ensure that the results of implementing the test organization satisfy the organizational goals and objectives

G2 Establish Test Functions for Test Specialists

Test functions with accompanying job descriptions are established and assigned to the test specialists to allow for objective performance evaluations and to identify areas for development and training.

P2.1 Identify test functions

A set of test functions is identified.

Example work products

1. List of identified test functions

Subpractices

1. Analyze the test policy, test strategy and standard test process for typical test roles
2. Identify a set of test functions that cover the typical test roles

Examples of test functions include:

- Junior test engineer
- Senior test engineer
- Test manager
- Test architect
- Test consultant

3. Identify test functions for specialized areas

Examples of test functions for specialized areas include:

- Test automation architect
- Test automation engineer
- Performance test engineer
- Security test engineer
- Usability test engineer
- Test process improvement officer

P2.2 Develop job descriptions

For the identified test functions, job descriptions are developed. For non-test specialist functions, existing job descriptions are enhanced with typical test tasks and responsibilities.

Example work products

1. Job descriptions for test functions
2. Enhanced job descriptions for non-test specialists

Subpractices

1. Define job descriptions for each of the identified test functions

Job descriptions typically include:

- Name of the test function
- Short description
- Typical tasks to be performed
- Responsibilities and authorities
- knowledge and skills required
- Certifications required, e.g., ISTQB certifications

2. Enhance the job descriptions for non-test specialists to include their test tasks and responsibilities

Examples of non-test specialist job categories that typically encompass test tasks and responsibilities include:

- Software developer
- System integrator
- User representative
- Product owner

3. Use the organization's standard test process as a major input to define and enhance the job descriptions
4. Review the job descriptions with stakeholders

P2.3 Assign staff members to test functions

Staff members are assigned to the identified test functions.

Example work products

1. Staff members assigned to test functions as their job title

Subpractices

1. Assign staff members to test functions
2. Perform job interviews to fill open test specialist positions
3. Ensure that the test specialist positions (functions) are kept occupied
4. Periodically evaluate staff member performance against defined test function requirements
5. Take action based on the evaluation, if necessary

G3 Establish Test Career Paths

Test career paths are established, clarifying pathways for growth to allow testers to advance their careers in testing and improve their knowledge, skills, status and rewards.

P3.1 Define test career paths

Test career paths are defined that will allow testers to advance their careers.

Example work products

1. Test career path framework

Subpractices

1. Differentiate within test functions by defining multiple, relatively small, test career path steps
2. Link required knowledge and skills, typical tasks and responsibilities, training modules and experience level to the test career path steps for each of the differentiated functions
3. Position the defined test functions, and possibly the test career path steps within a test function, in a test career path framework that typically allows both vertical and horizontal growth
4. Link the test career path framework to other career path frameworks available in the organization, e.g., grow towards roles such as project management, product owner or SCRUM master

P3.2 Develop personal test career development plans

A personal test career development plan is developed and maintained for every staff member assigned to a test function.

Example work products

1. Personal career development plans

Subpractices

1. Develop personal development plans based on the test career path framework
2. Periodically review and update the personal development plan in collaboration with the staff member
3. Identify and document actions that are needed to advance the staff member's career development
4. Monitor the defined test career development actions to closure

G4 Determine, Plan and Implement Test Process Improvements

Strengths, weaknesses, and improvement opportunities for the organization's test process are identified periodically and as needed. Process changes are planned, implemented and deployed to ensure the test process and its improvements contribute to successfully meeting business objectives.

As a complementary and partly alternative set of practices, at (Agile or DevOps) team level, retrospectives can be conducted to assess the team's test process, and subsequently identify, plan and implement test process improvements.

P4.1 Assess the organization's test process

The organization's test process is assessed periodically to understand and monitor its strengths and weaknesses.

Example work products

1. Test process assessment report

Subpractices

1. Understand the organization's test process needs using the organizational test and test performance improvement directives, test policy and test strategy
2. Obtain sponsorship for the test process assessment from senior management
3. Define the scope of the test process assessment
4. Plan, schedule and prepare for the test process assessment
5. Conduct the test process assessment
6. Document the test assessment results
7. Present the test assessment results to stakeholders

P4.2 Analyze lessons learned using the organizational test process

Lessons learned from planning and performing the test process are analyzed to identify testing best practices and improvement opportunities for the organizational test process and test process assets.

Example work products

1. Test process improvement proposals
2. Candidate Best practices

Subpractices

1. Obtain feedback about the use of the organization's standard test process and test process assets

2. Analyze lessons learned from practical experiences applying the organization's standard test process and test process assets

   Projects will typically document their lessons learned in a test completion report [ISO 29119-3]. With Agile and DevOps teams, outputs from retrospective meetings capture lessons learned and improvement actions.

3. Identify possible best practices to be made available to other projects and teams and to be stored in the organization's test process asset library for sharing

   Refer to the practice P3.4 "Collect and Share Test Best Practices" from process area 3.3 Test Process and Integration for more information on test best practices.

4. Identify and document opportunities to improve the standard test process and test process assets

P4.3 Identify the organization's test process improvements

Improvement opportunities for the organization's test process and test process assets are identified based on the test assessment and lessons learned in practice.

Example work products

1. Prioritized list of test process improvements

Subpractices

1. Determine candidate test process improvements based on recommendations from the test assessment report
2. Determine candidate test process improvements based on lessons learned identified during a test completion phase or in a team retrospective

   In an Agile and DevOps context, create a mechanism whereby local test improvements are shared organization-wide, and escalated if beyond the team's control.

3. Prioritize the candidate test process improvements

   Examples of factors typically used in determining the priority of the candidate test process improvements include:
   - Synchronization with business and test goals
   - Alignment with the staged priorities of the TMMi model
   - Addressing the most visible process improvements first to create awareness and acceptance
   - Provision of clear business benefits
   - Cost and effort involved
   - Level of difficulty
   - Degree of acceptance
   - Mitigation of risks

4. Discuss and review the prioritized list of test process improvements with stakeholders
5. Identify the test process improvements that will be implemented

P4.4 Plan test process improvements

Actions that are needed to establish the improvements to the organization's test process and test process assets are planned.

Example work products

1. Test process improvement plan

Subpractices

1. Group related test process improvements that can jointly be implemented and deployed
2. Define the approach to establish the test process improvements

Examples of consideration when defining the approach include:

- Does the improvement need to be piloted during implementation before it is fully deployed?
- Will the improvement be deployed in all projects or teams, or only a subset of the organization's projects or teams?
- Is the new or updated process mandatory for all projects and teams, or do the Agile teams themselves decide whether or not the test improvement will become part of their way-of-working?

3. Identify and define actions needed to implement and deploy the test process improvements

Examples of implementation and deployment actions include:

- Update test process assets
- Training of affected staff
- Coaching of projects and teams

4. Establish process action teams to implement and deploy the test process improvements
5. Identify project risks to the implementation and deployment and define appropriate mitigation actions
6. Determine marketing and change management activities to support deployment

Examples of change management activities include:

- Communications to stakeholders
- Kick-off with all parties involved
- Discussion sessions
- Publications, e.g., for information purposes and on successes achieved

7. Document the test process improvement plan

Examples of elements of a test process improvement plan include:

- Test process improvement objectives
- Test process improvement organization structure
- Responsibilities and authorities
- Test process improvements to be implemented
- Procedures for monitoring and control
- Approach and actions for implementing and piloting test process improvements
- Resources and schedule
- Risks associated with the test process improvement plan

8. Review the test process improvement plan with affected stakeholders, e.g., management and members of process action teams
9. Revise the test process improvement plan as necessary

P4.5 Implement test process improvements

The set of test process improvements addressed by the test improvement plan is implemented.

Example work products

1. Results (solutions) of implementing test process improvements, e.g., newly developed or updated process assets including training material.
2. Implementation status reports

Subpractices

1. Create a solution to address the test process improvement, e.g., processes, templates, tools, knowledge and skills (training) and support
2. Pilot the solution as needed in one or more projects or teams
3. Evaluate the results of the pilot against improvement objectives and plan with stakeholders
4. Refine the solution based on the results and evaluation of the pilot
5. Incorporate the test process improvements into the organization's standard test process, as appropriate
6. Monitor and manage progress against the test process improvement plan
7. Provide status reports on the implementation of the test process improvements to stakeholders

P4.6 Manage the deployment

Manage the deployment of the test process improvements.

Example work products

1. Deployment status reports

Subpractices

1. Coordinate the deployment of test process improvements across the organization and within projects and teams
2. Conduct training on the test process improvements to projects and teams and other affected stakeholders
3. Provide coaching to projects and teams to support the deployment of the test process improvements
4. Perform marketing, both inside and outside testing, regarding achievements to keep staff motivated and involved
5. Monitor and manage deployment progress of the test process improvements and new testing technologies
6. Confirm that the deployment of the test process improvements is completed
7. Document and review the results, including lessons learned, of the test process improvements deployment with affected stakeholders

Refer to goal G3 "Deploy the Organization Test Process and Test Process Assets" of the process area 3.3 Test Process and Integration for more information on deployment.

1

Process Area 3.2 Test Training Program (TTP)

Purpose

The purpose of the Test Training Program process area is to develop a training program which facilitates the development of knowledge and skills of people so that test tasks and roles can be performed effectively and efficiently.

Value

Test Training Program aims to enhance individuals' testing skills and knowledge to improve test work performance.

Introductory Notes

Test Training Program covers the training to support the organization's business objectives and to meet the test training needs that are common across projects. Specific training needs identified by individual projects are handled at project level as part of test planning. The training program ensures that those involved in testing, both testers and other team members, have the necessary testing knowledge and skills to perform their test tasks, acquire the necessary domain knowledge and other knowledge and skills required for their role. The training program could be organized and managed by a dedicated training group or by the test organization itself.

Establishing a test training program is an additional commitment by management to support staff to have the necessary test knowledge and skills and to promote continuous test process improvement. The training program is periodically reviewed to ensure alignment with evolving business needs and technology trends. In testing, a variety of knowledge and skills are needed for testing, e.g., test engineering, test management, IT, interpersonal skills and domain expertise. A test training program, consisting of several training modules, is developed to address these categories. Note that at higher TMMi levels more advanced training categories will become important, e.g., defect prevention at TMMi level 5. Some skills may be effectively and efficiently imparted through informal methods, such as on-the-job training or mentoring, whereas others require more formal training. The term "training" is used throughout this process area to include all of these learning options. In an Agile environment, process documentation is often minimal and may not cover every possible scenario. As a result, training of staff becomes crucial, requiring dedicated attention and serving as a vital counterbalance to the lightweight processes.

The test training program is linked to test functions and test roles, and facilitates test career paths. The implementation of the Test Training Program process area involves identifying the organizational test training needs, developing or acquiring specific training modules, conducting training to address the identified needs and evaluating the effectiveness of the training program.

Scope

The process area Test Training Program addresses the establishment of an organizational test training plan and test training capability. It also addresses the delivery of the planned test training. Project-specific test training needs are not part of this process area, but are addressed in the process area Test Planning.

Goal and Practice Summary

G1 Establish a Test Training Capability and Organizational Test Training Plan

- P1.1 Identify test training needs
- P1.2 Establish a test training capability

P1.3 Establish an organizational test training plan

G2 Provide Test Training

P 2.1 Deliver test training

P 2.2 Establish test training records

P 2.3 Assess test training effectiveness

Practices by Goal

G1 Establish Test Training Capability and Organizational Test Training Plan

Test training capability and an organizational test training plan are established and maintained to ensure that personnel are trained to perform their test tasks and roles effectively and efficiently.

P1.1 Identify test training needs

The strategic test training needs of the organization and short-term training needs are identified and maintained. The organization is responsible for addressing common cross-project/team test training needs.

Example work products

1. Organizational test training needs

Subpractices

1. Analyze knowledge and skills required to perform the test roles (per job descriptions) to identify organizational test training needs
2. Map the test training needs to the test functions (including test career paths) and test roles of the organization

   Especially in an Agile and DevOps context it is of utmost importance to also specify the test training needs for team members that perform a test role, other than testers.
3. Analyze the test training needs identified by various projects

   Analysis of specific project/team needs is intended to identify common test training needs that can be most efficiently addressed organization-wide. This analysis activity can also be used to anticipate future test training needs that are first visible at the project/team level.

   Refer to the practice P4.2 “Plan for test staffing” from process area 2.2 Test Planning for more information on project/team specific plans for test training.
4. Determine and prioritize the organizational test training needs

Examples of categories of test training needs include:

- Test engineering and process (e.g., organizational standard test process, test principles, test lifecycle, static test techniques, dynamic test techniques, test tools and test automation)
- Test management (e.g., product risk analyses, test estimation, monitoring and reporting)
- IT related training (e.g., requirements engineering, configuration management, software development, development lifecycle models)
- Interpersonal skills (e.g., communication, team building)
- Domain expertise

5. Review the organizational test training needs periodically and on an event driven basis, and update the training needs if required, e.g., keeping current with new and emerging technologies

P1.2 Establish a test training capability

A test training capability is established and maintained to address the organizational training needs.

Example work products

1. Test training materials and supporting artifacts

Subpractices

1. Select the appropriate approaches to satisfy specific test training needs

Examples of training approaches include:

- Classroom training
- E-learning
- Guided self-study
- Formal apprenticeship and mentoring programs
- Structured on-the-job training

2. Determine whether to develop test training materials internally or acquire them externally

Examples of criteria that can be used to determine the most effective mode of knowledge or skill acquisition include:

- Time available to prepare the training materials
- Availability of in-house expertise
- Availability of training (materials) from external sources
- Available budget
- Time required for maintenance of training material

3. Develop or obtain test training materials
4. Develop or obtain qualified instructors
5. Describe the training in the organization's test training curriculum

Examples of topics to be part of a test training description include:

- Training learning objectives
- Topics covered in the training
- Intended audience
- Prerequisites, e.g., other training courses and practical experience
- Preparation for participating
- Duration of the training
- Lesson plans
- Completion criteria for the course

6. Revise the test training materials and supporting artifacts as necessary

Examples of when to revise the test training materials include:

- Test training needs change (e.g., when new technology associated with the training topic is available)
- When evaluation of the test training identifies the need for change (e.g., evaluations of training effectiveness surveys, training program performance assessments, or instructor evaluation forms)

P1.3 Establish an organizational test training plan

An organizational test training plan is established and maintained.

Note, in many organizations this planning is performed annually with a review each quarter. When selecting those to be trained, also consider the need for managers to understand basic testing principles and the test strategy, developers to be able to perform their role in testing, users to be able to participate in acceptance testing, etc.

Example work products

1. Test training plan

Subpractices

1. Establish a test training plan

Examples of elements of an organizational test training plan include:

- Test training topics
- Schedules based on test training activities and their dependencies
- Approach used for training
- Training tasks, roles and responsibilities
- Required resources including tools, facilities, environments, and staffing
- Required skills and knowledge of the trainers
- Data to be collected for measuring training effectiveness

2. Review the test training plan with affected groups and individuals, e.g., human resources, test organization representatives and (project) management
3. Establish commitment to the test training plan from management, projects and teams
4. Revise the test training plan and commitments as needed

G2 Provide Test Training

Testers, and other individuals involved in testing, receive the training necessary to perform their roles effectively, increasing the likelihood of achieving test objectives.

P2.1 Deliver test training

Training is delivered according to the organizational test training plan.

When selecting those to be trained, consider the need for managers to understand basic testing principles and the test strategy, developers to be able to perform their role in testing, users to be able to participate in acceptance testing, etc.

Example work products

1. Delivered training course
2. Completed course evaluation forms

Subpractices

1. Select those who will receive the training necessary to perform their test role effectively

   Note a waiver may be provided for those already possessing the knowledge and skills required to perform well in their designated roles.

2. Schedule the training and organize required resources
3. Conduct the training
4. Gather course evaluation forms completed by participants

P2.2 Establish test training records

Records are created and maintained that document the test training delivered and conducted.

Example work products

1. Test training records

Subpractices

1. Keep records for those who participated in a course or other training activity
2. Keep records for the results of any examinations after attending a training course
3. Keep records of those who have been waived from specific training, including the rationale
4. Keep records of the training courses delivered including course evaluations and feedback
5. Make training records available to appropriate people for consideration in assignments, e.g., by providing a skill matrix with a summary of experience and education of employees

P2.3 Assess test training effectiveness

The effectiveness of the organization's test training program is assessed. The results of the assessments of test training effectiveness should be used to revise the organizational test training program.

Example work products

1. Training course performance assessments
2. Test training program effectiveness surveys

Subpractices

1. Assess the effectiveness of a training course

Examples of how to assess the effectiveness of a training course include:

- Course evaluations and feedback
- Examination results
- Performing post-training assessment to determine if the training delivered increased the capability of the participant to perform their test tasks and role
- Retrospective meetings

2. Assess the effectiveness of the test training program, e.g., by performing surveys of the test training program effectiveness

Process Area 3.3 Test Process and Integration (TPI)

Purpose

The purpose of Test Process and Integration is to establish, maintain and deploy a set of organizational test processes and test process assets necessary to perform the work, and to integrate and synchronize the test processes with the development lifecycle. The integrated lifecycle also aims to ensure an early involvement of testing and early feedback.

Value

Test Process and Integration aims to provide a capability to understand and consistently repeat successful and efficient test performance, and to support early feedback by testing.

Introductory Notes

A key responsibility of the test organization is to define, maintain and deploy a standard test process and set of test process assets, in line with the organization's test policy and goals. At TMMi level 3 the test process becomes a defined process. A fully defined process is sufficiently detailed that it can be performed consistently by trained and skilled people, making it both persistent and habitual. The organizational test processes and test process assets enable consistent test performance across the organization. The process area Test Process and Integration builds strongly on the practices of the TMMi level 2 process area Implementation and Habit.

The organization's test process asset library is a collection of items maintained for use by the projects and teams of the organization. The collection of items includes descriptions of test processes, descriptions of test lifecycle models (including supporting templates and guidelines for the test-related work products), supporting tools and process tailoring guidelines. Using these guidelines, the organization's set of standard test processes can be tailored by projects and teams to create their own specific processes. The organization's test process asset library also supports learning and process improvement by sharing test best practices across the organization. The implementation of the organizational test processes and test process assets is managed as a project, guided by a deployment plan.

Note, there is a difference between deployment in a traditional versus an Agile or DevOps organization. In a traditional organization the deployment most often results in processes and practices that shall be applied by all those involved. In Agile and DevOps organizations, teams have more autonomy and are self-managed. It is important in these organizations to distinguish between processes and practices that are mandatory for all teams, and processes and practices that are optional. Typically, this distinction will be addressed as part of the tailoring criteria and guidelines. With those elements that are optional, the Agile/DevOps teams can decide for themselves whether or not they will become part of their way-of-working.

The standard test processes are aligned with the development lifecycle models to integrate the testing activities in terms of phasing, milestones, deliverables, and activities. Lifecycle integration is performed in such a way that early involvement of testing in projects and teams is ensured, e.g., during planning and requirements specification. In a DevOps context, lifecycle integration is typically supported and achieved through toolchain integration. Early testing (shift-left) also integrates testing from the start to increase code coverage. Teams test early and run unit, integration, and smoke tests as part of continuous integration to ensure comprehensive coverage.

Scope

The process area Test Process and Integration encompasses practices for developing, maintaining, and deploying a usable set of organizational test processes and assets. It also ensures these activities are integrated

and synchronized with the development lifecycle, enabling early involvement of testing. Initially, it focuses on creating organizational test processes and assets for TMMi levels 2 and 3, but will need to be revisited when more process areas come into scope, e.g., from higher TMMi maturity levels. The process area Test Process and Integration also covers identifying test best practices and sharing them across the organization.

Goal and Practice Summary

G1 Develop the Organizational Test Process and Test Process Assets
- P1.1 Define a test process architecture
- P1.2 Develop standard test processes and test process assets
- P1.3 Develop and use tailoring criteria and guidelines
- P1.4 Develop test process asset library

G2 Integrate the Organizational Test process with the Development Lifecycle Model
- P2.1 Develop an integrated lifecycle model
- P2.2 Review integrated lifecycle model and obtain commitments

G3 Deploy the Organizational Test Process and Test Process Assets
- P3.1 Deploy the standard test process and test process assets
- P3.2 Monitor deployment
- P3.3 Evaluate process adherence
- P3.4 Collect and share test best practices

G1 Develop the Organizational Test Process and Test Process Assets

A set of organizational test processes and test process assets is developed and maintained, supporting the performance of all test levels and test types within scope, to improve consistency and thereby the likelihood of achieving objectives.

P1.1 Define a test process architecture

Determine what process assets are needed to perform the work and define a test process architecture that describes the structure of the organizational test processes and test process assets.

Example work products

1. List of test processes
2. List of test process assets (work instructions, templates and tools)
3. Test process architecture

Subpractices

1. Decompose the standard test process into constituent process elements to the level of detail needed to understand and describe the process

   A standard test process typically consists of groups of activities organized as test planning, test monitoring and control, test analysis, test design, test implementation, test execution and test completion.

2. Identify which test process assets are needed to perform the work, e.g., work instructions, templates

   Templates may also serve as work instructions, by incorporating the instructional content directly within the templates.

3. Identify test process architecture objectives describing how, why and by whom the process architecture will be used

4. Develop a test process architecture showing the structure for the various test processes and test process assets with their dependencies and interfaces
5. Review the test process architecture with stakeholders
6. Revise the list of test processes, test process assets and the test process architecture as necessary

P1.2 Develop standard test processes and test process assets

The organization's set of standard test processes and test process assets is developed and maintained to perform the work.

Multiple standard test processes may be needed to address the needs of different application domains, test levels, lifecycle models, methodologies and tooling. The organization's test process framework should comprehensively encompass all processes required across organizational, project, and team levels.

Example work products

1. Organization's set of standard test process descriptions
2. Work instructions
3. Templates and tools

Subpractices

1. Specify the attributes and format, e.g., flow charts, of each process description

Examples of attributes for process descriptions include:

- Entry criteria and inputs
- Process steps
- Exit criteria and outputs
- Roles and responsibilities
- Applicable standards, work instructions, templates and tools
- Quality gates, e.g., reviews
- Interfaces to other processes

2. Document the organization's set of standard test processes, work instructions, templates and tools

Examples of test work products typically supported by a template include:

- Test plan
- Test specification
- Test charter
- Test log
- Test progress and quality report
- Test completion report

Note, DevOps organizations typically rely heavy on automation, whereby processes are often largely captured in and mandated by tools.

3. Review the organization's set of standard test processes and test process assets to ensure they meet the needs of projects and teams, and satisfy the objectives of the organization
4. Revise the organization's set of standard test processes and test process assets as necessary

P1.3 Develop and use tailoring criteria and guidelines

The tailoring criteria and guidelines for the organization's set of standard test processes and test process assets are developed and maintained.

Example work products

1. Tailoring criteria and procedures for the organization's set of standard test processes

Tailoring criteria and procedures typically include:

- Mandatory minimum requirements that shall be satisfied by the tailored test processes
- Options that may be exercised and criteria for selecting among the options
- Procedures that must be followed in performing and recording test process tailoring

2. Template waiver request

Subpractices

1. Specify the selection criteria and procedures for tailoring the organization's set of standard test processes
2. Specify the procedures for submitting and obtaining approval of waivers from the organization's set of standard test processes
3. Review, approve and communicate the tailoring guidelines for the organization's set of standard test processes
4. Revise the tailoring guidelines as necessary

P1.4 Develop test process asset library

The organization's test process asset library is developed, maintained and made available for use.

Example work products

1. Organization's test process asset library
2. Catalogue of items in the organization's test process asset library

Subpractices

1. Design and implement the organization's test process asset library, including the library structure and support environment
2. Specify the criteria for including items in the library, e.g., criteria for acknowledging a project or team deliverable as a best practice
3. Specify the procedures for storing and retrieving items
4. Enter the selected items into the library and catalogue them for easy reference and retrieval

Examples of items to be stored in the organization's test process asset library include:

- Test policy and test strategy
- Test process descriptions
- Work instructions, e.g., test estimation procedure
- Templates
- Best practices, e.g., example test plan
- Training materials
- Tools, e.g., checklists

5. Make the items available for use in projects and teams
6. Periodically review the usefulness of each item and use the results to maintain the library contents, e.g., consider removing assets that are no longer used or viable

G2 Integrate the Organizational Test Process with the Development Lifecycle Model

The organizational test process is integrated with the development lifecycle model in terms of phasing, milestones, deliverables and activities to ensure alignment, shift-left and early involvement.

P2.1 Develop an integrated lifecycle model

A description of the integrated development and test lifecycle model for use in the organization is developed and maintained. Note, in a DevOps context, lifecycle integration is typically supported and achieved through toolchain integration.

Example work products

1. Description of the integrated development and test lifecycle model

Subpractices

1. Synchronize the test processes with the development lifecycle model
2. Ensure testing is involved early in the development lifecycle, e.g., during requirements engineering and backlog refinement sessions
3. Define interdependencies between development and testing activities, e.g., using entry and exit criteria
4. Define interdependencies between development and testing deliverables and milestones
5. Document the description of the integrated development and test lifecycle model
6. Revise the description of the integrated development and test lifecycle model as necessary

P2.2 Review integrated lifecycle model and obtain commitments

The integrated lifecycle model is reviewed with the stakeholders to promote their understanding and obtain commitments regarding the role of testing within the integrated development and test lifecycle model.

Example work products

1. Integrated lifecycle review log
2. Documented commitments

Subpractices

1. Organize reviews with stakeholders to facilitate their understanding of the role of testing within the integrated development and test lifecycle model
2. Review commitments with senior management as necessary

G3 Deploy the Organizational Test Process and Test Process Assets

The organizational standard test process and test process assets are deployed across the organization to ensure they are applied and the expected benefits are achieved.

The practices within this goal describe ongoing activities. Deployment of the standard test process and other organizational test process assets must be continuously supported within the organization, especially for new projects or teams at startup.

P3.1 Deploy the standard test process and process assets

The standard test process and test process assets are deployed across the organization, especially to projects and teams at their startup, and updates are deployed as appropriate throughout for each project and team.

It is important to involve not only those who are, or will be, performing the test process, but also other organizational functions as necessary, such as training and internal audit.

Example work products

1. Deployment plan
2. Deployment guidelines and other supporting materials for deployment, e.g., training
3. Deployment status reports

Subpractices

1. Select test process improvements to be deployed based on their priority and resource availability
2. Identify projects and teams within the organization that are in their start-up
3. Identify running projects and teams that need to implement the organization's (updates to the) standard test process and test process assets
4. Establish plans to deploy the selected organization's standard test process and test process assets in the identified projects and teams
5. Review the deployment plan with affected stakeholders
6. Document the changes to the organizational standard test process and test process assets to enable communication on their updates
7. Ensure the deployment is announced and communicated effectively
8. Develop and provide appropriate training to those who will perform the standard test process and use the test process assets
9. Pilot the identified test process improvements
10. Provide guidance and coaching on the use of the standard test process, test process assets and test process asset library
11. Assist projects and teams in tailoring the organization's standard test process and test process assets to meet their needs
12. Perform marketing, both inside and outside testing, regarding achievements to keep staff motivated and involved
13. Communicate the results of the deployment to stakeholders

P3.2 Monitor deployment

The deployment of the organization's standard test process and the use of the test process assets in projects and teams is monitored

Example work products

1. Status report and results of test process deployment across projects and teams
2. Updated deployment plan

Subpractices

1. Monitor projects and teams for their use of the organization's test process and test process assets, including any updates to them

   In an Agile and DevOps context, monitoring deployment can be part of the retrospective meetings.

2. Monitor the use of tailoring criteria and guidelines used to justify deviations from the standard test process

3. Review the results of the monitoring activities to determine how well the organization's standard test process and test process assets have been deployed
4. Identify, document, and manage to closure issues related to deploying the organization's standard test process and test process assets
5. Update the deployment plan as necessary

P3.3 Evaluate process adherence

Objectively evaluate adherence of the test processes and work products to organizational process descriptions, work instructions and templates, including their effectiveness, and address non-compliances.

Example work products

1. Process verification results

Subpractices

1. Develop and follow a test process verification approach
2. Evaluate how defined test processes are being followed and test process assets are being used

Examples of techniques that can be used to verify process adherence include the following:

- Objective assessments or audits
- Retrospectives
- Project reviews

3. Evaluate the effectiveness and usefulness of the defined test processes
4. Review selected test process artifacts created by a project or team to determine the level of compliance
5. Record the results of the test process verification activities
6. Communicate the results to stakeholders and ensure resolution issues of non-compliance
7. Derive and submit test process improvement proposals based on the results of the process verifications

P3.4 Collect and share test best practices

High-quality project-based or team-based test assets (both test processes and testware) that are valuable for sharing across the organization, and can possibly be re-used, are collected and deployed.

Example work products

1. Test best practices
2. Updated test process asset library

Subpractices

1. Identify test assets for best practice evaluation

Examples of activities where test assets for best practice evaluation can be identified include:

- Retrospectives / lessons learned sessions
- Test completion reports
- Test process assessments help identify areas of strength. These areas indicate high-quality test process components or testware that can be considered as best practice candidates
- Project reviews
- Project- or team-based test improvement initiatives

2. Document the background and context of each identified test best practice candidates
3. Submit test best practice proposals to the Test Organization
4. Evaluate the candidate test best practice candidates for their compliance with the best practice criteria

Examples of best practice criteria for test assets include:

- The test asset is in line with the organization's set of standard test processes
- The test asset is easy to understand and apply
- The test asset has been successfully applied in a project
- The test asset is not specific and can thus be applied in other projects and teams

5. Decide on test best practices to become part of the best practice library and to be deployed
6. Incorporate the selected test best practices into the organizational test process asset library

   To support best practice sharing, each test best practice is accompanied by a description that contains information supporting projects or teams in applying or tailoring the best practice.
7. Deploy the test best practices to projects and teams across the organization

Examples of deployment mechanisms for test best practices include:

- Presenting them in a test competence meeting
- Distributing them through an informational email
- Featuring in a periodic newsletter
- Ensuring best practices are included in introduction training for the test process asset library

8. Provide coaching, as appropriate, to support deployment of the new test best practices
9. Perform marketing, within and beyond the testing function, on successes achieved by the test best practices process to keep staff motivated and involved
10. Document and review the results of the test best practices process

Process Area 3.4 Non-Functional Testing (NFT)

Purpose

The purpose of the Non-Functional Testing process area is to improve test process capability by including non-functional testing in test planning, test design and test execution. This is achieved by defining a test approach for non-functional testing to mitigate the identified non-functional product risks, and developing and executing non-functional tests.

Value

Non-Functional Testing aims to ensure adequate verification and validation of identified non-functional quality characteristics of the software product.

Introductory Notes

Product quality is centered on satisfying stakeholders' needs. These needs have to be translated into functional ("what" the product does) and non-functional ("how" the product does it) requirements, e.g., user stories. The International Organization of Standardization has defined a set of non-functional characteristics that are used to define the quality of a software product or system [ISO 25010]. This process area addresses the capability for non-functional testing. The process area Non-Functional Testing builds strongly on the practices of the TMMi level 2 process areas Test Planning and Test Design and Execution. At lower TMMi levels, some non-functional testing may already be performed; however, it is only at TMMi Level 3 that it becomes a focus area for test improvement. In line with the objectives of TMMi level 3, non-functional testing should be fully integrated into the development lifecycle, also ensuring an early start (shift-left) of non-functional testing. This is already one of the fundamental changes for the better with Agile, in that quality characteristics are already tested early throughout iterations.

A test approach for non-functional testing is defined based on the result of a non-functional product risk assessment. Depending on the level and type of non-functional risks, it is decided which non-functional characteristics of the product will be tested, to what degree and how. Non-functional product risks and the test approach are defined collaboratively by test specialists and stakeholders. Non-functional test techniques are applied to evaluate the non-functional quality characteristics. These test techniques are used to derive and select non-functional test conditions and to create test cases based on non-functional requirements and design specifications. Test scripts are written to facilitate automation of non-functional testing, and test charters are prepared to support non-functional exploratory testing. Specific test data required to execute non-functional tests is created. Non-functional tests are executed, and when incidents are found, incident reports are created.

Scope

The process area Non-Functional Testing involves performing a non-functional product risk assessment and defining a test approach based on the identified set of risks. It also addresses the test design and test implementation activities required to derive and select non-functional test conditions and test cases. It covers the identification and creation of specific test data to support non-functional testing. Test execution, test logging and reporting of test incidents for non-functional testing are also part of this process area.

Test environment practices, which are often critical for non-functional testing, are not addressed by this process area. These are covered in the TMMi level 2 process area Test Environment and should now also support non-functional testing.

Goal and Practice Summary

G1 Perform a Non-functional Product Risk Assessment
- P1.1 Identify non-functional product risks
- P1.2 Analyze non-functional product risks

G2 Establish a Non-functional Test Approach
- P2.1 Identify non-functional features to be tested
- P2.2 Define the non-functional test approach
- P2.3 Define non-functional exit criteria

G3 Perform Non-functional Test Analysis and Design
- P3.1 Identify and prioritize non-functional test conditions
- P3.2 Identify and prioritize non-functional test cases
- P3.3 Identify specific test data
- P3.4 Maintain horizontal traceability with non-functional requirements

G4 Perform Non-functional Test Implementation
- P4.1 Implement non-functional tests
- P4.2 Create specific test data

G5 Perform Non-functional Test Execution
- P5.1 Execute non-functional tests
- P5.2 Report non-functional test incidents
- P5.3 Establish test log

Practices by Goal

G1 Perform a Non-functional Product Risk Assessment

A product risk assessment is performed to identify the critical areas for non-functional testing.

P1.1 Identify non-functional product risks

Non-functional product risks are identified and documented.

Example work products

1. Identified non-functional product risks

Subpractices

1. Identify and select stakeholders, potentially including specialists in non-functional quality domains, who need to contribute to the product risk assessment process
2. Identify non-functional product risks using requirements documents, e.g., non-functional (regulatory) requirements or non-functional acceptance criteria, and input from stakeholders

   With the importance of security for many organizations, threat modelling can be used to identify security requirements, pinpoint security threats and potential vulnerabilities.
3. Document the type of non-functional risk, background and potential consequences of the risks
4. Identify the relevant stakeholders for each non-functional risk

Note, in practice, the identification of non-functional products risks will typically be combined with P1.2 “Identify product risks” of the process area 2.2 Test Planning.

P1.2 Analyze non-functional product risks

Non-functional product risks are evaluated, categorized and prioritized.

Example work products

1. Non-functional product risk list, with a category and priority assigned to each risk

Subpractices

1. Analyze the identified non-functional products risks for likelihood and impact
2. Categorize and group non-functional product risks

Non-functional product risks are typically categorized by quality characteristic. Examples of non-functional quality characteristics include [after ISO 25010]:

- Compatibility
- Flexibility
- Maintainability
- Performance efficiency
- Reliability
- Safety
- Security
- Usability

3. Prioritize the non-functional product risks for mitigation
4. Establish traceability between non-functional product risks and requirements to ensure that the source of product risks is documented
5. Review and obtain agreement with stakeholders on the completeness, category and priority level of the non-functional product risks
6. Revise the non-functional product risks as necessary

G2 Establish a Non-functional Test Approach

A test approach for non-functional testing, based on identified non-functional product risks, is established and agreed upon to provide all stakeholders with a clear understanding of the approach, their specific tasks and objectives.

P2.1 Identify non-functional features to be tested

The non-functional features (also called non-functional quality characteristics) to be tested or excluded are identified based on the non-functional product risks.

Example work products

1. List of non-functional features (quality characteristics) to be tested and not to be tested

Subpractices

1. Breakdown the prioritized non-functional product risks into non-functional features to be tested and not to be tested
2. Document the risk level and source documentation (test basis) for each identified feature to be tested

P2.2 Define the non-functional test approach

The test approach is defined to mitigate the identified and prioritized non-functional product risks.

Example work products

1. Non-functional test approach (documented in a test plan)

The approach should be described in sufficient detail to be able to identify major test tasks and estimate their effort.

Subpractices

1. Select the non-functional test techniques to be used

Examples of non-functional test techniques include:

- For maintainability: static code analysis
- For performance efficiency: load, stress and volume testing
- For reliability: operational profiles, static code analysis
- For security: vulnerability scanning, penetration testing and ethical hacking
- For usability: heuristic evaluation, user survey and questionnaire

Note, black box techniques, white box techniques and experienced-based techniques such as exploratory testing and checklists can also be selected and used to test specific non-functional quality characteristics.

2. Define the approach to automation and identify supporting test tools to be used
3. Define the approach for non-functional regression testing
4. Identify significant constraints regarding the non-functional test approach, such as test resource availability, test environment features and deadlines
5. Align the non-functional test approach with the defined organization-wide or program-wide test strategy
6. Identify any areas of non-compliance with the test strategy and their rationale
7. Review the non-functional test approach with the stakeholders
8. Revise the non-functional test approach as necessary

P2.3 Define non-functional exit criteria

The exit criteria (also referred to as definition-of-done) for non-functional testing are defined to determine when testing is complete.

Example work products

1. Non-functional exit criteria (definition-of-done)

Subpractices

1. Define a set of exit criteria related to the non-functional quality characteristics to be tested

Examples of exit criteria related to non-functional product quality characteristics include:

- For maintainability: average effort to change, availability of documentation
- For performance: mean response time, memory utilization
- For reliability: Mean Time Between Failures (MTBF), Mean Time to Repair (MTTR)
- For usability: user satisfaction, average time to perform functions

2. Review the non-functional exit criteria with the stakeholders

Note, a practice for defining entry criteria is not defined for this process area. The entry criteria that were defined as part of the process area Test Planning are typically also applicable to non-functional testing. However, briefly revisiting the defined entry criteria defined at Test Planning from a non-functional testing perspective is expected.

G3 Perform Non-functional Test Analysis and Design

During test analysis and design the test approach for non-functional testing is translated into test conditions and test cases to ensure adequate coverage of the identified non-functional product risks.

P3.1 Identify and prioritize non-functional test conditions

Test conditions are identified and prioritized, based on an analysis of the non-functional features as specified in the test basis.

Example work products

1. Test basis issue log
2. Non-functional test conditions
3. Non-functional test design specification

Subpractices

1. Study and analyze the test basis (such as non-functional requirements, user stories, design and interface specifications), e.g., during backlog refinement sessions
2. Discuss issues regarding the test basis with the document owner
3. Derive the test conditions from the test basis in line with the documented non-functional test approach
4. Prioritize the test conditions based on identified non-functional product risks
5. Document the test conditions in a test design specification or, in an Agile context, as acceptance criteria
6. Review the test conditions with stakeholders
7. Revise the test conditions as necessary, e.g., when the requirements change

P3.2 Identify and prioritize non-functional test cases

Non-functional test cases are identified and prioritized to address the defined test conditions.

Example work products

1. Non-functional test cases

Subpractices

1. Derive the test cases from the test conditions or acceptance criteria to ensure their coverage
2. Prioritize the test cases based on identified non-functional product risks
3. Document the non-functional test cases as appropriate, e.g., by means of a test case specification
4. Review the test cases with stakeholders
5. Revise the test cases as necessary

P3.3 Identify specific test data

Specific test data necessary to support the non-functional test cases is identified.

Example work products

1. Test data specification

Subpractices

1. Identify the specific test data required to implement and execute the non-functional test cases
2. Document the specific test data

P3.4 Maintain horizontal traceability with non-functional requirements

Traceability between the non-functional requirements. e.g., user stories, and the non-functional test conditions / test cases is established and maintained.

Example work products

1. Non-functional requirements vs. test conditions / test cases traceability

Subpractices

1. Maintain non-functional requirements traceability to ensure that the source of non-functional test conditions / test cases is documented
2. Establish non-functional requirements vs. test conditions / test cases traceability
3. Set up the traceability such that monitoring non-functional requirements coverage is facilitated

G4 Perform Non-functional Test Implementation

Non-functional test procedures, test charters and/or automated scripts are developed to allow for an effective and efficient execution of the non-functional tests. Specific test data to support non-functional testing is created.

P4.1 Implement non-functional tests

Non-functional test conditions and test cases are implemented for execution in the format of test procedures, test charters and/or automated test scripts, in line with the defined test approach.

Example work products

1. Non-functional test procedure specifications
2. Non-functional test charters
3. Non-functional automated test scripts

Subpractices

1. Develop non-functional test procedures based on the identified non-functional test cases, ordering them for execution and including information needed for test execution
2. Develop non-functional test charters based on the identified non-functional test conditions as a basis for performing exploratory testing, e.g., a heuristic evaluation to test the usability
3. Develop non-functional automated test scripts based on the identified non-functional test conditions and test cases
4. Prioritize the non-functional test procedures, test charters and test scripts based on identified product risks
5. Review the non-functional test procedures, test charters and test scripts with stakeholders
6. Revise the non-functional test procedure specifications as necessary

Refer to the practice P2.4 “Develop test execution schedule” from the process area 2.4 Test Design and Execution for scheduling the execution of non-functional tests.

P4.2 Create specific test data

Specific test data is created to support non-functional testing.

Example work products

1. Specific test data

Subpractices

1. Create specific test data (as specified during the test design activity) required to perform the automated and manual non-functional tests
2. Archive the set of specific test data to allow the initial situation to be restored when required

Refer to the practice P3.2 “Perform generic test data management” from the process area 2.5 Test Environment for managing the created test data.

G5 Perform Non-functional Test Execution

Non-functional tests are executed to verify and validate the non-functional quality of the software product. Incidents are reported and test logs are created.

P5.1 Execute non-functional tests

The non-functional tests are executed (either manually, using documented test procedures and/or exploratory testing, or via automation using pre-defined test scripts) according to the defined test execution schedule. Typically, in a DevOps context, non-functional test execution is an integral part of the CI/CD pipeline.

Example work products

1. Test results

Subpractices

1. Execute non-functional tests using documented test procedures, test charters and/or automated test scripts
2. Record actual results
3. Compare actual results with expected results
4. After the receipt of a fix, perform confirmation testing by repeating non-functional test activities
5. Perform non-functional regression testing as necessary

Note, non-functional test execution is normally preceded by a smoke test. Refer to the practice P3.1 “Perform smoke test” from the process area 2.4 Test Design and Execution for more details on smoke testing a test object, and to the practice P2.4 “Perform test environment smoke test” from the process area 2.5 Test Environment for more details on the smoke testing a test environment.

P5.2 Report non-functional test incidents

Differences between actual and expected results are reported as non-functional test incidents.

Example work products

1. Non-functional test incident reports

Subpractices

1. Log a test incident when a discrepancy is observed
2. Analyze the non-functional test incident for further information on the issue
3. Determine the suspected cause of the non-functional test incident
4. Assign an initial priority and severity level to the test incident
5. Record the test incident using an incident report template
6. Review the test incident report with stakeholders
7. Store test incidents in a central repository

Refer to the goal G4 “Manage Test Incidents to Closure” from the process area 2.4 Test Design and Execution for more details on how test incidents are processed and managed to closure.

P5.3 Establish test log

Test logs are established (written or generated) to provide a record of relevant details about the execution of the non-functional tests.

Example work products

1. Test logs

Subpractices

1. Collect non-functional test execution data
2. Document the non-functional test execution data by means of a test log, based on a test log standard
3. Review the test log with stakeholders

Process Area 3.5 Peer Reviews (PR)

Purpose

The purpose of the Peer Reviews process area is to identify and resolve defects from selected work products early and efficiently.

Value

Peer Reviews aim to reduce cost and rework by uncovering defects and other issues as early as possible.

Introductory Notes

Reviews are an evaluation of work products by one or more individuals to early identify defects and areas where changes are needed. The work product to be reviewed could be a set of requirements, design documents, source code, test cases, a user manual, or any other type of document. Several types of reviews are defined, each with its own distinct purpose and objective. Reviews vary from very informal to highly formal (e.g., inspections). Peer reviews support the objectives of TMMi level 3 by making static testing (reviews) an integral part of the development lifecycle, ensuring early involvement of testers, early testing and early defect detection.

This TMMi process area covers the practices for defining a peer review approach within a project or team, and subsequently performing the peer reviews accordingly. A review approach defines which work products will be reviewed, who should be involved, when review activities should take place and whether those activities are formal or informal. Static analysis may be run automatically to review code against coding standards and identify vulnerabilities. It is important that testers are part of the review sessions, especially the reviews that address the test basis. Typically, testers provide a unique perspective whereby they will often identify missing details.

Testing covers both verification and validation and includes both static and dynamic testing. In line with this view, peer reviews are an intrinsic part of testing, serving as a verification, validation and static analysis technique. As a result, the peer review approach and dynamic test approach should be coordinated and aligned in the context of mitigating the identified set of product risks.

Establishing a peer review approach is also applicable to Agile and DevOps. However, typically, the review techniques applied and the way reviews are organized differ. With Agile and DevOps, work products are almost continuously reviewed by the team as part of the whole team approach, e.g., by mandating a code review in every pull request. In addition, validation-oriented backlog refinement sessions and product demonstrations are organized with business representatives.

Scope

The Peer Reviews process area covers the practices for preparing and performing peer reviews on work products, e.g., testers reviewing requirements for testability. It also covers the practices for defining a peer review approach, coordinated with the dynamic test approach, at project or team level. Project reviews (also known as management reviews) are not in scope of this process area.

Goal and Practice Summary

G1 Establish a Peer Review Approach

- P1.1 Develop peer review procedures
- P1.2 Identify work products to be reviewed

G2 Perform Peer Reviews

P2.1 Conduct peer reviews
P2.2 Testers review test basis documents
P2.3 Resolve issues identified in peer reviews

Practices by Goal

G1 Establish a Peer Review Approach

A peer review approach, coordinated with the dynamic test approach, is established and agreed upon to manage cost by targeting critical work products for peer review.

P1.1 Develop peer review procedures

Develop and maintain procedures and supporting materials used to prepare for and perform peer reviews.

Example work products

1. Peer review procedures
2. Peer review supporting materials

Subpractices

1. Develop and maintain procedures for performing peer reviews

Examples of elements of peer review procedures include:
- Criteria for selecting work products
- Deciding peer review type
- Work product evaluation criteria, e.g., INVEST for user stories [Wake]
- Peer review steps (planning, kick-off, preparation, meeting, rework and follow-up) [Gilb and Graham]
- Entry and exit criteria

2. Establish and maintain supporting materials for peer reviews

Examples of supporting materials for peer reviews:
- Peer review forms, e.g., for logging
- Checklist with common issues or defect types

3. Review the defined peer review procedures and supporting materials with stakeholders

P1.2 Identify work products to be reviewed

The (test) work products to be reviewed are identified, including the type of review and key stakeholders to be involved. The defined peer review approach is coordinated and aligned with the dynamic test approach.

Example work products

1. List of (test) work products to be reviewed
2. Peer review approach
3. Updated dynamic test approach

Subpractices

1. Select (test) work products that will undergo a peer review based on the criticality of the work product and their relationship to identified product risks
2. Determine what type of peer review will be conducted for the selected work products

Examples of peer review types include:

- Formal review, e.g., inspection, walkthrough and technical review [ISO/IEC 20246]
- Informal review, e.g., pair reviews, backlog refinement sessions and product demonstrations

3. Identify key stakeholders, e.g., business analysts/product owners, who should be involved in a peer review
4. Coordinate and align the peer review approach with the dynamic test approach in the context of mitigating the identified product risks
5. Revisit the dynamic test approach to consider updating it as a result of product risk coverage to be attained by peer reviews
6. Review the peer review approach with stakeholders

G2 Perform Peer Reviews

Peer reviews are performed on selected work products to improve work product quality and reduce cost and rework by uncovering issues early in the development lifecycle.

P2.1 Conduct peer reviews

Selected (test) work products are peer reviewed and issues are identified.

Example work products

1. Peer review schedule (may include date, time and reviewers)
2. Peer review issue list
3. Peer review action items

Subpractices

1. Develop a detailed peer review schedule
2. Ensure that the peer reviews follow the defined procedures
3. Identify defects and other issues in the work products being reviewed
4. Identify action items based on the peer reviews, e.g., update product risks list or revisit peer review approach
5. Record results of the peer reviews
6. Communicate the issues and action items to affected stakeholders

P2.2 Testers review the test basis documents

The documents used as a basis for testing are reviewed by the testers.

Depending on the peer review approach and involvement of testers, this practice can be an integral part of the previous practice (P2.1 “Conduct peer reviews”).

Example work products

1. Testability defects
2. Testability review report

Subpractices

1. Testers review the test basis for testability, e.g., is it possible to derive test cases and expected results unambiguously?

   During backlog refinement sessions testers can support the identification and definition of acceptance criteria for a certain user story.

2. Defects found during the review of the test basis are logged and reported
3. The test basis is improved based on the defects reported by testers

P2.3 Resolve issues identified in peer reviews

Resolve the issues and address action items, e.g., update product risk list or revisit peer review approach, that are identified in peer reviews.

Example work products

1. Updated peer review issue list
2. Updated peer review action list

Subpractices

1. Resolve defects and other issues identified during the review
2. Verify that all identified issues and action items during the review session are addressed
3. Record resolutions and results, and communicate to affected stakeholders

TMMi Level 4 Measured

Achieving the goals of TMMi level 2 and 3 has the benefits of putting in place a technical, managerial, and staffing infrastructure capable of thorough testing and providing support for test process improvement. Testing has become a defined process applied consistently throughout the organization, which is a prerequisite for a successful and meaningful measurement program. With all of this in place, testing can become a measured process to encourage further growth and accomplishment.

An organization-wide test measurement program will be implemented whereby test measurement objectives, test measures and procedures are defined based on identified information needs and business objectives to optimize resource usage. Information needs typically focus on improved understanding of test performance or results of test improvement actions. Test measurement is the process of identifying, collecting, and analyzing data from the test process activities and the products being developed, to understand the effectiveness and efficiency of the test processes. Test measurement will also provide support to individual projects and teams by offering historical data and metrics, e.g., to support accurate estimation and planning. An organizational test measurement repository is established to enable sharing of historical data across projects and teams.

With respect to product quality, the presence of a measurement program allows an organization to implement a product quality evaluation process at project or team level. By starting with identifying product quality needs, it is ensured that only the necessary product quality characteristics are included in the scope of measurement. (Work) products are evaluated using quantitative criteria for quality characteristics such as functional suitability, reliability, usability and maintainability. Product quality is understood in quantitative terms and is managed to the defined objectives throughout the lifecycle. Peer review as a defect detection technique is transformed, using sampling, into a technique to measure product quality. The overall objective is to contribute to satisfying the needs of the business, users and customers by delivering quality products.

The process areas at TMMi level 4 are:

4.1 Test Measurement

4.2 Product Quality Evaluation

Both of these process areas are described in more detail in the sections hereafter.

Process Area 4.1 Test Measurement

Purpose

The purpose of Test Measurement is to identify, collect, analyze and apply measurements to support an organization in understanding the effectiveness and efficiency of the test process and the results of test improvements.

Value

Test Measurement aims to define measurements to improve the understanding of progress towards achieving business and test objectives.

Introductory Notes

After achieving TMMi level 3, a defined organizational standard test process has been put in place with practices that enable thorough testing and provide support for test process improvement. With this infrastructure in place, a test measurement program can be established to drive further growth and accomplishments in testing. Test measurement is the process of identifying, collecting, and analyzing data on both the test processes and the products being developed to understand the effectiveness and efficiency of the test processes. Measurement and analysis methods and processes are defined to support a successful implementation of the test measurement program.

Test measurement objectives, test measures and procedures are defined based on identified information needs and business objectives to optimize resources usage. Information needs typically focus on improved understanding of test performance or results of test improvement actions. When implemented successfully, the test measurement program will become an integral part of the test culture, and measurement will become a practice adopted and applied by all projects and teams.

Test measurement will also provide support to individual projects and teams by offering historical data and metrics, e.g., to support more accurate estimation and planning. An organizational test measurement repository is established to enable sharing of historical data across projects and teams.

At TMMi level 2 an organization starts to collect data related to the testing process in the context of test performance indicators within the process area Test Policy and Strategy. When moving towards TMMi level 4, an organization will recognize the need for additional measures to achieve higher levels of test process maturity. At TMMi level 4 and above, the test measurement activities are structured and coordinated at organizational level and driven by information needs and business objectives.

Having a focused test measurement program with a foundation of added (business) value ensures that the TMMi Test Measurement process area remains fully consistent with the Agile principle of simplicity. Note that within Agile and DevOps the focus is often more team-based and related to systems thinking. This may result in a corresponding broadening of some of the measures to team level and overall system rather than being limited to only testing-specific aspects. This may well lead to more challenges at the measurement analysis phase.

Scope

The process area Test Measurement addresses measurement activities at the organizational level. Test Measurement covers practices such as defining measurement objectives, specifying operational definitions of test measures, gathering data, analyzing data and reporting the results. In addition, a test measurement repository is established to support projects and teams, e.g., to enable more accurate estimations. It also

encompasses organizational test measurement activities that were defined at lower TMMi levels, such as test performance indicators from Test Policy and Strategy.

Goal and Practice Summary

G1 Define Test Measurement Objectives
- P1.1 Establish test measurement objectives
- P1.2 Specify test measures
- P1.3 Specify data collection and storage procedures
- P1.4 Specify analysis and reporting procedures
- P1.5 Establish a test measurement repository

G2 Provide Test Measurement Results
- P2.1 Obtain test measurement data
- P2.2 Analyze test measurement data
- P2.3 Communicate results
- P2.4 Store data and results

Practices by Goal

G1 Define Test Measurement Objectives

Test measurement objectives, test measures and procedures are defined based on identified information needs and business objectives to optimize resource usage and increase the likelihood of achieving business results.

P1.1 Establish test measurement objectives

Organizational test measurement objectives based on identified information needs and business objectives are established and maintained.

Example work products

1. Test measurement objectives

Subpractices

1. Identify current and planned test process improvements

   Refer to the goal G4 “Determine, Plan and Implement Test Process Improvement” of the process area 3.1 Test Organization for an overview of the current and planned test process improvements.

2. Identify and prioritize information needs using input from stakeholders and other sources
3. Define the organization’s test measurement objectives based on the prioritized information needs

   Objectives typically address improved understanding of test performance or results of test improvement actions. A common test measurement objective is to improve estimation, planning and performance accuracy through the use of historical data.

4. Review the test measurement objectives, e.g., against information needs and business objectives, with stakeholders
5. Revisit the test measurement objectives on a regular basis and update them as necessary

P1.2 Specify test measures

Test measures are specified that address the test measurement objectives.

Example work products

1. Operational definitions of test measures

Subpractices

1. Identify test measures required based on documented test measurement objectives

   Measures can be either base or derived. Data for base measures is obtained by direct measurement. Data for derived measures, e.g., ratios, typically come from combining two or more base measures.

Examples of test measures include:

- Test estimates and actual data, e.g., on size, effort and cost
- Quality measures, e.g., number of defects found by priority level
- Test coverage

2. Specify operational definitions in exact and unambiguous terms for the identified test measures
3. Review the specification of test measures for their appropriateness with stakeholders

P1.3 Specify data collection and storage procedures

Collection methods are clearly defined to ensure appropriate and accurate data acquisition. Storage and retrieval procedures are specified to ensure that data is available and accessible for future use.

Example work products

1. Data collection and storage procedures

Subpractices

1. Identify the measurement data that is required for each test measure
2. Identify sources of the measurement data
3. Specify how to collect and store the measurement data

Examples of topics that need to be included in the collection and storage procedures include:

- Frequency of collection
- Points in the process or automated pipeline where data will be collected
- Responsibilities for collecting the data and data storage (including security)
- Mechanisms for ensuring data quality, e.g., accuracy, completeness, integrity, and validity
- Forms, templates and tools to collect data automatically
- Data retention period

4. Review data collection and storage procedures with stakeholders

P1.4 Specify analysis and reporting procedures

Data analysis procedures are specified in advance to ensure that appropriate analysis will be conducted and reliable test measurement data is reported in line with the defined test measurement objectives.

Example work products

1. Data analysis procedures
2. Data reporting procedures

Subpractices

1. Identify the data analysis that will be conducted and the measurement reports that will be prepared

   The analysis should explicitly address the documented test measurement objectives. Presentation of

the results should be clearly understandable by the stakeholders to whom the results are addressed

2. Select data analysis methods and tools
3. Specify how to analyze the data and communicate the results
4. Review data analysis and reporting procedures with stakeholders

P1.5 Establish a test measurement repository

A test measurement repository is established and maintained at organizational level to support the test measurement process.

Example work products

1. Organization's test measurement repository

Subpractices

1. Determine the organization's needs for storing, retrieving and analyzing test measurements

   This subpractice results in understanding and recording the requirements for the test measurement repository.

2. Design and implement the test measurement repository
3. Communicate the availability of the test measurement repository
4. Revise the test measurement repository as necessary

Examples of when the test measurement repository may need to be revised include:

- An update to the test measurement objectives
- New test processes added
- Test processes are revised are new test measures are needed
- Data with finer granularity is required

G2 Provide Test Measurement Results

Test measurement results that address the identified information needs and test measurement objectives are provided to stakeholders to provide insight into testing and quality performance and be able to identify actions needed to meet objectives.

P2.1 Obtain test measurement data

The test measurement data necessary for analysis are obtained according to their operational definition and checked for quality.

Example work products

1. Base and derived test measurement data

Subpractices

1. Gather data from within the organization, projects or teams for specified base measures
2. Calculate derived test measures
3. Store the test measurement data in the test measurement database
4. Perform data quality checks, e.g., accuracy, completeness, integrity and validity, as close to the source of the data as possible
5. Identify, communicate and track data quality issues to closure

P2.2 Analyze test measurement data

The collected test measurement data is analyzed as planned and additional analysis is conducted as necessary.

Example work products

1. Data analysis results
2. Draft test measurement reports

Subpractices

1. Conduct analysis, interpret results, and draw preliminary conclusions
2. Conduct additional analysis as necessary, and prepare results for presentation

   The results of initial analysis as planned, may suggest (or require) additional, unanticipated analysis.
3. Record analysis results and any significant deviation, and document them in a draft test measurement report
4. Review results with selected stakeholders to avoid misunderstanding and improve data analysis and communication
5. Refine operational definitions of test measures and/or data analysis procedures

P2.3 Communicate results

Results of test measurement analysis are communicated to affected stakeholders.

Example work products

1. Test measurement reports, including visuals (e.g., charts or dashboards)
2. Improvement proposals

Subpractices

1. Communicate test measurement results to stakeholders on a regular basis
2. Assist stakeholders in understanding the results

   Examples of actions to assist stakeholders in understanding of results include:
   - Discussing findings in feedback sessions
   - Providing memos with explanations
   - Offering training on the appropriate use and understanding of test measurement results
3. Define and implement corrective actions, e.g., to correct deviations, based on the test measurement results in collaboration with stakeholders
4. Define and submit improvement proposals based on the test measurement results in collaboration with stakeholders

P2.4 Store data and results

The test measurement data, analysis results and test measurement reports are stored.

Example work products

1. Stored test measurement data
2. Stored analysis results and test measurement reports

Subpractices

1. Store test measurement data according to the data storage procedures
2. Store the analysis results, test measurement reports and presentations

3. Restrict access to test measurement data to appropriate groups and personnel

Process Area 4.2 Product Quality Evaluation

Purpose

The purpose of Product Quality Evaluation is to develop a quantitative understanding of the quality of the products and thereby support the achievement of projects' specific product quality goals.

Value

Product Quality Evaluation aims to measure product quality to improve the understanding of progress towards achieving projects' product quality goals and thereby fulfilling stakeholders' product quality needs.

Introductory Notes

Product Quality Evaluation involves defining the project's quantitative product quality goals based on stakeholders' product quality needs. By starting with identifying product quality needs, we ensure that measurements for only the required product quality characteristics are in scope. The process area involves defining quality measures for evaluating (work) product quality. Subsequently, product quality status is monitored and corrective actions are taken when necessary. The overall objective is to contribute to satisfying the needs of the business, users and customers for quality products.

The practices of Product Quality Evaluation build on the practices of process areas at TMMi maturity levels 2 and 3. Test Design and Execution and Non-Functional Testing establish key test engineering practices at project or team level that are refined and advanced by Product Quality Evaluation. As product quality is articulated through exit criteria (definition-of-done) and/or through specific features, epics, or user stories with quantifiable acceptance criteria, the established methods and techniques outlined in process area 2.3 (Test Monitoring and Control) remain applicable for tracking and managing progress towards the defined quality objectives. Progress will be measured regularly and, in an Agile and DevOps context, discussed in the daily standup meeting as part of reporting on progress, quality and risk.

Note, this process area is a project- or team-focused process area in contrast to Test Measurement, which is an organizational process area.

Scope

The Product Quality Evaluation process area covers the practices at the project or team level for developing a quantitative understanding of the product being developed and achieving defined and measurable product quality goals. Both functional and non-functional quality characteristics [ISO 25010] are in scope when defining the product quality goals. Product Quality Evaluation is typically supported by the Test Measurement process area which provides the measurement infrastructure.

Goal and Practice Summary

G1 Establish Quantitative Project Goals for Product Quality

- P1.1 Identify product quality needs
- P1.2 Define the project's product quality goals
- P1.3 Define the approach for measuring product quality

G2 Manage Progress towards Achieving the Project's Product Quality Goals

- P2.1 Measure product quality
- P2.2 Analyze product quality measures and compare with product quality goals

Practices by Goal

G1 Establish Quantitative Project Goals for Product Quality

A set of quantitative project goals for product quality is established and maintained, based on identified product quality needs, to ensure clear focus on important quality characteristics and to optimize resource usage

P1.1 Identify product quality needs

Project product quality needs are identified and prioritized.

Example work products

1. Identified and prioritized product quality needs

Subpractices

1. Identify and select stakeholders who need to contribute to the identification of the project's product quality needs
2. Review the organization's objectives for product quality

   The intent of this review is to ensure that the project stakeholders understand the broader business context in which the project operates. Project's product quality needs are identified in the context of these overarching organizational product quality objectives.
3. Elicit product quality needs using input from stakeholders and other sources

Examples of ways to elicit product quality needs include:

- Customer feedback
- Questionnaires [Trienekens and Van Veenendaal]
- Interviews
- Focus groups
- User observation
- Telemetry
- Brainstorming

4. Prioritize the identified set of product quality needs
5. Discuss and resolve conflicts among product quality needs (e.g., if one need cannot be achieved without compromising another)
6. Review and obtain agreement with stakeholders on the completeness and priority level of the project's product quality needs
7. Revise the project's product quality needs as necessary

P1.2 Define the project's product quality goals

Quantitative product quality goals are defined for the project based on the identified product quality needs.

Example work products

1. Product quality measures
2. Product quality goals

Subpractices

1. Identify the product quality characteristics that address the project's product quality needs

Examples of product quality characteristics include [after ISO 25010]:

- Compatibility
- Flexibility
- Functional suitability
- Maintainability
- Performance efficiency
- Reliability
- Safety
- Security
- Usability

2. Prioritize the identified set of product quality characteristics based on the priorities of the product quality needs
3. Identify one or more product quality measures for each selected product quality characteristic

   Product quality measures can be either base or derived. The identified set of quality measures should support the shift-left principle, e.g., by using internal measures. Measuring product quality should start as early as possible to allow for early corrective action. However, in a continuous DevOps value stream environment quality characteristics are typically also measured after deployment in production (shift-right).

Examples of product quality measures include [ISO/IEC 25023]:

- For compatibility: co-existence with other products and data formats exchangeability
- Flexibility: operational environment adaptability and installation time efficiency
- For functional suitability: functional coverage and functional correctness
- For maintainability: coupling of components and coding rules conformity
- For performance efficiency: mean response time and mean memory utilization
- For reliability: failure rate and mean down time
- For security: data encryption correctness and user audit trail completeness
- For usability: entry fields defaults and error messages understandability

4. Define the identified product quality measures by means of a short description and its measurement function
5. For each product quality measure, a quantitative goal (target value) is defined. Product quality goals typically act as exit criteria (definition-of-done) for the project.

   Within Agile and DevOps frameworks, measurable product quality goals may be defined through the definition-of-done, or alternatively through features, epics, or user stories - each supplemented by acceptance criteria to ensure measurability.
6. Review the product quality measures and their goals with stakeholders
7. Revise the product quality measures and their goals as necessary

P1.3 Define the approach for measuring product quality

An approach is defined for gathering product quality measurement data and measuring product quality.

Example work products

1. Product quality measurement approach

Subpractices

1. Identify the measurement data that is required for the derived product quality measures

2. Identify sources of the measurement data for the defined product quality measures
3. Specify how to collect and store the product quality measurement data

Examples of ways to collect measurement data include:

- Peer reviews using sampling
- Prototyping
- Static analysis
- Dynamic testing
- Observability using monitoring, logging and traces

4. Specify the analysis that will be conducted on the product quality measurement data
5. Specify how product quality status will be communicated to stakeholders
6. Review and obtain agreement with stakeholders on the product quality measurement approach

Refer to the process area 4.1 Test Measurement for more information on data collection, storage and analysis.

G2 Manage Progress towards Achieving the Project's Product Quality Goals

Product quality is measured and monitored to determine whether the project's product quality goals are being achieved. Product quality status is reported, reviewed and discussed with stakeholders and corrective actions are identified as appropriate.

P2.1 Measure product quality

The quality of the project's work products and final deliverable is measured as defined and checked for compliance with product quality goals.

Example work products

1. Base and derived product quality measurement data

Subpractices

1. Gather product quality data from work products and the product for specified base product quality measures in accordance with the defined measurement approach
2. Calculate derived product quality measures
3. Store product quality measurement data in the project database
4. Perform data quality checks, e.g., accuracy, completeness, integrity and validity, as close as possible to the source of the data
5. Identify, communicate and track data quality issues to closure

P2.2 Analyze product quality measures and compare with product quality goals

The product quality measures are analyzed and compared to the project's product quality goals. Product quality reviews are held regularly, e.g., daily (in an Agile and DevOps context), weekly or bi-weekly.

Example work products

1. Product quality measurement reports, including visuals (e.g., charts or dashboards)
2. List of product quality issues needing corrective actions

Subpractices

1. Conduct analysis of the product quality measures and determine product quality status

2. Compare the product quality measures against the project's product quality goals, and draw preliminary conclusions

   Measures that indicate low product quality (compared with set product quality goals) should be subject to further investigation.

3. Communicate, review and discuss product quality measures and the level of achievement of product quality goals with project or team members, and stakeholders
4. Identify and document significant product quality issues and their impact
5. Determine and document the appropriate actions needed to address the identified product quality issues
6. Manage corrective actions to closure

   Refer to goal G3 "Manage corrective actions to closure" from the process area 2.3 Test Monitoring and Control for more information on managing corrective actions to closure.

TMMi Level 5 Optimization

The achievement of all previous test improvement goals at levels 2 through 4 of TMMi has created an organizational infrastructure for testing that supports a completely defined and measured process. At TMMi maturity level 5, an organization is capable of continuously improving its processes based on a quantitative understanding of statistically controlled processes. Improving test process performance is carried out through incremental and innovative process and technological improvements. The testing methods and techniques are constantly being optimized and there is a continuous focus on fine tuning and process improvement. To support the continuous improvement of the test process infrastructure, and to identify, plan and implement test improvements, a permanent test process improvement group (also known as a Test Process Group) is formally established and is staffed by members specifically trained in the knowledge and skills required. Support for a Test Process Group formally begins at TMMi level 3 when the test organization is introduced. At TMMi level 4 and 5, its responsibilities grow as more high-level practices are introduced.

At TMMi level 5, testing completes its journey from being a detection-focused process to being one centered on defect prevention. Defect prevention is a mechanism used to evaluate the development and testing process and to identify the most effective improvements relating to product quality. It involves analyzing selected defects, identifying their root and common causes across teams, products and value streams, and defining specific actions to prevent the occurrence of similar defects in the future.

Quality control consists of the procedures and practices employed to ensure that a work product or deliverable conforms to requirements and standards. The principles of quality control can also be applied to the process creating the product, thereby creating a feedback loop in line with the prevention-oriented and optimizing approach of TMMi level 5. At TMMi level 5, organizations use quality control to drive the testing process. The process area Quality Control addresses the practices for establishing a statistically controlled test process, and for testing being performed, based on operational profiles and statistical techniques.

At the highest level of the TMMi, the test process is subject to continuous improvement across the entire organization. The test process is quantified, improved and fine-tuned, and capability growth is an on-going process. Test Process Optimization introduces a mechanism to continuously improve testing. The process area Test Process Optimization addresses the practices for continuously identifying test process improvements, evaluating and selecting new testing technologies and deploying them in the organization's standard test process.

The process areas at TMMi level 5 are:

5.1 Defect Prevention

5.2 Quality Control

5.3 Test Process Optimization

Each of these process areas is described in more detail in the sections hereafter.

Process Area 5.1 Defect Prevention

Purpose

The purpose of Defect Prevention is to identify and analyze root and common causes of defects found, and to define actions to prevent similar defects from re-occurring.

Value

Defect Prevention aims to address root cause issues, thereby reducing cost of quality, improving product quality and increasing productivity.

Introductory Notes

At TMMi level 5, testing completes its journey from being a detection-focused process to being prevention-focused. Industry data show that it is more cost-effective to prevent defects and issues from occurring than to detect defects and issues after they have been introduced [Boehm]. Defect prevention is a mechanism by which to evaluate the development and testing process and to identify the most effective improvements relating to product quality. As part of the Defect Prevention practices, trends are analyzed to track the types of defects that have been encountered, where they were introduced, and to identify those that are most likely to re-occur. Both product defects and process defects are in scope of the Defect Prevention process area.

Defect Prevention involves analyzing selected defects, identifying their root and common causes across teams, products and value streams, and defining specific actions to prevent the occurrence of similar defects in the future. The selection of defects to be analyzed is based on various factors, including impact and frequency of occurrence. Since defect prevention at this level needs measurement data, defect prevention builds upon the TMMi level 4 measurement practices and its data regarding development, testing and product quality. The test organization performs the Defect Prevention activities in close cooperation with other disciplines, e.g., business analysis, software development and IT operations, as proposed improvement actions will typically also affect them.

In an Agile and DevOps context, two types of defect prevention activities can be distinguished: those at team level and those at cross-team level. The team-related defect prevention activities (retrospectives) are the responsibility of the team and those were introduced at TMMi level 3. The cross-team defect prevention activities are performed by the test organization and it is these that are specifically addressed by Defect Prevention.

Scope

The process area Defect Prevention addresses the practices for identifying and analyzing root and common causes of defects, and defining specific actions in the format of improvement proposals, to prevent similar defects from re-occurring in the future. All defects are within the scope of the process area, whether found during development, testing or in the field. Process defects that have resulted in outliers or have not met expected process performance are also within scope. Defect Prevention builds on the TMMi level 4 measurement practices and available measurement data regarding development, testing and product quality.

Goal and Practice Summary

G1 Determine Root and Common Causes of Defects

- P1.1 Define defect selection parameters
- P1.2 Select defects for causal analysis

P1.3 Perform causal analysis for selected defects

G2 Define Actions to Systematically Eliminate Root Causes of Defects

P2.1 Propose solutions to eliminate root causes

P2.2 Define and submit improvement proposals

Practices by Goal

G1 Determine Root and Common Causes of Defects

Root and common causes of selected defects are systematically determined to increase the likelihood of meeting product quality and test performance objectives and preventing defects from re-occurring.

P1.1 Define defect selection parameters

Since it is impractical to perform causal analysis on all defects, selection parameters are defined for the defects to be analyzed for their root cause. Also, the defect classification scheme is revisited.

Example work products

1. Defect selection parameters
2. Updated defect classification scheme

Subpractices

1. Determine defect selection parameters for root cause analysis

Examples of defect selection parameters include:

- Impact
- Source
- Frequency of occurrence
- Similarity
- Cost of rework
- Safety considerations
- Security considerations

2. Review the defined defect selection parameters with stakeholders
3. Revisit and possibly update defect classification scheme

   The defect classification scheme and/or test incident report template initially defined in the Test Design and Execution process area, and possibly enhanced at later stages, e.g., with Test Measurement, may need to be updated. The defect classification scheme is revisited from a defect prevention perspective. For example, is all defect data that is needed for an effective and efficient defect prevention process recorded?

P1.2 Select defects for causal analysis

Defects are selected from the defect repository for causal analysis.

Example work products

1. Defects selected for causal analysis

Subpractices

1. Identify and involve stakeholders who need to contribute to the defect selection process
2. Prepare for defect selection

Examples of activities to be carried out during the preparation for defect selection include:

- Establish a comprehensive list of all defects. Product defects can originate from static testing, dynamic testing, and actual usage during operation.
- Make an initial defect selection. Defects that have a small likelihood of being selected, for instance minor defects, are removed from the list. Defects that adhere to the defect selection parameters are identified.
- Perform an initial analysis on the defects, e.g., identify defect types that have high occurrence.

3. Select defects (or defect types) for causal analysis

P1.3 Perform causal analysis for selected defects

Perform causal analysis on selected defects to determine their root causes and identify common causes.

Example work products

1. Root causes of selected defects
2. Common causes of defects

Subpractices

1. Identify and involve affected stakeholders
2. Perform root cause analysis on selected defects using both statistical and qualitative techniques

Examples of supporting techniques to determine root causes include [ISTQB-ITP]:

- Ishikawa fishbone diagrams
- Fault tree analysis
- 5 why's technique
- Use of defect classifications
- Failure Mode Effects Analysis (FMEA)
- Hardware Software Interaction Analysis

3. Determine common causes whereby the defects are grouped based on their root causes

Examples of categories of common causes include:

- Process
- People (skills and knowledge)
- (Project) organization
- Communication
- Architecture
- Technology, e.g., tools, test environment

G2 Define Actions to Systematically Eliminate Root Causes of Defects

Solutions are proposed and improvement proposals submitted to systematically address root and common causes of defects to improve performance by preventing defects from re-occurring.

P2.1 Propose solutions to eliminate root causes

Solutions are proposed to eliminate root and common causes.

Example work products

1. Solutions to address root and common causes

Subpractices

1. Cluster common causes that could be addressed through one or more related solutions, e.g., by development phase, technology, development lifecycle, test process or discipline
2. Determine the type(s) of solutions that are most likely to address the root and common cause, e.g., process, work product standards, training, review, test strategy or communication
3. Define a solution to address the root and common cause based on the identified type(s) of solutions

   Note, it is also possible that best practices within the organization are part of the solution.
4. Evaluate the solutions using statistical and qualitative techniques to determine if the proposed solution prevents the selected defects from re-occurring

Examples of techniques to evaluate proposed solutions include:

- Prototype
- Pilot
- Review
- (Manual) simulation
- Verify whether the change is statistically significant

5. Refine the proposed solution based on the results of the evaluation

P2.2 Define and submit improvement proposals

Improvement proposals based on the proposed solutions are defined and submitted.

Example work products

1. Improvement proposal
2. Root cause analysis and resolution data

Subpractices

1. Develop an improvement proposal

Examples of topics to be addressed in an improvement proposal include:

- Originator of the improvement proposal
- Affected stakeholders
- Defect(s) being addressed
- Description of the root cause
- Level of priority
- Description of the solutions
- Description of necessary tasks and actions
- Time, costs, and other resources required
- Expected benefits
- Estimated costs of not fixing the root cause

   When performed at Agile or DevOps team level, the improvement proposal may take the format of an additional backlog item.
2. Review the improvement proposal with stakeholders
3. Submit the improvement proposal according to the defined process
4. Record root cause analysis data and resolutions

Data is recorded so that projects or teams that experience similar root causes in the future can leverage the experience, and possibly implement a related type of solution.

Process Area 5.2 Quality Control

Purpose

The purpose of Quality Control is to statistically manage and control the test process. At this level, test process performance becomes fully predictable and stabilized within acceptable limits. Testing is performed using statistical techniques based on representative samples to provide reliable and quantitative evidence of product quality, and make testing more efficient.

Value

Quality Control aims to reduce defects and variability, thereby lowering costs, improving efficiency, establishing process predictability and optimizing testing effort while increasing reliability.

Introductory Notes

Quality control consists of the procedures and practices employed to ensure that a work product or deliverable conforms to requirements and standards. The principles of quality control can also be applied to the process creating the product, thereby creating a feedback loop in line with the prevention-oriented and optimizing approach of TMMi level 5. At TMMi level 5, organizations use quality control to drive the testing process.

Process quality control starts by defining test performance objectives for a selected set of test processes from the standard test process, as defined at TMMi level 3. Based on historical data, baselines for these test processes are established that represent the typical or expected performance of a process under normal conditions. Subsequently, control limits are statistically determined that define the natural variation of the process. If the process outputs stay within these limits, it is considered "under control". Observations outside these limits indicate special cause variation, signaling potential problems that require investigation and corrective action. Test process data is plotted over time, often in the form of control charts (see figure 4.1), to visualize trends, shifts, or patterns. Over time, process quality control supports test process improvement by reducing variability, lowering defects, enhancing efficiency and establishing a predictable, consistent test process capability.

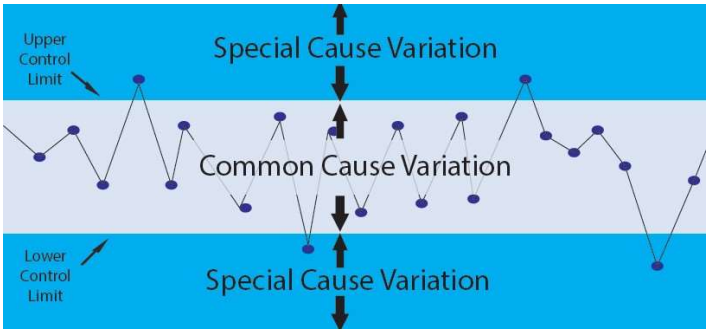

Figure 4.1: Example Control Chart

Product quality control builds on usage models and operational profiles of the products in their intended environment to make statistically valid inferences, resulting in a representative sample of test cases. This approach, especially useful at the system test level, uses statistical techniques to predict product quality based on this representative sample. In other words, when testing a subset of all possible usages (as represented by the usage model or operational profile) the test results can serve as the basis for conclusions about the overall performance of the product. Product quality and reliability objectives, along with their confidence levels, are defined. By using test results and statistical models, it is possible to determine whether the objectives are achieved. At TMMi level 5, these factors are typically the main drivers used to determine when to stop testing. Note that addressing product quality control and statistical testing requires a great deal of expertise in statistical techniques, including usage modeling, reliability modeling, statistics, and measurement. Specialists must be selected and trained to become leaders in this area of testing.

Scope

The process area Quality Control addresses the practices for establishing a statistically controlled test process (process quality control [Goal 1]), and testing being performed based on operational profiles and statistical techniques (product quality control [Goal 2]). Both types of quality control build strongly on the deployed measurement practices and stored test measurement data of the Test Measurement process area at TMMi level 4. Product quality control and statistical testing typically also build on the practices and measurement data of the Product Quality Evaluation process area at TMMi level 4.

Goal and Practice Summary

G1 Establish a Statistically Controlled Test Process
- P1.1 Establish test process performance objectives
- P1.2 Establish test process performance measures
- P1.3 Establish test process performance baselines
- P1.4 Understand and address test process performance variations
- P1.5 Monitor test process performance against objectives

G2 Perform Testing using Statistical Techniques
- P2.1 Develop operational profiles
- P2.2 Generate and execute statistically selected test cases
- P2.3 Apply statistical test data to make stop-test decisions

Practices by Goal

G1 Establish a Statistically Controlled Test Process

A statistically controlled test process is established whereby statistical techniques, e.g., control charts, are used to monitor, control and improve the organization's standard test processes to achieve reliable and improved performance.

P1.1 Establish test process performance objectives

Establish and maintain quantitative objectives for test process performance.

Example work products

1. Test process performance objectives (traceable to business objectives)

Subpractices

1. Study the business needs and objectives regarding product quality and test process performance
2. Study the test policy with respect to the defined test goals and test performance indicators

   Refer to the process area 2.1 Test Policy and Strategy for more information on business needs and objectives, test goals and test performance indicators.
3. Define the organization's objectives for test process performance in cooperation with stakeholders

   Objectives are established directly for test process performance (e.g., test effort and defect removal effectiveness) or indirectly for product quality (e.g., reliability) that are the result of the test process. The objectives for product quality are thereby covered by the defined set of test process performance objectives.

Examples of test process performance objectives include:

- Achieve a specified level of test productivity
- Deliver products with no more than a specified number of latent defects

4. Resolve conflicts among the test process performance objectives, e.g., if one objective cannot be achieved without compromising another
5. Review the defined objectives for test process performance with affected stakeholders
6. Identify and select test processes that that are expected to be the largest contributor to meeting the defined set of test process performance objectives, and as such need to be included in the set of statistically controlled test processes.

   Typically, it will not be possible, useful, or economically justifiable to apply statistical management techniques to all test processes of the organization's standard set of test processes.
7. Revise the organization's objectives test process performance as necessary

P1.2 Establish test process performance measures

Identify and select test measures and statistical analysis techniques to quantitatively manage test performance and achieve test process performance objectives.

Example work products

1. Test process performance measures (for selected test processes)
2. Selected statistical analysis techniques

Subpractices

1. Identify test measures from the organization's test measurement repository that support appropriate insight into test process performance and achievement of objectives

   Note, additional test measures may need to be identified that cover critical work products and test process attributes of the selected test processes.
2. Select test measures that are appropriate to manage test processes using statistical and other quantitative techniques
3. Specify operational definitions for newly identified test measures
4. Identify statistical analysis and other quantitative techniques to be used

   These statistical and other quantitative techniques should help to characterize process variation, recognize when variation is excessive and identify when statistically unexpected behavior occurs.

Examples of statistical techniques to analyze test process performance include:

- Statistical process control
- Regression analyses
- Analysis of variance including central tendency and distribution
- Hypothesis testing for statistical significance

5. Incorporate the selected test measures into the organization's test measurement process
6. Revise the test measures for test process performance as necessary

P1.3 Establish test process performance baselines

Use statistical and other quantitative techniques to establish and maintain baselines for the selected test processes. The organization's test process baselines are a measurement of test performance at various levels of detail, as appropriate.

Example work products

1. Test process performance baselines

Subpractices

1. Collect and analyze test measurements from projects or teams

   Data is collected from within the organization, projects or teams for specified base measures. Derived test measures are calculated.

2. Establish and maintain the organization's test process performance baselines from the collected test measurements and analyses

   Test process performance baselines include a central tendency and distribution or range of results that characterize the expected test process performance.

3. Review the validity of, and get agreement with stakeholders about the test process performance baselines

4. Make the test process performance baselines available across the organization through the organization's test measurement repository

5. Revise the set of test process performance baselines as necessary

P1.4 Understand and address test process performance variations

Use statistical techniques, e.g., statistical process control, to establish and maintain an understanding of test process performance variations. Root causes of special cause outliers are identified and addressed to improve test process performance.

Example work products

1. Control limits for test process performance measures
2. Root causes of outliers
3. Solutions to eliminate root causes of outliers

Subpractices

1. Determine control limits (upper and lower boundaries) for test process performance measures based on test measurement data and analysis

   Control limits of an attribute are the range within which variation normally occurs. All processes will show some variation in measurements each time they are executed. Typically control limits (upper and lower boundaries) are set to 2 or 3 sigma. The control limits help to separate signals from noise. The variation of data points inside the control limits is due to noise in the process (see figure 4.1).

2. Collect test measurements from projects or teams

3. Evaluate test measurements in relation to natural bounds and control limits identifying outliers or other signals of potential non-random behavior

   Examples of tools used to identify outliers with statistical process control include:
   - Control charts
   - Run charts
   - Scatter diagram

4. Perform root cause analysis for outliers

   An important activity during the identification of causes of variation is to determine if a process variation is caused by special circumstances (special cause) or by a variation inherent in the nature of

the process itself (natural cause). Note that some outliers may simply be extremes of underlying distribution rather than problems.

Refer to the process area 5.1 Defect Prevention, especially practice P1.3 “Perform causal analysis for selected defects”, for more information on performing causal analysis to determine root causes and identify common causes.

5. Propose solutions to eliminate root causes

   Refer to the process area 5.1 Defect Prevention for more information on proposing solutions to eliminate root causes (practice P2.1), and possibly subsequently defining and submitting improvement proposals (practice P2.2).

6. Revise the control limits for test process performance measures as necessary

P1.5 Monitor test process performance against objectives

Monitor the performance of the selected test processes to determine their capability to satisfy the defined test process performance objectives and identify corrective actions as needed. Test process behavior exhibited is statistically determined together with the probability that the test process will meet its test process performance objectives.

Example work products

1. Test process performance results
2. Test process capability deficiencies
3. Test process improvement actions

Subpractices

1. Analyze test measurements and test performance objectives against current performance to evaluate the ability to satisfy objectives.
2. Identify potential improvement areas where actual test performance is not meeting its objectives
3. Periodically, review the performance of each selected test process for its capability to manage progress towards achieving the test process performance objectives
4. Communicate results to affected stakeholders
5. Identify and document test process capability deficiencies
6. Determine and document actions needed to address test process capability deficiencies

G2 Perform Testing using Statistical Techniques

Tests are designed and executed guided by statistical techniques based on operational or usage profiles to provide reliable and quantitative evidence of product quality under actual usage conditions, while using testing resources efficiently.

P2.1 Develop operational profiles

Operational profiles (or usage models) are developed and maintained to serve as a basis from which a well—founded statistically based sample of test cases can be derived.

Example work products

1. Operational profile of the system to be tested

Subpractices

1. Develop the user profile

The user profile is the complete set of user groups (the set of actual users that will engage the system in the same way) and their associated frequency distribution across the profile.

2. Develop the system mode profile

   The system mode profile is the set of system modes (a set of functions or operations grouped in order to analyze execution behavior) and their associated occurrence probabilities.

3. Develop the functional profile

   The functional profile provides (per system mode) a quantitative view of the relative usage of each of the different system functions.

4. Develop the operational profile

   An operation represents a task being accomplished by a system. The final operational profile is developed by a series of steps using information from the profiles already developed. This includes dividing the execution into runs, identifying the input space, partitioning the input space into operations and determining the occurrence probability for the operations [Musa].

5. Review and agree the operational profile with stakeholders
6. Revise the operational profile as necessary

P2.2 Generate and execute statistically selected test cases

Test cases are generated based on statistically selected samples of product usage, and subsequently executed.

Example work products

1. Test cases
2. Test results

Subpractices

1. Generate test cases using random or pseudo-random sampling ensuring the selection follows the operational profile

   The set of test cases should reflect probabilities in the operational profile and represent a sample of the input space according to the usage patterns.

2. Execute test cases and record actual results, e.g., defects found
3. Monitor test coverage for actual usage

   Testing will use tools and measurements to determine if the set of executed test cases is representative of actual usage. Only when the tests are representative can the test results together with other data be used to make stop-test decisions.

4. Enhance the set of the test cases when test coverage of actual usage is not sufficient
5. Fix and re-test defects
6. Repeat and continue testing until the decision is made to stop testing (see P2.3 hereafter)

P2.3 Apply statistical test data to make stop-test decisions

In statistical testing using operational profiles, a stop-test decision determines when testing can be ended with acceptable confidence that product quality and reliability goals are met.

Example work products

1. Product quality and reliability objectives
2. Product quality and reliability measurements
3. Documentation of stop-test decision taken

Subpractices

1. Define quantitative product quality and reliability objectives to be used as exit criteria (definition-of-done) and to make stop-test decisions

Examples of types of product quality and reliability objectives include:

- Maximum acceptable defect probability
- Maximum acceptable defect density
- Required Mean Time Between Failure (MTBF)
- Defect-free execution goals, e.g., system must execute x operations with zero defects
- Confidence-level based goals, e.g., with 95% confidence, MTTF ≥ x hours

2. Review the product quality and reliability objectives with stakeholders
3. Select a statistical reliability model

Examples of types of reliability models include [Musa and Ackerman]:

- Static model, best applied to unchanging software with an unchanged operational profile
- Basic model, useful for modeling failure occurrences for software being tested and continuously debugged
- Logarithmic Poisson model, which is best applied when it is assumed that some defects are more likely to cause failures, and that, on average, the improvement in failure intensity decrease exponentially with each correction.

4. Collect data on defects and execution time

   Continuously record number of tests executed, number of defects and time between defects/failures.

5. Apply statistical decision rules at checkpoints to evaluate level of achievement of exit criteria
6. Make stop-test decision: stop testing if the objectives are met with required confidence, or continue testing if objectives are not yet satisfied.
7. Document stop-test decision taken

   Record assumptions, data, confidence levels, and conclusions.

Process Area 5.3 Test Process Optimization

Purpose

The purpose of Test Process Optimization is to continuously improve the existing testing processes and test process assets used in the organization and identify new testing technologies and transition them into the organization as appropriate. The improvements support the achievement of organization's product quality and test process performance objectives as derived from the organization's business objectives.

Value

Test Process Optimization aims to ensure that the test processes, test process assets and their improvements contribute to successfully meeting business objectives.

Introductory Notes

At the highest level of the TMMi, the test process is subject to continuous improvement across the entire organization. The test process is quantified, improved and fine-tuned, and capability growth is an on-going process. An organizational infrastructure exists to support this continuous growth. This infrastructure, compromising policies, standards, training facilities, tools and organizational structures, has been created through goal achievement processes that constitute the TMMi maturity level hierarchy. Test Process Optimization is about developing a system to continuously improve testing.

To support Test Process Optimization, the organization will typically establish a Test Process Group (TPG) that focuses on process-related activities such as process definition, analysis and assessment, action planning, and evaluation. The TPG builds on the test organization already formally established at TMMi level 3. The TPG will also be the coordinating body for other process-related activities including quality control and defect prevention. The organization now fully understands the value of process change and test improvement and is willing to provide the group with adequate resources and funding. Ideally, the TPG operates as a permanent organizational entity guided by developed and distributed charter or policy statement.

Continuously improving the testing process means proactively identifying and evaluating improvement opportunities. These improvements are then systematically implemented in the organization's standard test process and project-specific processes. Often, test process improvement activities are needed as a result of a changing environment, e.g., the business context, the test environment itself or a new development lifecycle. All of this is done with sponsorship of higher-level management. Training and incentive programs are established to enable and encourage participation in test process improvement activities. Test improvement opportunities are identified and evaluated for their potential return on investment guided by the organization's business goals and objectives. Pilots are performed to assess, measure and validate the test process changes before they are incorporated into the organization's standard process.

As part of test process optimization, the centrally organized Test Process Group (or test competence center) proactively searches the market for new technologies that can improve the testing capability of the projects and teams, such as tools, methods, techniques or technical innovations. By maintaining an awareness of test-related technology innovations and systematically evaluating and experimenting with them, the organization, in consultation with team representatives, selects appropriate testing technologies to improve the quality of its products and the productivity of its testing activities. Pilots are performed within projects and teams to assess new and unproven testing technologies before they are incorporated into standard practice.

Test improvements and appropriate new testing technologies are deployed across projects and teams to improve testing. Test improvements and new testing technologies are incorporated into the organizational test process assets as appropriate, with training and support being organized and offered. Using statistical and other quantitative techniques improvements are validated against defined product quality and test performance objectives. Information about innovations is disseminated across the organization.

Scope

The process area Test Process Optimization addresses the practices for continuously identifying test process improvements, evaluating and selecting new testing technologies and deploying them in the organization's standard test process, including planning, establishing, monitoring, evaluating and measuring the test improvement actions. This process area complements and extends the processes and practices defined by the Test Organization and Test Process and Integration process areas at TMMi level 3.

Goal and Practice Summary

G1 Select Test Process Improvements
- P1.1 Collect and analyze test process improvement proposals
- P1.2 Pilot test process improvement proposals
- P1.3 Select test process improvement proposals for deployment

G2 Evaluate New Testing Technologies
- P2.1 Identify and analyze new testing technologies
- P2.2 Pilot new testing technologies
- P2.3 Select new testing technologies for deployment

G3 Implement and Deploy Test Improvements
- P3.1 Plan test improvements
- P3.2 Implement test improvements
- P3.3 Manage the deployment
- P3.4 Measure improvement effects

Practices by Goal

G1 Select Test Process Improvements

Test process improvements are selected, which are expected to contribute to meeting product quality and test process performance objectives, to maximize return on investment by focusing resources on those improvements that have most added value.

P1.1 Collect and analyze test process improvement proposals

Continuously collect and analyze test improvement proposals for the organization's standard test process from within the organization.

Example work products

1. Analyzed test process improvement proposals
2. List of test process improvement proposals to be piloted

Subpractices

1. Continuously collect test process improvement proposals from internal sources, e.g., from test assessments, lessons learned documents, test measurements, team retrospectives and defect prevention activities such as causal analysis
2. Analyze the costs and benefits of the test process improvement proposals

   Test process improvement proposals that do not have an expected positive return on investment are rejected. The major criterion is the expected contribution of the test process improvement proposal to the organization's product quality and test process performance objectives.
3. Analyze the project risks of the test process improvement proposals

Examples of risk factors for test process improvements include:

- Complexity of the improvement
- Knowledge and skills of potential users
- Lack of short-term visible benefits
- Resistance

4. Estimate the effort required for implementing and deploying the test improvement proposals
5. Identify the process improvement proposals that need to be piloted prior to a decision for implementation and organization-wide deployment

   Alternatives to piloting can be considered as appropriate, e.g., controlled experiments, simulations or case studies.
6. Document the results of the test process improvement proposal analysis

P1.2 Pilot test process improvement proposals

Pilots for the test process improvement proposals are performed, as appropriate, to evaluate new and unproven major changes to the organization's standard test process and determine whether they are beneficial to implement and deploy.

Example work products

1. Pilot evaluation reports

Subpractices

1. Plan the test process improvement pilots

   Ensure each pilot is performed in an environment that is sufficiently representative of the real-life environment.
2. Define pilot objectives and evaluation criteria
3. Review the test process improvement pilot plans with stakeholders
4. Perform the test process improvement pilots

   Allow for additional resources for the pilot projects as necessary.
5. Coach and support the projects or teams performing the pilots
6. Evaluate, review and document the results of the pilots

Examples of actions to be performed at the evaluating of the pilots include:

- Evaluate the results of the pilot against the defined objectives and evaluation criteria
- Update the test process improvement proposal based on the pilot results
- Identify and document lessons learned encountered during the pilot

P1.3 Select test process improvement proposals for deployment

Test process improvement proposals are selected for implementation and deployment across the organization.

Example work products

1. Test process improvement proposals selected for deployment
2. Selection process results (including rationale for decisions made)

Subpractices

1. Prioritize the candidate test process improvements proposals for deployment

Examples of criteria to be used during the prioritization include:

- Expected contribution to test performance objectives
- Expected contribution to product quality objectives
- Expected costs and benefits
- Expected risk to be mitigated
- Ease of implementation
- Expected level of acceptance (some may initially cause resistance)

2. Select the test process improvement proposals to be implemented and deployed based on their priorities and available resources
3. Review selection with affected stakeholders
4. Document the results of the selection process and communicate results

G2 Evaluate New Testing Technologies

New testing technologies such as methods, techniques, tools or technical innovations are identified and evaluated to determine their effect on the organization's standard test process and product quality. This goal aims to enhance test innovation and achieve test objectives more efficiently and effectively.

P2.1 Identify and analyze new testing technologies

Continuously identify and analyze new testing technologies which could improve test process performance and product quality.

Note that the practice P1.1 "Collect and analyze test process improvements", analyzes proposals that are collected *internally*. The purpose of the practice P2.1 is to actively search *externally* for interesting new testing technologies.

Example work products

1. Analyzed new testing technologies
2. List of new testing technologies to be piloted

Subpractices

1. Continuously identify new testing technologies

Examples of identification activities include:

- Systematically maintaining awareness of leading technical testing work and trends, e.g., by visiting test conferences, attending webinars and studying literature
- Periodically searching for commercially available innovative and new testing technologies
- Studying new testing standards for their applicability in the organization

- Reviewing test processes, tools and methods used externally
- Benchmarking test process performance against industry performance data
- Reviewing examples of test process improvements that were successfully adopted elsewhere
- Participating in special interest groups in testing
- Collaborating with other improvement initiatives in the organization to find opportunities that could also be beneficial to testing

2. Analyze new testing technologies to understand their effects on the organization's standard test process

   As part of the analysis, consider constraints, prioritization of possible features, hardware/software issues, suppliers' track records and integration with existing test technologies and processes.

3. Analyze the costs and benefits of the new testing technologies

   New testing technologies that do not have an expected positive return on investment are rejected. The major criterion is the expected contribution of the new testing technology to the organization's product quality and test process performance objectives. Both short-term, and long-term recurring (maintenance) costs should be taken into account. As part of this subpractice, alternative solutions, e.g., a test process change, that can possibly provide the same benefits are considered.

4. Analyze the project risks of the new testing technologies
5. Estimate the effort required for implementing and deploying the new testing technologies
6. Identify new testing technologies that need to be piloted prior to a decision for implementation and organization-wide deployment
7. Document the results of the evaluation of new testing technologies and communicate results

P2.2 Pilot new testing technologies

Pilots for the new testing technologies are performed, especially those that have a major impact, to determine whether they are beneficial to implement and deploy organization-wide.

Example work products

1. Pilot evaluation reports

Subpractices

1. Plan the new testing technologies pilots

   Ensure each pilot is performed in an environment that is sufficiently representative of the real-life environment.

2. Define pilot objectives and criteria to evaluate results
3. Review the new testing technologies pilot plans with stakeholders
4. Perform the new testing technologies pilots

   Allow for additional resources for the pilot projects as necessary.

5. Coach and support the projects or teams performing the pilots
6. Evaluate, review and document the results of the pilots

P2.3 Select new testing technologies for deployment

New testing technologies are selected for implementation and deployment across the organization.

Example work products

1. New testing technologies selected for deployment

2. Selection process results (including rationale for decisions made)

Subpractices

1. Prioritize the candidate new testing technologies for deployment

Examples of criteria to be used during the prioritization include:

- Expected contribution to test performance objectives
- Expected contribution to product quality objectives
- Expected costs and benefits
- Expected risks to be mitigated
- Ease of implementation
- Expected level of acceptance (some may initially cause resistance)

2. Select the new testing technologies to be implemented and deployed based on their priorities and available resources
3. Review selection with affected stakeholders
4. Document the results of the selection process and communicate results

G3 Implement and Deploy Test Improvements

Selected test process improvements and new testing technologies are implemented and deployed across the organization to further improve the testing process. Test improvements are planned, implemented and deployed to ensure the test process and its improvements contribute to meeting business objectives.

P3.1 Plan test improvements

Establish and maintain plans for the implementation and deployment of the selected test process improvements and new testing technologies.

Example work products

1. Test improvement plan or action plans (covering implementation and deployment)

Subpractices

1. Group related test process improvements and new technologies that are suitable for joint implementation and deployment
2. Define the approach to establish the test process improvements and new testing technologies

Examples of consideration when defining the approach include:

- Does the improvement need to be piloted during implementation before it is fully deployed?
- Will the improvement be deployed in all projects or teams, or only in a subset of the organization's projects or teams?
- Is the new or updated process mandatory for all projects and teams, or can Agile / DevOps teams decide for themselves whether the test improvement will become part of their way-of-working?

3. Identify and define actions needed to implement and deploy the test process improvements and new testing technologies

Examples of implementation and deployment actions include:

- Update test process assets
- Update test environments
- Training of affected staff
- Coaching of projects and teams

4. Establish process action teams to implement and deploy the test process improvements
5. Identify project risks to the implementation and deployment and define appropriate mitigation actions
6. Determine marketing and change management activities to support deployment

Examples of change management activities include:

- Communications to stakeholders
- Kick-off with all parties involved
- Discussion sessions
- Publications, e.g., for information purposes and on successes achieved

7. Establish objectives and measures (acceptance criteria) for confirming the value of each test process improvement and new testing technology

Examples of measures for confirming the value include:

- Improvement in test process performance
- Improvement in product quality
- Return on investment
- Payback period

Refer to process area 4.1 Test Measurement for more information on establishing measures and the measurement and analysis process.

8. Align test improvement activities with other improvement initiatives from within the organization
9. Document the plan for implementing and deploying the test process improvements and new testing technologies

Refer to process area 3.1 Test Organization for more details on a test process improvement plan.

10. Review the plan for implementing and deploying the test process improvements and new testing technologies with affected stakeholders, e.g., management and members of process action teams
11. Revise the plan for implementing and deploying the test process improvements and new testing technologies as necessary

P3.2 Implement test improvements

The test improvements, addressed by the test improvement plan or action plans, are implemented in the organization's standard test process.

Example work products

1. Results (solutions) of implementing test improvements, e.g., newly developed or updated test process assets including training material
2. Implementation status reports

Subpractices

1. Create a solution to address the test improvement, e.g., develop or update process description, templates, tools and/or training
2. Pilot the solution as needed in one or more projects or teams
3. Evaluate the results of the pilot against improvement objectives with stakeholders
4. Refine the solution based on the results and evaluation of the pilot
5. Incorporate the test improvements into the organization's standard test process, as appropriate

6. Monitor and manage progress against the test improvement plan
7. Provide status reports on the implementation of the test improvements to stakeholders

P3.3 Manage the deployment

Manage the deployment of the test process improvements and new testing technologies.

Example work products

1. Deployment status reports

Subpractices

1. Coordinate the deployment of test process improvements and new testing technologies across the organization and within projects and teams
2. Align the deployment with other improvement initiatives from within the organization
3. Conduct training on the test process improvements and new testing technologies to projects, teams and other affected stakeholders
4. Provide coaching to projects and teams to support the deployment of the test process improvements and new testing technologies
5. Perform marketing, both inside and outside testing, regarding achievements to keep staff motivated and involved
6. Monitor and manage deployment progress of the test process improvements and new testing technologies

   Monitoring deployment is typically supported by objective process evaluations. (Refer to process area 3.2 Test Process and Integration for more information on process adherence evaluations.)
7. Confirm that the deployment of the test process improvements and new testing technologies is completed
8. Document and review the deployment results, including lessons learned, of the test process improvements and new testing technologies with affected stakeholders

For more information on deployment refer to the goal G3 “Deploy the Organization Test Process and Test Process Assets” of the process area 3.3 Test Process and Integration.

P3.4 Measure improvement effects

Use statistical and other quantitative techniques to measure the effect of deployed test process improvements and new testing technologies against improvement expectations (acceptance criteria), business objectives, product quality and/or test performance objectives.

Example work products

1. Measures of the effects of the test improvements
2. Improvement effects measurement reports

Subpractices

1. Measure the cost and effort of applying the test improvements
2. Measure the benefits of the test improvements
3. Measure the contribution of the test improvements to achieving the organization’s test process performance objectives
4. Analyze the contribution of the test improvements to achieving the organization’s test process performance objectives, and take corrective action as needed

5. Record the results and communicate with affected stakeholders
6. Store the measures in the organization’s test measurement repository

Annex A Artificial Intelligence and TMMi

A.1 Introduction

Software testing is rapidly evolving with the rise of Artificial Intelligence (AI). AI can support testers by automatically generating test cases, analyzing large amounts of test data, and detecting defects or regressions more quickly, making testing processes more efficient and intelligent. At the same time, AI introduces new challenges, such as ensuring the reliability and transparency of AI models, security risks, dealing with potential bias in data, and requiring testers to develop new skills. An important aspect is the role of the tester in evaluating the results produced by generative AI (GenAI). Testers must critically assess whether the generated test cases, analyses, or suggestions are correct, relevant, and complete. To do this effectively, a mature and well-defined test process is required (e.g., based on TMMi), which testers must thoroughly understand. While AI can enhance testing activities, testing processes and human expertise both remain essential to validate outcomes and ensure quality.

Before discussing the relationship between AI and TMMi in detail, we first need to point out an important difference between “Testing AI systems" and "Using AI for testing". These are two different concepts both relate to the application of Artificial Intelligence (AI) in the software testing domain.

Testing AI-systems refers to the process of evaluating and validating the performance, functionality, and quality of software applications or systems that incorporate AI components. The primary goal is to ensure that AI algorithms, models, and functionalities within a software system work as expected, produce adequate results, and meet the desired quality standards. Testing AI systems involves a variety of testing activities such as input data testing, model testing, integration testing, functional testing, performance testing, security testing, compliance testing, and testing for ethical considerations like bias and fairness. Note, that is also possible to use AI when testing AI-based systems thereby combining the two concepts.

Using AI in testing involves the application of artificial intelligence techniques and tools to enhance and optimize (activities within) the software testing process itself. The primary objective is to enhance testing efficiency, accuracy, and coverage by using AI technologies to support and partially automate various testing activities while enabling data-driven decision-making. However, AI can also be applied in testing for activities such as estimation, defect prediction, test analysis, test case generation, test data generation, test execution, and test result analysis, among others. AI can also help to prioritize test cases based on risk and historical data.

In summary, "testing AI systems" is about ensuring the quality of AI-driven software, while "using AI in testing" involves leveraging AI to improve the efficiency and effectiveness of the testing process itself. In the context of this document the scope is using AI in testing and how it can support the various TMMi process areas. The TMMi Foundation has also released the document “Testing AI-systems and TMMi” [TMMiAI] which explains how the TMMi test improvement model can be used and applied beneficially when testing AI systems. The TMMi level 2 and 3 process areas and their goals are discussed individually. It is stated in this document how these relate to testing AI systems and what the practices will typically look like.

Hereafter, each of the TMMi process areas is discussed, organized by maturity level. It is stated whether AI can support the TMMi improvement goals of the process area, and what the AI/GenAI support typically covers and comprises. In general, more support is available for the test-engineering-oriented TMMi process areas than for the organizational process areas. Although GenAI support is provided per process area, this does not imply it can always be implemented in parallel. For example, for test estimation, historical data is needed to

be able to use GenAI effectively. Typically, at TMMi level 2, a large amount of historical data is not yet available and only becomes available at higher TMMi levels. Also, the application of AI for a TMMi process area can also imply that the test process itself will need to be changed to allow for a streamlined integration and/or make better use of the possibilities of AI tooling. When starting to use AI to support test processes, it is essential for an organization to have a well-defined AI adoption and implementation strategy in place to ensure success. This should include building extensive knowledge and skills and an approach to prompt engineering. Since GenAI will be applied across various TMMi process areas a coherent approach is needed, whereby their interrelationships and traceability are managed and monitored. It is also important that AI-driven recommendations and actions shall operate within a governed framework, where human-in-the-loop validation, defined approval checkpoints, and auditability mechanisms ensure that AI outputs are reliable, transparent, and aligned with organizational and regulatory requirements.

Note that, given the rapid evolution of AI, the examples provided in the following section should be regarded as indicative rather than complete, definitive, or final. Throughout the exercise of identifying and defining AI support for the TMMi process areas, the following sources were used: [ISTQB-AI], [ISTQB-GenAI], [Ruigrok] and [Veenendaal25].

Remember that the benefits that can be achieved with AI, will only be achieved when the underlying process is defined and thoroughly understood, and when those involved posses the knowledge and skills to evaluate and validate the AI generated results.

A.2 AI support for TMMi process areas

TMMi Level 2 Process Area	AI/GenAI support and application
2.1 Test Policy and Strategy	**G1 Establish a Test Policy** Providing the test vision of the organization, policy statements and domain context, blended with some examples of test policies, AI can generate a draft test policy that can subsequently be discussed within the organization. The document should also address the policy for using AI in testing. **G2 Establish a Test Strategy** Providing AI with a list of reported field defects, allows for a review of the current test strategy and for AI to propose improvements to the defined organizational test strategy. **G3 Establish Test Performance Indicators** GenAI can provide guidance on selecting appropriate test performance indicators aligned with test process improvement objectives set by management. It can also provide the operational definition for the selected test performance indicators. Subsequently, using scripts, it can support data gathering and calculate the test performance indicators.
2.2 Test Planning	**G1 Perform a Product Risk Assessment** Perform a Product Risk Assessment can be supported by defect prediction. Defect prediction can be used to predict whether a defect is present, how many defects are present, or whether defects can be found. Results can be used to prioritize testing. Defect prediction is typically based on source code metrics, process metrics and/or people and organizational metrics. Note that defect

TMMi Level 2 Process Area	AI/GenAI support and application
2.2 Test Planning (continued)	prediction is most effective when based on prior experiences in a similar situation (e.g., with the same code base and/or the same developers). **G2 Establish a Test Approach** Based on an analysis of the requirements (user stories), GenAI can propose the most appropriate test design techniques (e.g., boundary value analysis and/or equivalence partitioning) and a test approach. **G3 Establish Test Estimates** GenAI can compare a new user story that has been assigned a risk level, with a large set of reference user stories to find the closest match and propose estimates based on historical data, taking various parameters into account. **G4 Develop a Test Plan** GenAI can help optimize planning of test resources and scheduling of test tasks, by analyzing test cases and their interdependencies, priorities, risks, dependencies between user stories and available test resources.
2.3 Test Monitoring and Control	**G1 Monitor Test Progress and G2 Monitor Product Quality** GenAI can generate monitoring scripts and assist in establishing monitoring processes, enabling faster and less resource-intensive monitoring. GenAI can also generate test reports, charts, real-time dashboard and natural language summaries about progress, quality and risks, highlighting trends and predictive insights. This support helps ensure that all stakeholders have access to relevant information on status and progress, enabling well-informed decision-making. **G3 Manage Corrective Actions to Closure** GenAI can analyse data produced by monitoring activities and, using predictive analytics, recommend control actions or even initiate them automatically. It can, for example, re-prioritize tests, adjust test schedules and re-allocate resources as needed.
2.4 Test Design and Execution	**G1 Perform Test analysis and Design** During *test analysis*, the test basis is reviewed by the tester to determine its testability, answering the question “Can test cases, including expected results, be derived from the test basis?” GenAI can identify ambiguities, inconsistencies, or missing information. It can also provide advice on the completeness and correctness of the acceptance criteria for user stories. This is an activity where AI can provide huge support, both in terms of efficiency and effectiveness. Applying a rule set and providing defect lists from previous reviews, will enhance the quality of the output, e.g., the review log. Note, that GenAI can also be applied during requirements elicitation and definition, helping to improve the overall quality of requirements provided as input to the test analysis activity. *Test design* is the activity to identify test conditions and develop test cases using appropriate test techniques. GenAI can support the identification of test conditions. Providing a description of, or reference to, the technique being used

TMMi Level 2 Process Area	AI/GenAI support and application
2.4 Test Design and Execution (continued)	is recommended to ensure consistent output. It is also advisable to supply an output test specification template or an example of a previous test specification. An example output could be a table showing equivalence classes covered and a set of test cases. A significant weakness of AI-generated tests is the lack of traceability. It is therefore important to ensure that traceability anchors are embedded within the test design generation process. By mapping requirements or user stories to test conditions, AI can perform coverage analysis to determine whether all aspects of the test basis are covered. Note that with GenAI, it is easy to generate a large number of test conditions and test cases, but quantity does not necessarily equate to depth. GenAI can also assist in generating ideas, and thereby test charters, for exploratory testing. These ideas may serve as a starting point or enhancement, but they should not replace the testers' creativity, which is essential to the value of exploratory testing. **G2 Perform Test implementation** Using AI to generate tests can be a highly effective way to rapidly create component tests and maximize code coverage. These tests are generated typically from the source code and user interface, and may also draw on a machine-readable test model. GenAI can generate test cases (including preconditions and inputs) for system testing through to complex end-to-end testing, based on identified test conditions and/or test objectives, using system requirements, user stories or other elements of the test basis. GenAI can also generate the expected results. GenAI can then create manual test procedures or automated test scripts based on the set of test cases, interpreting test steps and translating them into code compatible with various test automation frameworks, including keyword-driven test automation frameworks. Note, AI-generated test cases and fixes require human review before use, especially in regulated or high-risk context. **G3 Perform Test execution** GenAI can orchestrate a test automation suite and schedule the execution of automated test cases, enabling continuous testing. It can also analyze test results and compare them with expected outcomes. To prevent regression test suites from becoming too large, they should be regularly optimized by selecting, prioritizing, and even augmenting test cases to maintain an effective and efficient suite. For example, an AI-based tool can optimize the regression test suite by analysing previous test results, related defects, and recent system changes, such as frequently failing features or tests that cover code affected by those updates. For minor user interface or API changes, GenAI can be used to automatically adjust test scripts to prevent unnecessary failures ensuring that test scripts remain stable over time.

TMMi Level 2 Process Area	AI/GenAI support and application
2.4 Test Design and Execution (continued)	GenAI can support analyzing test results by creating summaries and classifying incidents found by severity and priority. It can also help with automatic compilation of comprehensive incident reports including test logs, screenshots and test environment data. **G4 Manage Test Incidents to Closure** To support incident reporting and managing incidents to closure, AI can be used to categorize incidents, analyze text within incident reports and extract information, such as the area of functionality affected. AI models can also suggest which developers are best suited to fix particular defects, based on the defect content and previous developer assignments.
2.5 Test Environment	**G1 Develop Test Environment Requirements** GenAI can support the elicitation and definition of requirements for software test environments and test data by analysing existing systems, documentation, and user-provided inputs, such as emails and meeting minutes, to suggest necessary configurations, dependencies, and data scenarios. It can help identify gaps and translate high-level needs into detailed requirements. GenAI can also be used to harmonize language, enforce the use of template and eliminate vague and complex wording. **G2 Perform Test Environment Implementation** GenAI can support creating Infrastructure-as-code scripts or analyzing existing scripts to suggest improvements. GenAI can generate test data sets, set boundary values and create different combinations of test data. Regulatory requirements often mandate the use of anonymized or synthetic test data. GenAI can effectively support the anonymization process and can also generate synthetic test data that closely resembles production data, including edge cases and diverse test conditions. **G3 Manage and Control test environments** GenAI can assess how test environments are being used and automatically adjust their configuration when performance thresholds are exceeded. GenAI can also optimize the use of test environments based on historical and current project data.
2.6 Implementation and Habit	**G3 Manage configurations** AI and GenAI can enhance configuration management by automating configuration creation, supporting troubleshooting, predicting the impact of changes and generating documentation. They work best as an intelligent layer on top of existing tools, improving efficiency and accuracy.

Table A.1: TMMi level 2 process areas and AI support

TMMi Level 3 Process Area	AI/GenAI support and application
3.1 Test Organizations	**G4 Determine, Plan and Implement Test Process Improvements** GenAI can propose test process improvements that address common issues across projects and teams based on an analysis of test assessment reports, test completion reports and retrospective data.
3.2 Test Training Program	**G1 Establish Test Training Capability and Organizational Test Training Plan** By analysing job descriptions, test career paths, personal development plans, deployment plans, and historical training records, GenAI can produce a draft organizational test training plan for the upcoming period.
3.3 Test Process and Integration	**G3 Deploy the Organizational Test Process and Test Process Assets** GenAI can support the deployment of test documentation per organizational test process. For example, a GenAI script can generate draft test completion reports, highlighting successes and lessons learned.
3.4 Non-Functional Testing[1]	**Performance testing** AI can analyze real user behavior to design more realistic performance and load test scenarios. It can create and update performance scripts based on application changes thereby reducing manual scripting efforts. AI can also analyze performance metrics in real-time, detecting unusual behaviors and identifying bottlenecks. **Usability testing** AI models can be applied to assess the quality of user interfaces using approaches such as heuristics and supervised learning. Tools built on these models can detect incorrectly rendered elements, identify objects that are inaccessible or difficult to perceive, and uncover a range of other user-interface issues. AI-based computer vision can be used to compare images (e.g., screenshots) to identify unintended changes to the layout, the size, position, color, font or other visible object attributes. The results of these comparisons can be used to check that changes to the test object have not adversely affected the user interface. **Security Testing** AI in security testing can automate vulnerability detection, enhance threat modeling, and apply predictive analysis to identify risks faster and more accurately. It is used to address risks such as prompt injection, data poisoning, and unauthorized access. Key approaches include using AI for automated penetration testing and analyzing code patterns for vulnerabilities.

[1] For the process area 3.4 Non-Functional Testing, the AI/GenAI support options are listed by non-functional quality characteristic, rather than by goal. This is done to facilitate an clearer overview and better understanding of AI for non-functional testing.

TMMi Level 3 Process Area	AI/GenAI support and application
3.5 Peer Reviews	**G2 Perform Peer Reviews** With GenAI, all types of documents and software code can be reviewed for verification. Especially when source documents, rules, standards, checklists and previous review defects lists are provided, GenAI can deliver a high-quality review outcome.

Table A.2: TMMi level 3 process areas and AI support

TMMi Level 4 Process Area	AI/GenAI support and application
4.1 Test Measurement	**G1 Define Test Measurement Objectives** GenAI can provide guidance on selecting appropriate test measures aligned with test measurement objectives and can supply the operational definitions for those test measures. Subsequently, using scripts, it can support data gathering and calculate the test measures. **G2 Provide Test Measurement Results** AI can summarize and partly analyze test measurement data, evaluate them against objectives, and create reports and charts.
4.2 Product Quality Evaluation	**G1 Establish Quantitative Project Goals for Product Quality** GenAI can provide guidance on selecting appropriate product quality measures aligned with projects' product quality goals. It can also provide operational definitions for the selected product quality measures. Subsequently, using scripts, it can support data gathering and calculate the product quality measures. **G2 Manage Progress towards Achieving the Project's Product Quality Goals** AI can summarize and partly analyze product quality data, and create reports and charts. Comparing product quality data to product quality goals and criteria, it can trigger alerts for control actions. AI can support real-time insight into product quality status. By integrating AI capabilities across functional and non-functional testing and measurement, it can to provide an overall view of product quality, enabling a holistic assessment of product quality.

Table A.3: TMMi level 4 process areas and AI support

TMMi Level 5 Process Area	AI/GenAI support and application
5.1 Defect Prevention	**G1 Determine Root and Common Causes of Defects** AI can be used to categorize and group reported defects by analyzing text within defect reports and extracting topics, such as the area of affected functionality. AI models trained on the most critical defects can help identify those most likely to cause system failures, so they can be prioritized for causal analysis. In addition, AI can analyze large numbers of defects to identify patterns and recurring root causes.

TMMi Level 5 Process Area	AI/GenAI support and application
5.1 Defect Prevention (continued)	**G2 Define Actions to Systematically Eliminate Root Causes of Defects** GenAI can support this goal by automatically clustering common root causes (e.g., by lifecycle phase or technology) and recommending tailored solution types such as process changes, training needs, or test strategy improvements. It can also suggest solutions based on established best practices in the field.
5.2 Quality Control	**G1 Establish a Statistically Controlled Process** GenAI can define meaningful performance objectives and measures based on historical project data and industry benchmarks, and generate baselines from past test results. It can continuously analyze test execution data to detect patterns, trends, and sources of variation. Additionally, GenAI can enable real-time monitoring through dashboards and narrative summaries, comparing current performance against objectives and highlighting risks and deviations. **G2 Perform Testing Using Statistical Techniques** Based on production and usage data, GenAI can derive realistic operational profiles that reflect how systems are actually used. It can generate statistically representative and risk-based test cases aligned with these profiles, ensuring efficient coverage of high-probability and high-impact scenarios. During execution, GenAI can analyze test results in real time, apply statistical models to assess reliability and defect detection rates, and provide recommendations on stop-test decisions. Note, AI generated operational profiles required human validation against real-life production usage before being used for decision-making.
5.3 Test Process Optimization	**G1 Select Test Process Improvements** AI can support predictive optimization by identifying future test improvement opportunities and recommending proactive actions before issues arise. **G3 Implement and Deploy Test Improvements** GenAI can assist in analyzing the effects of test improvements. Based on this analysis, it can support decisions on whether to continue, adapt or enhance test improvement initiatives.

Table A.4: TMMi level 5 process areas and AI support

Annex B Quality Engineering Mapping

Quality engineering consists of a great number of related activities. These activities can be grouped in organizing and performing topics [Marcelis]. The organizing topics cover the activities at organizational level and management of quality assurance and quality control, e.g., testing. The performing topics cover the execution of quality assurance and quality control activities in a project or team.

Hereafter, a mapping is provided from the quality engineering topics to the TMMi process areas and improvement goals. As a conclusion, TMMi largely to fully covers all quality engineering topics. Only the topics tools and test automation are not covered by the TMMi. Both topics are part of the technology dimension within the People, Process and Technology (PPT) framework (see paragraph 1.4.3) and are, as such, not intended to be covered by the TMMi. TMMi of course, being a *process* model.

Quality engineering organizing topic	**TMMi process area**
Quality and Test Policy	2.1 Test Policy and Strategy
Responsibilities and Roles	3.1 Test Organization; as part of G2 job descriptions with responsibilities are developed for the various roles. 3.2 Test Training Program; ensures training for the required knowledge skills.
Monitoring and Control	2.3 Test Monitoring and Control
Anomaly Management	2.4 Test Design and Execution; G4 covers incident management.
Reporting and Alerting	2.3 Test Monitoring and Control; reporting and communication to stakeholders is part of monitoring and control.
Estimating	2.2 Test Planning; G3 covers test estimation.
Planning	2.2 Test Planning; G4 covers overall test planning and establishing a test plan. 2.4 Test Design and Execution (TDE); G2 covers defining a detailed schedule for test execution.
Infrastructure	2.5 Test Environment
Tools	Tools are part of the technology dimension with the PPT improvement framework and as such are not covered by the TMMi model. 3.2 Test Training Program; covers training for acquiring the skills to use and apply the tools.
Metrics	4.1 Test Measurement
Continuous Improvement	3.1 Test Organization; G4 covers identifying improvement opportunities based on the assessments and lessons learned in practice, and implementing them. 5.3 Test Process Optimization; continuous improvement is the central theme of TMMi level 5 and covered by Test Process Optimization.

Table B.1: Quality engineering organizing topics and TMMi process areas

Quality engineering performing topic	TMMi process area
Quality Risk Analysis & Test Strategy	2.1 Test Policy and Strategy; G2 covers a generic product risk analysis en test strategy at organizational level. 2.2 Test Planning; G2 covers product risk analysis and G3 defining a test approach (both at project or team level)
Acceptance Criteria	2.2 Test Planning; exit criteria (also called definition-of-done) are defined as part of the test approach. 2.3 Test Monitoring and Control; exit criteria monitoring is covered in G2.
Quality Measures	3.3 Test Process and Integration supports standardization (per G1) and shift-left of testing (per G2). In addition, 5.1 Defect Prevention and 5.2 Quality Control are process areas that aim to build quality from the start. These are examples of preventive quality measures that are part of TMMi. 2.4 Test Design and Execution, 3.4 Peer Reviews and 3.5 Non-Functional Testing jointly cover the detective quality measures 2.4 Test Design and Execution covers the investigation of incidents, and managing incidents to closure. Corrective quality measures such as refactoring of code and fixed problems performed by software engineers are not addressed within TMMi.
Reviewing	3.5 Peer Reviews
Test Design	2.4 Test Design and Execution
Test Data Management	2.5 Test Environment; G3 covers test data management.
Test Automation	Test Automation is part of the technology dimension with the PPT improvement framework and as such it is not covered by the TMMi model. 3.2 Test Training Program; covers training for acquiring the correct to skills to use and apply the tools.
Test Execution	2.4 Test Design and Execution
Investigate and Assess Outcome	2.4 Test Design and Execution; G3 covers comparing actual results with expected results and reporting test incidents, G4 deals with further investigation of the incident and managing it to closure

Table B.2: Quality engineering performing topics and TMMi process areas

Annex C The Improvement Process

C.1 Introduction

Primarily, TMMi is a list of best practices and, at various levels, a reference model that can be used for test process improvement. TMMi does not provide an approach to a change management and improvement process in an organization. To support the implementation of models such as CMMI, the Software Engineering Institute (SEI) has developed a change management process: IDEAL [IDEAL]. This model has also been proven to be very useful when implementing TMMi. IDEAL offers an extensive and practical reference standard for process change, including what needs to be done when implementing TMMi improvements. The IDEAL model is a five-phase improvement cycle as shown below in table C.1:

Acronym	Phase	Goal
I	Initiating	Establishing the initial improvement infrastructure for a successful improvement process.
D	Diagnosing	Determining the organization's current state as opposed to what it wants to achieve.
E	Establishing	Planning and specifying how the chosen situation will be established.
A	Acting	Executing the plan.
L	Learning	Learning by experience and improving the abilities to implement changes.

Table C.1: IDEAL five phase improvement cycle

Organizations are free to choose any improvement approach for the implementation of TMMi. In addition to IDEAL, there are several other models available. In general, all improvement implementation models are based on the Edward Deming Plan-Do-Check-Act cycle. The Deming cycle starts with making a plan that determines the improvement goals and how they will be achieved (Plan). The improvements are then implemented (Do) and it is determined whether the planned benefits have been achieved (Check). Based on the results of this evaluation, further actions are taken as needed (Act).

Throughout the change and improvement process, communication with stakeholders is of utmost importance. Communication is of the main pillars of the change management because it helps ensure that stakeholders understand the change, why it's happening, and how it will impact them. Effective communication reduces resistance, builds trust, and aligns everyone with the change goals. This annex provides an overview of the phases and activities of the IDEAL improvement process.

C.2 The Improvement Process

The phases and main activities of an improvement program in accordance with IDEAL are shown in Figure C.1 and described thereafter.

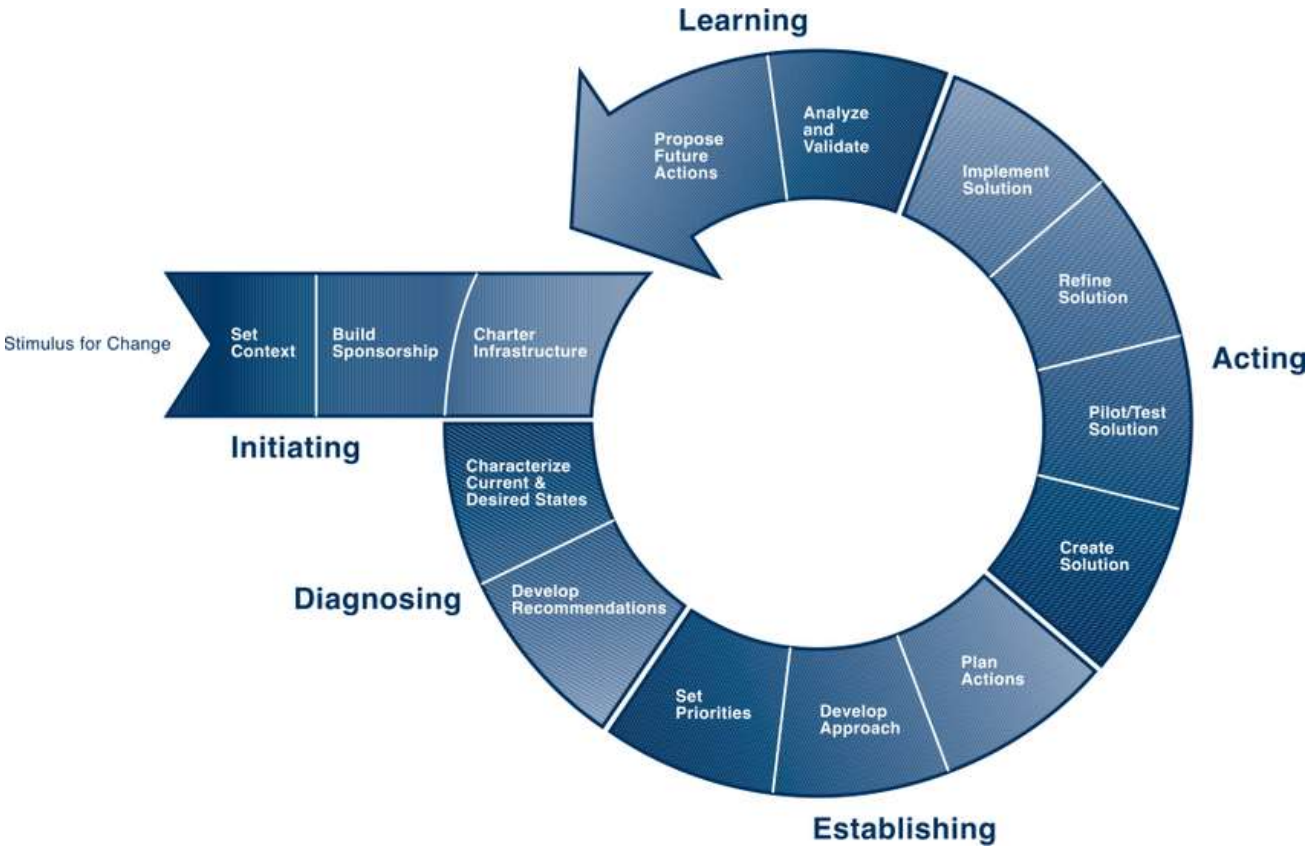

Figure C.1: Phases of an improvement program in accordance with IDEAL [IDEAL]

C.2.1 I – Initiating: The Initiating Phase

In the Initiating phase the infrastructure for a successful change management and improvement process is established. The goals and expected results with respect to the change and what needs to be contributed by the different parties concerned are defined. The goals of a TMMi implementation need to be in line with the business and quality goals of the organization. In this phase the goals cannot always be formulated in a SMART (Specific, Measurable, Achievable, Realistic and Time-based) way, which is why the goals are specified in more detail at the Establishing phase. In this phase, management commitment requires explicit attention. The initiating phase consists of the following activities:

- Identify Stimulus for Change
- Set Context
- Build Sponsorship
- Charter Infrastructure.

Identify Stimulus for Change

Before the improvement and change process is started, the organization needs to realize that change is necessary. This can be stimulated, for example, by dissatisfaction with the results of current testing, unexpected incidents, a senior management initiative, a benchmark result, a TMMi assessment, customer demand, market trends, or information taken from an internal measurement repository. The proposed change

must contribute to the success of the organization and complement the existing quality and business goals. The extent to which the change is in accordance with the business goals largely determines the success rate of the change.

Set Context

Management needs to determine how the change effort fits the quality and business strategy. Which specific business goals will be realized or supported by the TMMi implementation? How will current projects be influenced by the improvement? What proceeds need to be yielded, for example in terms of fewer defects and incidents, or shortening the lead time for test execution? During the improvement project, the context and effects will become more concrete, but it is important to be as clear as possible early in the project.

Build Sponsorship

Gaining support from the responsible managers and building sponsorship is extremely important in improvement projects. This concerns IT management and business management sponsorship, because these stakeholders will be influenced by the change. Sponsorship is important during the entire project, but because of the insecurity caused by changes, active support at the beginning of the project is particularly important. Supporting the improvement program is an important part of sponsorship, but sponsorship also includes providing active participation or promoting the project when there is resistance.

Charter Infrastructure

As a final activity in the Initiating phase, the way in which a change project is to be executed is determined, with an infrastructure put in place for this. The infrastructure must be described, including responsibilities and qualifications. Usually, the infrastructure will consist of a project board guiding several improvement teams. The project board consists of the sponsor, possibly sponsors of other improvement projects and the manager of the improvement program. There may also be an (external) TMMi expert. The project board is ultimately responsible for the improvement program and agrees on plans, milestones and results to be achieved. The board has the power to decide and is the highest escalation level.

C.2.2 D – Diagnosing: The Diagnosing Phase

In the Diagnosing phase it is determined where the organization stands in contrast to what it wants to achieve. To do this, assessments are conducted in which compliance with the reference framework, e.g., TMMi, is determined. The current state of the organization is determined and its desired future state is clearly formulated. The Diagnosing phase consists of the following activities:

- Characterize Current and Desired States
- Develop Recommendations

Characterize Current and Desired States

TMMi can be used to define the current and desired state of the testing processes. An assessment, either formal or informal, is conducted to determine the current state. The assessment uses the goals and practices as a checklist to determine the level of compliance with the process areas. The desired state shall be aligned with the stimulus for change as determined in the Initiating phase and it must be achievable for the organization.

Develop Recommendations

The recommendations suggest a way of proceeding with the proposed improvement activities. Which TMMi process area will be implemented first? Which part of a process area will be addressed and in what way? The recommendations are formulated under the guidance of internal or external TMMi experts in the specific process area.

C.2.3 E – Establishing: The Establishing Phase

During this phase a detailed TMMi implementation plan is developed to implement the recommendations. The improvement goals, defined in the Initiating phase, are elaborated on and detailed. The recommendations are prioritized taking into account factors such as available resources, visibility, likely resistance and contribution to business goals. The Establishing phase consists of the following activities:

- Set Priorities
- Develop Approach
- Plan Actions.

Set Priorities

The first activity of this phase is to set priorities for the recommendations and improvement actions to be executed. When the priorities are defined, it is determined which process area(s) and which improvement goals are to be implemented first.

Develop Approach

Using the recommendations and priorities, an approach is developed for achieving the desired situation, e.g., a TMMi level, and the resources needed to achieve this situation are estimated. The approach will typically address training, coaching, developing process descriptions and possibly tool selection. It will also take into account and address factors such knowledge and experience, expected resistance, sense of urgency, and the organization's culture.

Plan Actions

With the approach defined, the corresponding detailed actions can be determined. Together with information from prior activities, these are combined into a plan. The plan will include activities, the schedule, milestones, decision points, resources, responsibilities, measurements, monitoring mechanisms, project risks and the implementation strategy.

C.2.4 A – Acting: The Acting Phase

This phase is about concrete activities; this is where the action is! The plan will now be executed. The recommendations are specified in detail and will be implemented. Obviously, this phase consumes most of the effort, while developing the solution takes up about 30% of total effort, implementing the solution takes up to about 70% [Veenendaal and Cannegieter]. The Acting phase consists of the following activities:

- Create Solution
- Pilot/Test Solution
- Refine Solution
- Implement Solution.

Create Solution

The Acting phase begins with developing solutions to address the outlined issues and recommendations. The solutions should satisfy the purposes and practices of TMMi and contribute to achieving the desired situation. Solutions can include developing processes, templates, acquiring tools, acquiring knowledge and skills (training), providing information and support. The solutions are often developed by improvement teams, preferably involving the testers, and can also include a TMMi consultant.

Pilot/Test Solution

Following Tom Gilb's advice, "If you don't know what you are doing, don't do it on a large scale," the created solution should first needs be tested in one or more projects or teams. Sometimes only practical experience

can reveal the effect of a solution. In pilots, usually one or more test projects or teams are appointed in which the improvements are implemented and evaluated before additional projects adopt them.

Refine Solution

Based on the results of the pilot, the solution will be adapted and improved. Several iterations of the pilot process may be necessary to reach a satisfactory solution that will work for all projects or teams. A solution should be "good enough". Waiting for a perfect solution may unnecessarily delay the implementation.

Implement Solution

Once the solutions are ready, they can be implemented throughout the (test) organization, projects and teams. This is the activity that usually provokes the most resistance. As part of the implementation, training takes place. Training is a critical part of the implementation as it prepares people to adapt to new processes, tools, or behaviors required. Several implementation approaches can be applied, such as:

- Big bang - all the changes are implemented at the same time
- One project/team at a time - in each project/team the change is implemented at a set time
- Just in time - change are implemented when the process is executed within the project/team.

No single implementation approach is always better than another. The approach to be selected should be based on the nature of the improvement and the organizational context. For a major change, implementation may require substantial time, resources, effort and attention from management.

C.2.5 L – Learning: The Learning Phase

The Learning phase completes the improvement cycle. One of the goals of the IDEAL model is to continuously improve the ability to implement change. In the Learning phase, the entire IDEAL experience is reviewed to determine what was accomplished, whether the intended goals were achieved, and how the organization can implement future changes more effectively and efficiently. The Learning phase consists of the following activities:

- Analyze and Validate
- Propose Future Actions.

Analyze and Validate

This activity aims to answer several questions, such as:

- How did the improvement program go?
- What has been accomplished; have the initial goals been achieved?
- What worked well?
- What could have been done more efficiently or effectively?

Using these questions for guidance, lessons learned are collected, analyzed, summarized and documented.

Propose Future Actions

Based upon the previous activities, recommendations are formulated, which are intended to improve future improvement programs. These recommendations are provided to IT management and business management for consideration.

Annex D TMMi Level 1 Practices

Hereafter some testing practices are listed and briefly described that are typically performed by a TMMi level 1 organizations that are already “on their way”. This sample set of TMMi level 1 testing practices can also be used as an initial checklist by organizations that have yet to start on their test improvement journey. The practices are organized by their corresponding TMMi level 2 process area.

Test Policy and Strategy

- Senior management defines test objectives.
- Test levels are identified, but typically they are not thoroughly defined and aligned.
- Some level of responsibility for each of the identified test levels is assigned.

Test Planning

- Project management identifies what is important to test and sets the approach.
- High level test estimates are developed, typically a top-down estimate, to perform the testing work.
- The project plan will include allocated lead time and effort for the test execution phase.
- A list of test execution tasks is developed and people are assigned to these tasks.
- Project risks associated with the testing work are identified and recorded.
- Specifically in an Agile/DevOps context:
  - User story estimation is mainly focused on the development effort needed.
  - Definition-of-done criteria are defined, but typically high-level and not measurable.
  - The task board has a test execution column.

Test Monitoring and Control

- The completion of test tasks is recorded.
- Monitoring test progress is performed on the basis of tests executed and completed.
- Monitoring product quality is performed based on incidents found and their status.
- In non-Agile/DevOps context a list of incidents found, with their criticality and status, is available.
- Test progress or product quality issues are identified and managed to closure.

Test Design and Execution

- Test cases are developed, but typically without clear test objectives and not traceable to requirements.
- Tests are executed to verify whether requirements are implemented.
- Test execution is performed based on the availability of resources.
- Incidents found are recorded and resolved, but typically no template is used to document incidents.
- Test results are recorded and communicated.
- A user acceptance test is performed to validate the solution functions as intended.
- Regression testing is performed after changes, but typically not on a daily basis.

Test Environment

- A test environment exists for each system under test.
- The availability of test environments is shared across projects and teams.
- A representative set of production-like test data is typically missing.
- Test environment updates occur, but typically without a defined schedule and proper communication.

Implementation and Habit

- Senior management decides that test improvement is important.
- Version control is performed on the test work products.

Glossary

acceptance criteria	The criteria that a work product, component or system must satisfy in order to be accepted by the stakeholders.
acceptance testing	Testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.
actual result	The behavior produced/observed when a component or system is tested.
audit	An independent examination of a work product or process performed by a third party to assess whether it complies with specifications, standards, contractual agreements, or other criteria.
base measure	A measure that can be obtained by direct measurement.
best practice	A superior method or innovative practice that contributes to the improved performance of an organization within a given context, usually recognized as 'best' by other peer organizations.
black-box test technique	Technique to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure.
boundary value analysis	A black box test technique in which test cases are designed based on boundary values.
branch testing	A white box test technique in which test cases are designed to execute branches.
Capability Maturity Model Integration (CMMI)	An integrated model of best practices that enable business to improve performance by improving their processes. The CMMI provides a best-practice framework for building, improving and sustaining process capability. [CMMIV3]
checklist	Checklists are 'stored wisdom' aimed at helping to interpret the rules and explain their application. Checklists are used to increase effectiveness at finding major defects in a specification during a review. A checklist usually takes the form of a list of questions. [Gilb and Graham]
code coverage	An analysis method that determines which parts of the software have been executed (covered) by the test suite and which parts have not been executed, e.g., statement coverage or branch coverage.
common cause	The underlying source of a number of defects of a similar type, so that if the root cause is addressed the occurrence of these types of defects is decreased or removed.
component	A part of a system that can be tested in isolation.
component integration testing	Testing performed to expose defects in the interfaces and interaction between integrated components.
component testing	A test level that focuses on individual hardware or software components.
confidence level	The likelihood that the software is defect-free. [Burnstein]
configuration	The composition of a component or system as defined by the number, nature, and interconnections of its constituent parts.

configuration identification	A configuration management activity that involves selecting a product's configuration items, assigning them unique identifiers, and recording their functional and physical characteristics in technical documentation.
configuration item	Work products designated for configuration management and treated as a single entity in the configuration management process.
configuration management	The process of managing the integrity of work products using configuration identification, version control, change control, and audits.
confirmation testing	A type of change-related testing performed after fixing a defect to confirm that a failure caused by that defect does not re-occur.
continuous representation	A capability maturity model structure wherein capability levels provide a recommended order for approaching process improvement within specified process areas. [CMMI]
decision table testing	A black-box test technique in which test cases are designed to exercise the combinations of conditions and the resulting actions shown in a decision table.
defect	An imperfection or deficiency in a work product where it does not meet its requirements, specifications or impairs its intended use.
defect classification scheme	A set of categories, including phase, defect type, cause, severity, priority, to describe a defect in a consistent manner.
defect density	The number of defects identified in a component or system divided by the size of the component or system (expressed in standard measurement terms, e.g., lines-of-code or number of classes).
Definition-of-done	A set of criteria that must be met for the team to have a potentially releasable product. [ISTQB-ATLAS]
Definition-of-Ready	A set of criteria that must be met for the team to start production development work. [ISTQB-ATLAS]
Defect Detection Percentage (DDP)	The number of defects found by a test level, divided by the number found by that test level and any other means afterwards.
defect prevention	The activities involved in identifying defects or potential defects, analyzing these defects to find their root causes and preventing them from being introduced into future products. [After Burnstein]
defect report	Documentation of the occurrence, nature, and status of a defect. [ISO 29119-1]
defined process	The subset of organization process assets that are essential for any tailored or managed process. A fully defined process has enough detail that it can consistently performed by trained and skilled people and is both persistent and habitual. A defined process is required at TMMi level 3.
deliverable	Any (work) product that must be delivered to someone other than the (work) product's author.
Derived measure	A measure that is obtained from combining two or more base measure, e.g., a ratio.
driver	A software component or test tool that replaces a component that takes care of the control and/or the calling of a component or system.
dynamic testing	Testing that involves the execution of the software of a component or system.
entry criteria	The set of generic and specific conditions for permitting a process to go forward with a defined task, e.g., test phase.

equivalence partition	A subset of a value domain for which the behavior of a component or system is assumed to be the same, based on the specification.
equivalence partitioning	A black box test technique in which test cases are designed to execute representatives from equivalence partitions. In principle test cases are designed to cover each partition at least once.
exit criteria	The set of conditions for officially completing a defined task.
expected component	TMMi components that describe the activities that are important in achieving a required TMMi component, guiding those who implement improvements or perform appraisals.
expected result	The behavior predicted by the specification, or another source, of the component or system under specified conditions.
experienced-based test technique	Procedure to derive and/or select test cases based on the tester's experience, knowledge and intuition.
exploratory testing	An approach to testing whereby the testers dynamically design and execute tests based on their knowledge, exploration of the test item and the results of previous tests.
external measure	Measure of the degree to which a system or software product enables the behavior to satisfy stated and implied needs for the system including the software to be used under specified conditions
failure	An event in which a component or system does not perform a required function within specified limits.
feature	An attribute of a component or system specified or implied by requirements documentation (for example reliability, usability or design constraints).
functional testing	Testing performed to evaluate if a component or system satisfies functional requirements.
functionality	The degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. [ISO 25010]
goal	A required TMMi model component that describes the unique characteristics that must be present to satisfy the test process area.
heuristic evaluation	A usability test technique to determine the compliance of a user interface with recognized usability principles (the so-called "heuristics").
higher level management	The person or persons who provide the policy and overall guidance for the process, but do not to provide direct day-to-day monitoring and controlling of the process. Such persons belong to a level of management in the organization above the intermediate level responsible for the process and can be senior managers.
horizontal traceability	The tracing of requirements for a test level through the layers of test documentation (e.g., test plan, test design specification, test case specification and test procedure specification or test script). The horizontal traceability is expected to be bi-directional.
improvement proposal	A change request that addresses a proposed process or technology improvement and typically also includes a problem statement, a plan for implementing the improvement, and quantitative success criteria for evaluating actual results of the deployment.
incident	Any event occurring that requires investigation.
incident logging	Recording the details of any incident that occurred, e.g., during testing.

incident management	The process of recognizing, investigating, taking action and disposing of incidents. It includes logging incidents, classifying them and identifying the impact.
incident report	Documentation of the occurrence, nature, and status of an incident. [ISO 29119-1]
independence of testing	Separation of responsibilities, which encourages the accomplishment of objective testing.
informal review	A type of review that does not follow a defined process and has no formally documented output.
informative component	TMMi components that help model users understand TMMi required and expected components.
inspection	A formal review of a work product to identify issues, which uses defined roles and measurement to improve the inspection process [ISO 20246]
institutionalization	The ingrained way of doing business that an organization follows routinely as part of its corporate culture.
integration testing	A test level performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing.
internal measure	Measure of the degree to which a set of static attributes of a software product satisfy stated and implied needs for the software product to be used under specified conditions.
maintainability	The degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements. [ISO 25010]
managed process	A performed process that is recorded, followed, updated, and made persistent and habitual in use. A managed process is required at TMMi level 2.
master test plan	A test plan that is used to coordinate multiple test levels or test types.
maturity level	A rating which describes the degree to which processes in an organization unit meet the intent and goals of a predefined set of process areas.
Mean Time Between Failures (MTBF)	The average time between failures of a component or system.
Mean Time to Repair (MTTR)	The average time between failures of a component or system.
measure	Variable to which a value is assigned as the result of measurement. [ISO 25040]
measured process	A defined process whereby product quality and process attributes are consistently measured, and the measures are used to improve and make decisions regarding product quality and process-performance. A measured process is required at TMMi level 4.
measurement	Set of operations having the object of determining a value of a measure. [ISO 25040]
metric	A measurement scale and the method used for measurement.
Milestone	A point in time in a project at which defined (intermediate) deliverables and results should be ready.
non-functional testing	Testing the attributes of a component or system that do not relate to functionality, e.g., reliability, performance efficiency, usability, maintainability and portability.

non-functional test technique	Technique to derive and/or select test cases for non-functional testing based on an analysis of the specification of a component or system without reference to its internal structure.
operational profile	The representation of a distinct set of tasks performed by the component or system, possibly based on user behavior when interacting with the component or system, and their probabilities of occurrence. A task is logical rather than physical and can be executed over several machines or be executed in non-contiguous time segments.
optimizing process	A quantitatively managed process that is continuously improved to improve its capability. These continuous improvements can be made through both incremental and innovative improvements. An optimizing process is required at TMMi level 5.
peer review	The examination of work products performed by similarly skilled personnel during the development of work products to identify defects for removal.
performance efficiency	The degree to which a component of system uses time, resources and capacity when accomplishing it designated functions [ISO 25010]
planning poker	A consensus-based estimation technique, mostly used to estimate effort or relative size of user stories in Agile software development. It is a variation of the Wideband Delphi method using a deck of cards with values representing the units in which the team estimates.
portability	The degree to which a component or system can be transferred from one hardware, software or other operational or usage environment to another. [ISO 25010]
post condition	The expected state of a test item and its environment at the end of test execution.
practice	An expected TMMi model component that is considered important in achieving the associated goal. The practices describe the activities expected to result in achievement of the goals of a TMMi process area.
precondition	The required state of a test item and its environment prior to test execution.
priority	The level of (business) importance assigned to an item, e.g., defect.
product risk	A risk that a product may be defective in some specific aspect of its function, quality, or structure [ISO 29119-1]
project risk	A risk related to management and control of the (test) project, e.g., lack of staffing, strict deadlines, changing requirements, etc. [After ISO 29119-1]
quality assurance	Activities focused on providing confidence that quality requirements will be fulfilled.
quality characteristic	Category of software quality attributes that bears on software quality [ISO 25010]
quality control	Activities designed to evaluate the quality of a component or system.
quality engineering	The discipline of engineering concerned with product quality assurance and control whereby a systematic approach is applied to building quality into the product from the very beginning. It encompasses a set of principles and practices aimed at preventing defects rather than merely detecting them.
quantitatively managed process	A defined process evaluated and controlled using statistical and other quantitative techniques.
regression testing	Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the software or its environment is changed.

release note	A document identifying test items, their configuration, current status and other delivery information delivered by development to testing, and possibly other stakeholders, at the start of a test execution phase.
reliability	Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. [ISO 25010]
required component	TMMi components that are essential to achieving process improvement in a given process area, implemented across the organization.
requirement	A need or expectation that is stated, generally implied or obligatory.
requirements-based testing	An approach to testing in which test cases are designed based on and test conditions derived from requirements.
result	The consequence/outcome of the execution of a test. It includes outputs to screens, changes to data, reports, and communication messages sent out. See also actual result, expected result.
resumption criteria	The testing activities that may have to be repeated when testing is re-started after a suspension. [After ISO 29119-3]
review	A type of static testing in which a work product or process is evaluated by one or more individuals.
reviewer	The person involved in the review that identifies issues in the work product. A reviewer may be a subject matter expert, someone working on the project or any other stakeholder with an interest in the work product. [ISO 20246]
risk	A factor that could result in future negative consequences; usually expressed as impact and likelihood.
risk analysis	The process of assessing identified risks to estimate their impact and probability of occurrence (likelihood).
risk-based testing	A test approach in which the management, selection, prioritization, and use of testing activities and resources is consciously based on corresponding risk types and risk levels.
risk identification	The process of finding, recognizing and describing risks using techniques such as brainstorming, checklists and failure history.
risk level	The importance of a risk as defined by its characteristics, impact and likelihood. A risk level can be expressed either qualitatively (e.g., high, medium, low) or quantitatively.
risk management	Systematic application of procedures and practices to the tasks of identifying, analyzing, prioritizing, and mitigating risk.
risk mitigation	The process through which decisions are reached and protective measures are implemented for reducing or maintaining risks to specified levels.
risk type	A set of risks grouped by one or more common factors.
root cause	A source of a defect such that if it is removed, the occurrence of the defect type is decreased or removed.
root cause analysis	An analysis technique aimed at identifying the root causes of defects.
safety	The degree to which a component or system under defined conditions can avoid endangering human life, health, property, or the environment. [ISO 25010]

sampling	A statistical practice concerned with the selection of an unbiased or random subset of individual observations within a population of individuals intended to yield some knowledge about the population of concern as a whole.
security	The degree to which a component or system protects its data and resources against unauthorized access or use and secures unobstructed access and use for its legitimate users. [ISO 25010]
severity	The degree of impact that a defect has on the development or operation of a component or system.
shift-left	A test approach to perform testing and quality assurance activities as early as possible in the software development lifecycle.
smoke test	A test suite that covers the main functionality of a component or system to determine whether it works properly before planned testing begins.
software engineering	An engineering approach to software development that deals with the design, development, testing, and maintenance of software applications.
staged representation	A model structure wherein attaining the goals of a set of process areas establishes a maturity level; each level builds a foundation for subsequent levels. [CMMI]
standard	Formal possibly mandatory, set of requirements developed and used to prescribe consistent approaches to the way of working or to provide guidelines.
statement coverage	The percentage of the set of all executable statements of a test item that are covered by a test set. [ISO 29119-1]
static analysis	Analysis of software artifacts, e.g., requirements or code, carried out without execution of these software artifacts.
static testing	Testing in which a test item is examined against a set of quality or other criteria without code being executed, e.g., reviews and static analysis. [ISO 29119-1]
statistical process control	Statistical that identifies common and special causes of process variation and seeks to maintain process performance within limits. [CMMIV3]
statistical technique	Mathematical techniques used with the collection, analysis, interpretation, and presentation of masses of numerical data to understand process variation and predict process performance. [CMMIV3]
statistical testing	A test approach in which a model of the statistical distribution of the input is used to construct representative test cases.
stub	A skeletal or special-purpose implementation of a software component, used to develop or test a component that calls or is otherwise dependent on it. It replaces a called component.
subpractice	An informative TMMi model component that provides guidance for interpreting and implementing a practice.
suspension criteria	Criteria used to (temporarily) stop all or a portion of the testing activities. [ISO 29119-1]
system	A collection of components organized to accomplish a specific function or set of functions.
system integration testing	A test level that focuses on interactions between systems.
system testing	A test level that focuses on verifying that a system as a whole meets specified requirements.

Systems engineering	An engineering discipline whose responsibility is creating and executing an interdisciplinary process throughout a system's entire life cycle to ensure that customer and other stakeholder needs are satisfied.
T-shirt sizing	An estimation technique used in Agile software development to estimate the size or effort required to complete user stories or tasks. It involves assigning sizes based on T-shirt sizes, like XS, S, M, L, XL.
technical review	A type of formal review of a work product by a team of qualified personnel that examines the suitability of the work product for its intended use and identifies discrepancies from specifications and standards. [After ISO 20246]
telemetry	The automatic collection, transmission, and analysis of data from systems and applications to provide visibility into their performance, status, and behavior.
test	A set of one or more test cases.
test approach	The implementation of the test strategy for a specific project or team. It typically includes the decisions made that consider the (test) project's goal and the risk assessment carried out, starting points regarding the test process, the test design techniques to be applied, exit criteria and test types to be performed.
test basis	Body of knowledge used as the basis for the design of tests and test cases. [ISO 29119-1]
test case	Set of test case preconditions, inputs (including actions, where applicable), and expected results, developed to drive the execution of a test item to meet test objectives. [ISO 29119-1]
test case specification	A document specifying a set of test cases (objective, inputs, test actions, expected results, and execution preconditions) for a test item. [After ISO 29119-1]
test charter	A statement of test objectives, and possibly test ideas about how to test. Test charters are used in exploratory testing.
test completion	During the test completion phase of a test process data is collected from completed activities to consolidate experience, test ware, facts and numbers. The test completion phase consists of finalizing and archiving the test ware and evaluating the test process, including preparation of a test completion report.
test completion report	A document produced at the end of the test process summarizing all testing activities and results. It also contains an evaluation of the test process and lessons learned. [After ISO 29119-3]
test condition	Testable aspect of a component or system, such as a function, transaction, feature, quality characteristics, or structural element identified as a basis for testing. [ISO 29119-1]
test control	A test management task that deals with developing and applying a set of corrective actions to get a test project on track when monitoring shows a deviation from what was planned.
test data	Data created or selected to satisfy the input requirements for executing one or more test cases, which may be defined in the Test Plan, test case or test procedure. [ISO 29119-1]
test design	The process of transforming general testing objectives into tangible test conditions and test cases.
test design specification	Documentation specifying the features to be tested and their corresponding test conditions. [ISO 29119-1]

test design technique	Activities, concepts, processes, and patterns used to construct a test model that is used to identify test conditions for a test item, derive corresponding test coverage items, and subsequently derive or select test cases. [ISO 29119-1]
test environment	Facilities, hardware, software, firmware, procedures, and documentation intended for or used to perform testing of software. [ISO 29119-1]
test execution	The process of running a test on the component or system under test, producing actual result(s).
test execution phase	The period of time in a software development lifecycle during which the components of a software product are executed, and the software product is evaluated to determine whether or not requirements have been satisfied.
test execution schedule	A scheme for the execution of test suites within a test cycle. The test scripts or procedures are included in the test execution schedule in their context and in the order in which they are to be executed.
test estimation	An approximation estimate related to various aspects of testing (e.g., effort spent, completion date, costs involved, number of test cases, etc.).
test implementation	The activity that prepares the testware needed for test execution, e.g., developing and prioritizing test procedures, creating test data and writing automated test scripts,
test improvement plan	A plan for achieving organizational test process improvement objectives based on a thorough understanding of the current strengths and weaknesses of the organization's test processes and test process assets. A test improvement plan records the objectives, activities, resources, oversight, schedule and associated risks to improve test processes.
test item	The individual element to be tested. There usually is one test object and many test items.
test level	A group of test activities that are organized and managed together. A test level is linked to the responsibilities in a project. Examples of test levels are component test, integration test, system test and acceptance test.
test log	A chronological record of relevant details about the execution of tests. [After ISO 29119-3]
test logging	The process of recording information about tests executed into a test log.
test manager	The person responsible for project management of testing activities and resources, and evaluation of a test object.
test management	The planning, scheduling, estimating, monitoring, reporting, control and completion of test activities. [ISO 29119-1]
Test Maturity Model integration (TMMi)	A five-level staged framework for test process improvement describing the key elements of an effective and efficient test process.
test monitoring	The activity that checks the status of testing activities, identifies any variances from planned or expected, and reports status to stakeholders.
test object	The component or system to be tested.
test objective	A reason or purpose for designing and executing a test.
test performance indicator	A high-level metric used to monitor towards quantitative test improvement objectives. Collectively, test performance indicators provide a metric for determining business success.

test phase	A distinct set of test activities collected into a manageable phase of a project, e.g., the execution activities of a test level.
test plan	A detailed description of test objectives to be achieved and the means and schedule for achieving them, organized to coordinate testing activities for some test item or set of test items. [ISO 29119-1]
test planning	The activity of establishing or updating a test plan.
test policy	A high-level document describing the principles, approach and major objectives of the organization regarding testing.
test procedure specification	A sequence of test cases in execution order and any associated actions that may be required to set up initial preconditions and any wrap-up activities after execution.
test process	The set of interrelated activities comprising of test planning, test monitoring and control, test analysis, test design, test implementation, test execution, and test completion.
test process area	A cluster of related test practices in an area that, when implemented collectively, satisfy a set of goals considered important for making test improvements in that area.
test process assessment	A disciplined evaluation of an organization's test processes against a reference model.
test process asset library	A collection of test process asset holdings that can be used by an organization or project.
test process capability	The recorded range of expected results that can be achieved by following a test process.
Test Process Group (TPG)	A person or team who hold a test process role and are responsible for developing, deploying, and updating the organization's test process assets.
test process improvement	Tasks and activities planned, performed and used to improve an organization's test process capability and test performance to achieve business objectives more effectively.
test process performance	A measure of results achieved by following a test process.
test process performance baseline	A record and description of historical test process performance resulting from following a defined process, which can include central tendency, e.g., mean, median, mode, variation, and reflects how the test process is being performed.
test process performance objectives	Objectives and requirements for product quality, service quality and test process performance.
test progress report	A type of test report produced at regular intervals about the progress of test activities against a baseline, risks, and alternatives requiring a decision.
test run	Execution of a test suite on a specific version of the test object.
test schedule	A list of activities, tasks or events of the test process, identifying their intended start and finish dates and/or times, and interdependencies.
test script	Sequence of instructions for the execution of an automated test.
test session	An uninterrupted period of time spent in executing tests.
test specification	The complete documentation of the test design, test cases and test procedures for a specific test item. [ISO 29119-1]

test strategy	A high-level description of the test levels to be performed and the testing within those levels for an organization or program (one or more projects).
test suite	A set of test scripts or test procedures to be executed in a specific test run.
test summary report	A type of test report produced at completion milestones that provides an evaluation of the corresponding test items against exit criteria.
test type	A group of test activities aimed at testing a component or system focused on a specific test objective or quality characteristic, i.e. functional test, security test, usability test, regression test, etc. [After ISO 29119-1]
testability	The degree to which test conditions can be established for a component or system, and tests can be performed to determine whether those test conditions have been met.
testability review	A detailed check of the test basis to determine whether the test basis is at an adequate quality level to act as an input document for the test process.
tester	A person who performs testing.
testing	The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.
testware	Artifacts produced during the test process required to plan, design, and execute tests, such as documentation, scripts, inputs, expected results, set-up and clear-up procedures, files, databases, environment, and any additional software or utilities used in testing. [After ISO 29119-1]
three-point estimate	An estimation technique whereby a task is estimated considering three different estimates: the most optimistic, the most pessimistic and the most likely one.
usability	Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. [ISO 25010]
V-model	A sequential development lifecycle model describing a one-for-one relationship between major phases of software development from business requirements specification to delivery, and corresponding test levels from acceptance testing to component testing.
validation	Confirmation by examination that a work product matches a stakeholder's needs.
verification	The process of confirming that a work product fulfills its specification.
walkthrough	A type of review in which an author leads members of the review through a work product and the members ask questions and make comments about possible issues. [ISO 20246]
white-box test technique	Technique to derive and/or select test cases based on an analysis of the internal structure of a component or system.
Wide Band Delphi	An expert based test estimation technique that aims at making an accurate estimation using the collective wisdom of the team members.

References

[Baah] A. Baah (2017), Agile quality assurance, Bookbaby

[Black] R. Black (2002), Chapter 17, The Bug Reporting Process, in: E van Veenendaal (2002), *The Testing Practitioner*, *2nd edition*, UTN Publishing

[Boehm] B.W. Boehm (1981), *Software Engineering Economics*, Prentice Hall

[Burnstein]. I. Burnstein (2002), *Practical Software Testing*, Springer Professional Computing

[Cohn] M. Cohn (2009), Succeeding with Agile: Software Development using Scrum, Addison-Wesley

[CMMI] M.B. Chrissis, M. Konrad and S. Shrum (2007), *CMMI Second Edition, Guidelines for Process Integration and Product Improvement*, Addison Wesley

[CMMIV3] ISACA, *CMMI Model V3.0* (2024), Information Systems Audit and Control Association

[Crispin] L. Crispin and J Gregory (2008), *Agile Testing: A Practical Guide for Testers and Agile Teams*, Pearson Education

[Gilb and Graham] T. Gilb and D. Graham (1993), *Software Inspection*, Addison-Wesley

[Hollenbach and Frakes] C. Hollenbach and W. Frakes (1996), Software process re-use in an industrial setting, in: *Proceedings Fourth International Conference on Software-Reuse,* Orlando, April 1998, pp. 22-30

[IDEAL] SEI (1997), *IDEAL: A Users Guide for Software Process Improvement,* Software Engineering Institute

[Garousi] Vahid Garousi, Michael Felderer and Tuna Hacaloglu (2018), What We Know about Software Test Maturity and Test Process Improvement, in: *IEEE Software*, January/February 2018

[ISO 20246] ISO/IEC 20246 (2017), *Software and systems engineering – Work product reviews,* International Organization of Standardization

[ISO 25010] ISO/IEC 25010 (2023), *Software and systems engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model*, International Organization of Standardization

[ISO 25023] ISO/IEC 25023 (2016), *Software and systems engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality*, International Organization of Standardization

[ISO 25040] ISO/IEC 25040 (2024), *Software and systems engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality evaluation framework*, International Organization of Standardization

[ISO 29119-1] ISO/IEC 29119-1 (2022), *Software and systems engineering – Software testing – Part 1: Concepts and definitions*, International Organization of Standardization

[ISO 29119-2] ISO/IEC 29119-2 (2021), *Software and systems engineering – Software testing – Part 2: Test Processes,* International Organization of Standardization

[ISO 29119-3] ISO/IEC 29119-3 (2021), *Software and systems engineering – Software testing – Part 3: Test documentation*, International Organization of Standardization

[ISO 33002] ISO/IEC 33002 (2015), Information Technology – Process Assessment – Requirements for Performing Process Assessments, International Organization

[ISTQB] ISTQB (2025), *Standard Glossary of Terms Used in Software Testing V4.6*, International Software Testing Qualifications Board

[ISTQB-ATLAS] ISTQB (2023), *Certified Tester - Agile Test Leadership at Scale (CT-ATLaS) Syllabus V2.0*, International Software Testing Qualifications Board

[ISTQB-AI] ISTQB (2021), *Certified Tester – AI Testing (CT-AI) Syllabus V1.0*, International Software Testing Qualifications Board

[ISTQB-GenAI] ISTQB (2025), *Certified Tester – Testing with Generative AI (CT-GenAI) Syllabus V1.0*, International Software Testing Qualifications Board

[ISTQB-ITP] ISTQB (2010), *Certified Tester Expert Level Syllabus – Improving The Testing Process, V1.02,* International Software Testing Qualifications Board

[Leavitt] H. J. Leavitt (1964). Applied Organization Change in Industry: Structural, Technological and Humanistic Approaches. In: J. G. March (Ed.), *Handbook of Organizations*, Routledge

[Marcelis] R. Marcelis, B. van Veenendaal, D. Geurts and W. Ruigrok (2022), *Quality for DevOps teams – 3rd edition*, Sogeti Nederland B.V.

[Musa] J. Musa (1998), *Software Reliability Engineering Testing*, McGraw-Hill Education

[Musa and Ackerman] J. Musa and A. Ackerman (1989), Quantifying software validation: when to stop testing, in: *IEEE Software,* Vol. 6, No. 3, May 1989

[Ruigrok] W. Ruigrok, J. Egelmeers and R. Marselis (2025), *Amplified Quality Engineering*, Sogeti Nederland BV

[Survey23] TMMi Foundation (2023), *TMMi world-wide user survey 2023 V1.2,* TMMi Foundation

[TAMAR] C. Bates and E. van Veenendaal (ed.'s) (2026), *TMMi Assessment Method Application Requirements R1.3*, TMMi Foundation

[TMMiAgile] E. van Veenendaal (ed.) (2020), *TMMi in the Agile world V1.4,* TMMi Foundation

[TMMiAI] E. van Veenendaal (ed.) (2026), *Testing AI-systems and TMMi V1.1,* TMMi Foundation

[TMMiDevOps] E. van Veenendaal (ed.) (2025), *TMMi in the DevOps world V1.1,* TMMi Foundation

[TMMiReleaseV2] TMMi Foundation (2026), Release notes TMMi Version 2, TMMi Foundation

[Trienekens and Van Veenendaal] J. Trienekens and E. van Veenendaal (1997), *Software Quality from a Business Perspective,* Kluwer Bedrijfsinformatie

[Veenendaal02] E. van Veenendaal (2002), *The Testing Practitioner – 2nd edition*, UTN Publishing

[Veenendaal16] E. van Veenendaal (2016), *TMMi and ISO/IEC 29119: Friends or Foes?*, TMMi Foundation

[Veenendaal22] E. van Veenendaal (2022), ISTQB, TMMi or Test Automation, Three Pillars for Success: The PPT Framework, in: *SQ Magazine,* Issue no. 12, pp. 11-14, September 2022

[Veenendaal24] E. van Veenendaal, R. Black and D. Graham (2024), *Foundations of Software Testing – ISTQB Certification - 5th edition*, Cengage

[Veenendaal25] E. van Veenendaal (2025), Test Design, its Value and the AI Solution, in: *Quality Matters,* Issue 20, September 2025

[Veenendaal and Cannegieter] E. van Veenendaal and J.J. Cannegieter (2011), *The little TMMi – Objectives-Driven Test Process Improvement*, UTN Publishing

[Wake] B. Wake (2003), INVEST in Good Stories, and SMART Tasks, xp123.com/articles/invest-in-good-stories-and-smart-tasks