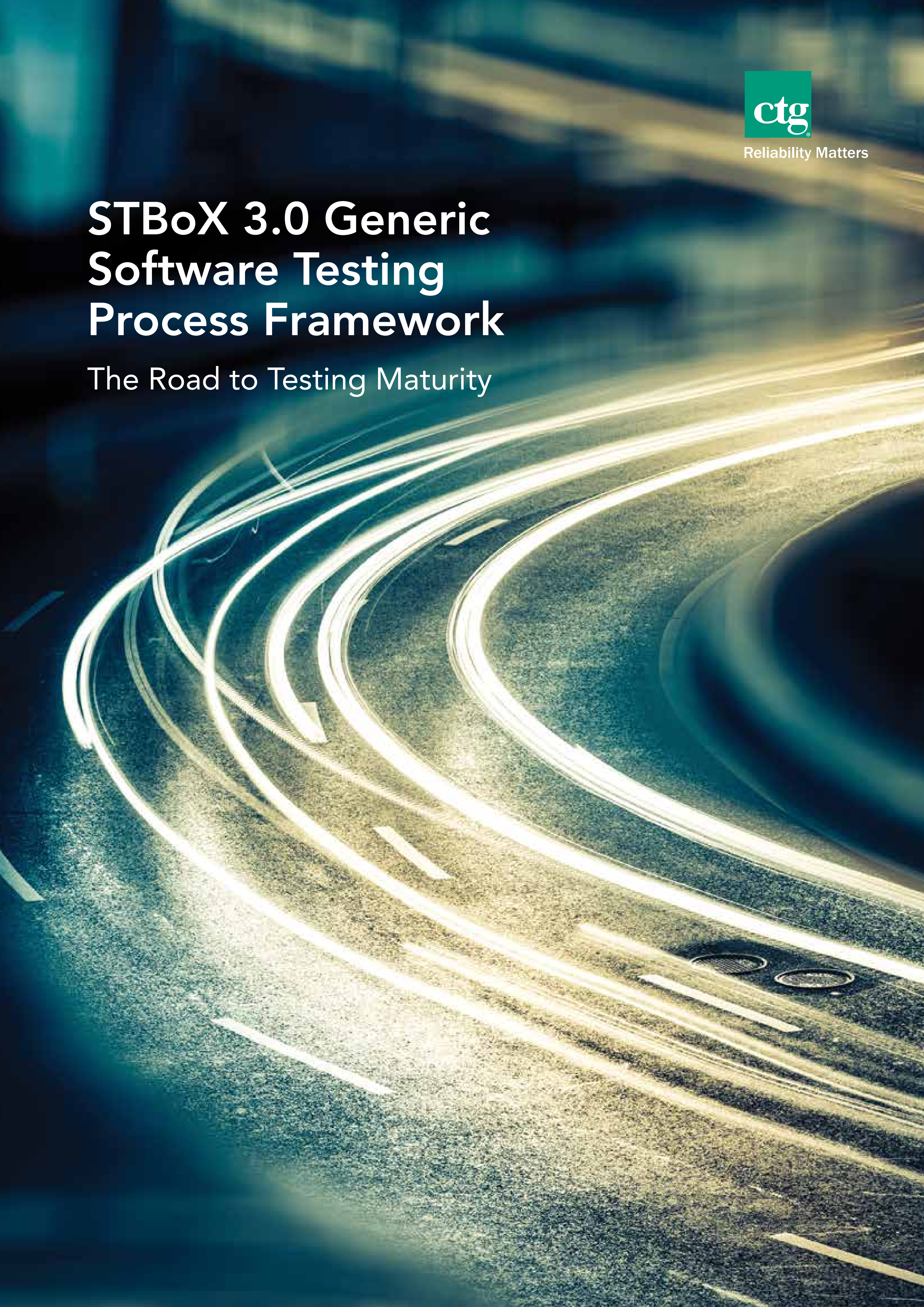
Reliability Matters

STBoX 3.0 Generic Software Testing Process Framework

The Road to Testing Maturity



DATA
DATA
Progress

STBoX 3.0 Generic Software Testing Process Framework

STBoX 3.0 provides a generic software testing process framework that offers testing governance practices based on CTG's extensive testing experience with many projects and clients.

Contents

CTG Belgium N.V. / S.A • Woluwelaan 140A • B-1831 Diegem, Belgium • +32 2 720 51 70
www.ctg.eu • http://jobs.ctg.eu • https://training-eu.ctg.com • www.stbox.eu

Preface

It was in 1998 that CTG, as one of the first IT services providers to do so, positioned software testing as a marketable service and began assisting many companies in many industries in setting up software testing practices. In 2006, the experience gained during those first years led to the launch of STBoX (Software Testing Based on CTG eXperience), CTG's own software testing methodology.

Today, I am proud to announce STBoX 3.0, the result of CTG continuously working to improve our testing services offering. STBoX 3.0 represents the evolution of STBoX from a methodology for software testing (released in 2006), to a software testing knowledge base for different environments (released in 2009), to this generic software testing process framework, applicable to many different situations and contexts. STBoX 3.0 proves CTG's leadership in the testing services market.

STBoX 3.0 offers project managers, test managers, and quality managers a generic software testing process that can be tailored to the specific context of the project they are working on. Through an ingenious system of add-ons, STBoX 3.0 offers tips and tricks on tuning testing practices for the project management methodology that is being applied, the specific test object being tested, or the specific test type at hand. The modularity of the add-ons supports easy maintenance of STBoX 3.0 to keep up with the newest developments in the software testing arena. Add-ons for Agile and mobile application testing are readily available, and add-ons for performance and security testing will be added soon.

Additionally, STBoX 3.0 offers policy makers a generic testing maturity model to measure and improve their company's testing processes as needed. Software testing has become a mainstream software development activity. It's no longer a question of introducing software testing, but a question of how software testing can be improved to face new challenges, which are ample. DevOps and shortened time to market, the internet of things and the multitude of mobile devices, and the ever-increasing importance of quality characteristics like security, performance, and usability are only a few examples of current software testing challenges that need to be conquered.

Finally, STBoX 3.0 also offers all test professionals a source of inspiration to get their testing started quickly. STBoX 3.0's clear and transparent organization will swiftly lead professionals to the right testing choices, no matter which software testing challenges they are facing.

Happy reading and happy testing!

[illegible]

Pieter Vanhaecke
Director, Testing Services
CTG Europe

Generic Software Testing Process

STBoX 3.0 offers a generic software testing process that allows project managers, test managers, and quality managers to plan, design, execute, and monitor testing in any project, tailored to the needs of that particular project.

"STBoX 3.0 offers a generic software testing process that allows project managers, test managers, and quality managers to plan, design, execute, and monitor testing in any project, tailored to the needs of that particular project."

The generic software testing process provides the following key features:

- Supports management of testing activities
- Supports testing by enforcing a common way of working to design tests, document test execution, and log defects
- Centralizes information in a central repository in a consistent way
- Prevents data loss
- Provides project intelligence by delivering clear metrics related to the test activities (test design, test execution) and the quality of the test object (e.g., defects raised)
- Facilitates the data flow between different stakeholders (developers, test engineers, project managers, etc.)

The framework is constructed around 15 core testing activities within six testing phases. These six phases are Test Project Preparation, Test Build, Test Execution, Test Project Closure, Test Management, and Quality Management. The phases group a number of testing activities together. They are related to each other in terms of their respective goals, their place in the software development life cycle, and the (testing) staff involved.

For each testing activity corresponding to the testing phases listed above, the framework offers a detailed description that consists of four sections: goals, steps, maturity, and add-ons.

Goals

In the goals section, a high-level definition of the respective testing activity will be given.

Maturity

Through the maturity section, it is made clear to what level of professionalism the respective testing activity can be executed. It will explain how the different steps of the respective testing activity need to be executed in order to achieve a certain maturity level.

Steps

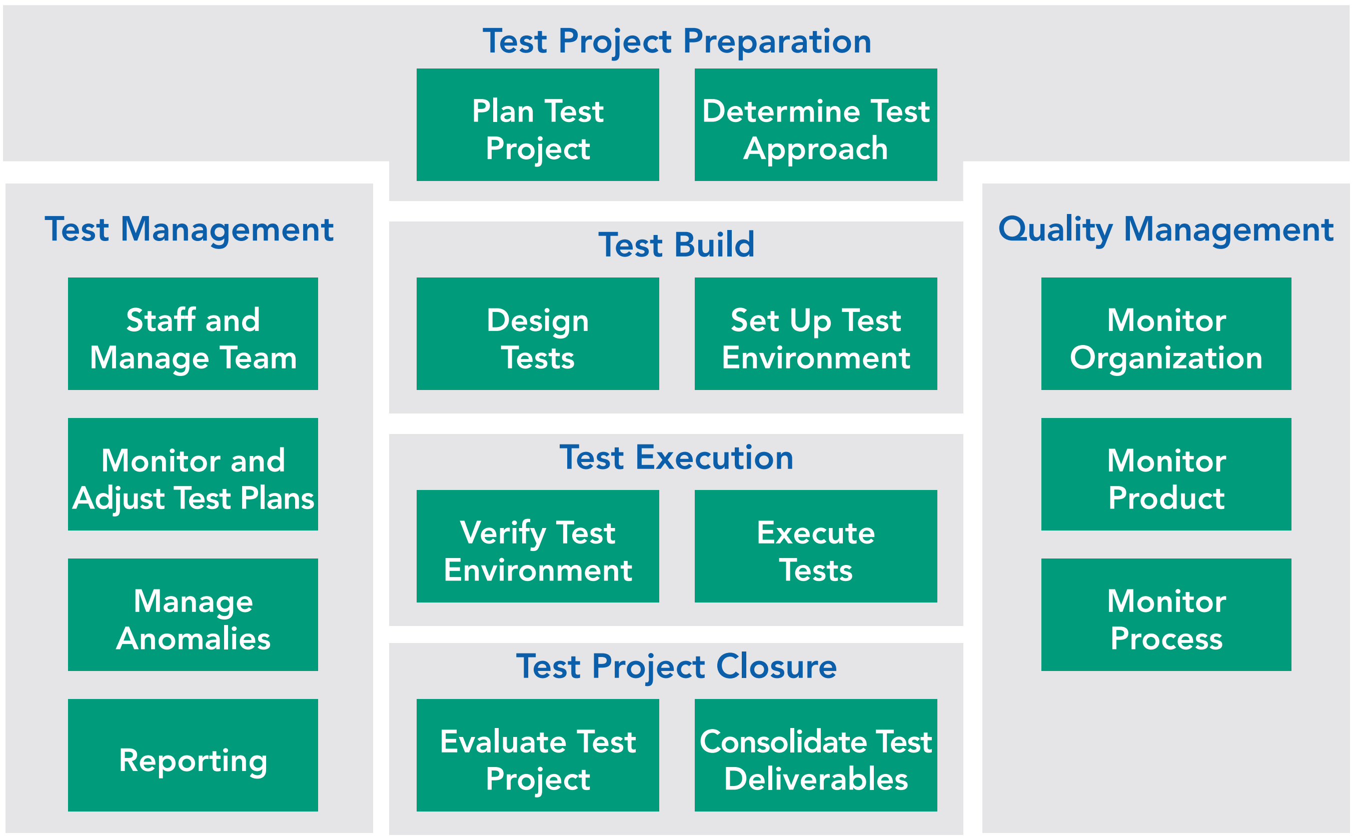
The steps section provides more detailed insight on what needs to be done during the respective testing activity. The complete series of testing steps needs to be executed in order to complete the respective testing activity.

Add-ons

In the add-ons section, relevant tips and tricks will be given on how to tailor the respective testing activity to a particular test object, testing type, or even software development life cycle. Add-ons are available for:

- Agile testing
- Mobile testing

Testing Phases



Test Project Preparation

A test project always starts with the Test Project Preparation phase in order to get a good overview of the project and begin to organize testing activities. All of the practical aspects (such as scope, approach, schedule, budget, and test environment) are discussed, decided upon, and compiled in the test plan. The objective is to:

- Determine the scope of the test assignment
- Develop a differentiated test approach
- Agree upon the overall test approach
- Agree upon test schedule and budget
- Foresee the required test environment

During this phase, the following testing activities will be executed:

- Plan test project
- Determine test approach

Test Build

The Test Build phase ensures that all testing activities that require completion before test execution can start and are conducted on time. The test model is delivered and the test environment is established. Delivering the test model includes:

- Complete features to test tree
- Create test procedures
- Create test cases, test scripts, and checklists
- Create test execution schedules

During this phase, the following testing activities will be executed:

- Design tests
- Set up test environment

Test Execution

Test Execution takes place after the tests have been built and the test environment is ready for use. All or part of the different test execution schedules that were prepared in the previous testing phase will now be executed according to the differentiated test approach developed during the Test Project Preparation phase. It includes:

- Verify readiness of the test environment
- Execute test execution schedules
- Report anomalies

During this phase, the following testing activities will be executed:

- Verify test environment
- Execute tests

Test Project Closure

When all other testing activities have been completed, the test project needs a clear and clean closure. This entails:

- Evaluate the test project that is about to be closed
- Determine lessons learned for future (test) projects
- Hand over all test deliverables to future owner
- Archive test deliverables for future reuse
- Archive test deliverables for the purpose of compliance with relevant regulation framework

During this phase, the following testing activities will be executed:

- Evaluate test project
- Consolidate test deliverables

Test Management

Test Management involves a set of continuous test project activities that are needed to ensure that the test project is managed professionally. Test Management includes:

- Manage the test team
- Follow up on the test project (planning, budget, scope, quality)
- Provide status report on all required topics and levels of detail
- Follow up on all detected anomalies until closure

During this phase, the following testing activities will be executed:

- Staff and manage test team
- Monitor and adjust test plans
- Manage anomalies
- Reporting

Quality Management

The purpose of Quality Management is to manage, guide, and improve the quality of the test project as a whole. This clearly exceeds the task of providing insight into the quality of the object which is the focus of the test project itself. Quality Management is about continuously monitoring and adjusting (improving) the quality of:

- The test process that is being applied in the test project
- The (intermediate) test project deliverables
- The testing organization

During this phase, the following testing activities will be executed:

- Monitor organization
- Monitor product
- Monitor process

Generic Testing Maturity Model

STBoX 3.0 also offers a generic testing maturity model that allows policy makers to measure and improve a company's testing processes according to their needs.

The generic testing maturity model provides the following key features:

- Supports measuring the actual testing performance of the company (as is)
- Supports comparing the actual testing performance of the company to industry standards
- Supports defining the targeted testing performance of the company (to be)
- Helps to design an improvement roadmap from "as is" to "to be"

The framework recognizes four different test maturity levels: initial, starter, experienced, and master. These levels provide a generic profile of the stages through which a company's testing activities evolve when gaining maturity in the way these activities are being executed. We say that, for a particular testing activity, a company (or project) is in a particular **test maturity area** when it is on its way to completing all of the conditions described in the corresponding generic maturity profile for that activity. When all conditions have been completed, we say the company (or project) is at a particular **test maturity level**.

Different test maturity levels for the respective testing activities allow testing projects or companies to focus on those areas that are most important to them. This way, the model provides a continuous test process improvement model.

"STBoX 3.0 also offers a generic testing maturity model that allows policy makers to measure and improve a company's testing processes according to their needs."

Test Maturity Levels

Initial

If being executed at all, testing is executed in an uncontrolled and even chaotic way. No formal techniques, nor test tools, are being applied. The quality of the testing activities and results exclusively rely on the quality of the individuals (heroes) executing these activities. There is no (structured) testing process available to support and align the different testing activities in the project. Stakeholders are hardly involved.

Experienced

Testing activities are being executed in a controlled and uniform way. A documented test process is available for all projects. Formal techniques are being used. Test tools are applied to support the documented test process. The quality of the testing activities now depends on the availability of repeatable test processes on a corporate level. All stakeholders play an active role in the test process.

Starter

Testing activities are being executed in a controlled way, but no formal techniques are being used. If applied at all, tools are being used on an ad hoc basis. The quality of the testing activities depends on local (to the project) organization of test processes. However, on a corporate level, there is no uniform test process defined nor documented. Principal stakeholders are consulted and informed about the major deliverables (test plan, test approach, test reports) of the test project.

Master

Testing activities are being executed in a controlled and uniform way. A corporate test policy and test strategy are available for all (test) projects. A documented test process implementing the corporate test policy and test strategy is available for all projects. Advanced formal techniques offering maximum control over test activities are being used. The use of test tools is totally integrated and documented with the test process. The quality of the testing activities is continuously monitored and improved upon. All stakeholders play an active role in the test process.

1. Test Project Preparation (TPP)

1.1 Plan Test Project

Goals

In the Plan Test Project phase, the test plan of the project is defined. The test plan answers questions regarding purpose, scope, organization, timing, budget, test environment, approach, and tools needed during the test project. The test plan is created preferably in parallel with the preparation of the general project plan. It is a living document which will be monitored and adjusted throughout the project life cycle. This activity involves the initial set up of the test plan; monitoring and adjusting are described in a separate activity.

Control over the testing process of **determining the test plan for the respective test project** that satisfies the business requirement for testing of **having a consensus and approval on what the goals of the test project are, how they will be achieved, what it will cost, and how the test activities are organized** by focusing on:

- The context of the project
- The definition of the test scope
- The approach that will be followed to test the defined scope
- The workload needed to obtain the test goals
- The setup of the test organization
- The timing in which the full test scope can be tackled
- The test environment needed to obtain the test results
- The way risks are handled during the project life cycle
- The consolidation of all activities performed during the Plan Test Project phase
- The communication of the test project plan

and is achieved by

- Studying the project context
- Determining the test scope
- Determining the test approach
- Setting up the test organization
- Creating a test schedule
- Determining the infrastructural needs
- Setting up risk management
- Creating a test plan
- Distributing the test plan

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

TPP 1.1.1	TPP 1.1.2	TPP 1.1.3	TPP 1.1.4	TPP 1.1.5
Study Project Context	Determine Test Scope	Determine Test Approach	Determine Workload	Set Up Test Organization
TPP 1.1.6	TPP 1.1.7	TPP 1.1.8	TPP 1.1.9	TPP 1.1.10
Determine Infrastructural Needs	Create Test Schedule	Set Up Risk Management	Create Test Plan	Distribute Test Plan

TPP 1.1.1 Study Project Context

In order to plan and organize all test activities, a solid understanding of why the project itself exists and what the expectations of the project are is crucial. In this step, the test manager makes sure he understands the project background and what the goals of the project are.

TPP 1.1.2 Determine Test Scope

The test manager defines what will be and what will not be tested. He, in other words, comes to a consensus on what the scope of the test project is. Defining the scope also clearly implies the definition of aspects which are out of scope for testing. A first indication of scope is made on test levels, test types, test items, and regression testing.

TPP 1.1.3 Determine Test Approach

Determining the test approach is clearly part of planning the test project. Based on the project's scope, now the test approach is defined. This test approach needs to be dedicated to the test project's particular needs. As many things need consideration when doing so, this step is described as a separate testing activity, "Determine Test Approach." The project's test approach is documented and maintained in the test plan.

TPP 1.1.4 Determine Workload

Meeting the test scope by using the defined test approach implies a list of test tasks that need to be accomplished. The test manager estimates what the workload of those tasks is in order to determine the workload of the test project.

TPP 1.1.5 Set Up Test Organization

The defined approach and estimated workload require test competences and resources. The test manager defines how the test team is assembled, who the team members are, and what their roles and responsibilities are.

TPP 1.1.6 Determine Infrastructural Needs

The test tasks that need to be performed require specific test environment needs. Those needs are defined by the test manager and can be on the level of hardware, network, middleware, system software, development software, testing software, application architecture, system management procedure, etc.

TPP 1.1.7 Create Test Schedule

The test manager needs to define the existing dependencies and constraints which will influence the test tasks. Dependencies can exist between the defined test tasks and/or between other projects. Constraints can be introduced due to timing, resource, and/or quality restrictions. The test schedule represents the timing and budget of the test tasks, taking into account all defined dependencies and constraints.

TPP 1.1.8 Set Up Risk Management

The risks that could possibly impact the to-be-achieved test scope are identified by the test manager. Mitigating measures are defined together with the risks, and risk monitoring is put in place.

TPP 1.1.9 Create Test Plan

The test manager consolidates all available information on the test project obtained in the previously described steps in a test plan. This is the initial version of the test plan which will be monitored and adapted during the "Monitor and Adjust Test Plan" activity.

TPP 1.1.10 Distribute Test Plan

The test manger distributes the initial version of the test plan.

Maturity

The testing activity **"Plan Test Project"** that satisfies the business requirement of **having a consensus and approval on what the goals of the test project are, how they will be achieved, what it will cost, and how the test activities are organized** is at:

Initial level

The "Plan Test Project" testing activity is executed in an uncontrolled and chaotic way.

- The steps described in this activity are organized for some projects, but not for all projects in the organization.
- If the activity is organized, it is not organized by using a common approach. Each project decides on its own how the test project planning is performed (chaotic and ad hoc).
- For some projects, a test plan is created and documented.
- The stakeholders are not involved in the test project planning.

Starter level

The "Plan Test Project" testing activity is executed for all projects in the organization, but not by using a common approach. The approach for planning the test project is defined by the project, not by the organization.

- A test plan is created for each project and documented.
- The test plan is not necessarily a separate document, it can be part of the project plan.
- The test plan has no fixed format. Instead, the project defines which format is used. There is no general template within the organization.
- The steps followed to perform this activity do not follow a uniform process.
- The test project manager or project manager decides on the content of the test plan.
- Stakeholders are not involved; they can be consulted during the creation of the test plan, but do not participate actively in the process. It is the test manager who makes the decisions.
- Upon completion, the test plan is only distributed to the principal stakeholders of the project.

Experienced level

The "Plan Test Project" testing activity is executed for all projects by making use of a common approach.

- A test plan is created for each project following a fixed format, which is applied to all projects.
- A uniform test plan template is available and used within the organization.
- The test basis is defined by making an inventory of all relevant information about the project. This enables the ability to keep track of which documents are available and which documents are not yet available.
- The following topics are clearly addressed in the test plan: test scope, test approach, test workload, test organization, test schedule, test environment, and risk management.
- A test plan is created with active involvement of the test manger and the principal stakeholders.
- Upon completion, the test plan is only distributed to the principal stakeholders of the project.
- A kickoff meeting with the principal stakeholders is organized to explain the content of the test plan.
- The approval of the test plan is implicit.
- Acceptance criteria and entry/exit criteria are defined.

Master level

The "Plan Test Project" testing activity is executed for all projects and documented in a test plan which follows the corporate test policy and test strategy guidelines.

- The following topics are clearly addressed in the test plan: test scope, test approach, test workload, test organization, test schedule, test environment, and risk management.
- Multiple test plans with different levels or types can be created depending on the scope, complexity, and/or criticality of the test projects.
- The test basis is defined and frozen by making an inventory of all relevant information about the project. This enables the ability to keep track of which documents are available and which documents are not yet available.
- A corporate test policy and strategy are available.
- All stakeholders fully participate in the creation of the test plan, depending on their responsibility.
- The test manager and all involved stakeholders cooperate to produce the test plan.
- Upon completion, the test plan is distributed to all stakeholders.
- A kickoff meeting with the principal stakeholders is organized to explain the content of the test plan. Separate kickoff meetings can be organized depending on the needs of the different stakeholders or the size of the project.
- Workshops are organized to define the content of the test plan.
- The approval of the test plan is implicit, but can be formally organized if needed.
- The test plan is treated as a living document during the project life cycle and is regularly updated if needed, and version management is applied.
- Acceptance criteria, entry and exit criteria, pass and fail criteria, and suspension and resumption criteria are defined.

Agile Testing Add-ons

Release Planning

In an Agile environment, the Test Project Preparation phase is an activity that is part of release and roadmap planning. The test manager and testers, together with all other team members, participate in the release and roadmap planning session in order to give input concerning complexity, dependencies, priorities, effort, test strategy, and the approach that need to be taken into account.

The intention of release planning is to understand the goal of the project and to identify the high-level features that need to be delivered, breaking features and epics into user stories and defining acceptance criteria. The product owner sets the release goal and release timeframe. The team performs high-level effort estimation, and based on this result and the velocity of the team, the release planning is performed. The length of iteration influences the frequency at which the developed features can be delivered to the customer. At the beginning of the project, the tester assists with selection of iteration length.

Test Scope

The test scope is based on defined features added to the product backlog and the prioritization made by the product owner. Product backlog prioritization is not static, but a recurrent process that constantly impacts the test scope. Business value, clarity, and dependencies are some factors that need to be considered when prioritizing the user stories. The role of the tester is to assist the product owner in defining the priorities and verifying whether features are testable.

Requirements are not yet known in detail at the beginning of the project. An Agile environment allows for changes to be made after the initial planning by taking the client's feedback into account. The customer has the opportunity to make changes and add, remove, or reprioritize features, resulting in a changed test scope.

Workload

In comparison to many Waterfall projects, in an Agile environment, the test effort is not separately estimated, but is part of the total effort of developing and delivering a user story or requirement. The estimation is provided by the whole team and is based on the complexity of the story and the test activities that have to be executed to meet definition of done.

Tasks that have to be taken into account during the estimation meeting are:

- Set up environment
- Design test cases
- Execute test cases
- Perform exploratory testing
- Perform test automation

There are several estimation techniques that are used to define the workload:

- Planning Poker
- T-Shirt Size
- Swimlane Sizing

Risk Management

Agile teams promote communication, which enables the team to respond quickly to change. This makes them better equipped to deal with risks than teams in more traditional development environments. It is a good practice to visualize and track associated risks and actions on a risk board. During the daily stand-up meeting, risks and impediments can be escalated and are picked up by the Scrum master to unblock the situation. Risks are also formally included in the agenda of the iteration planning meetings, iteration reviews, daily stand-up meetings, and retrospectives.

Test Plan

The Agile manifesto clearly favors working software over comprehensive documentation, and responding to change over following a plan. So, often only a lightweight document that centralizes the most crucial information is created as a test plan. All practical aspects such as scope, strategy, approach, schedule, budget, and test environment are discussed, decided upon, and compiled. According to James Bach, the plan is the intersection of strategy (the set of ideas that guide your test design) and logistics (the application of resources to fulfill the strategy).[1]

[1] Bach, James, and Michael Bolton. "Rapid Software Testing – Appendices." Published class lecture. Satisfice, Inc., 2015. http://www.satisfice.com/rst-appendices.pdf

Test Approach

There are several factors that play an important role in determining the test approach. It is crucial to understand that the goal of each sprint is to produce a potentially shippable product. A cross-functional team with the necessary skills to complete the sprint backlog, as well as explicit definitions of ready and done, is key to successfully delivering working software at the end of each sprint.

A definition of done is a set of criteria that describes what will be delivered at the end of each iteration, not only in terms of functionality, but in terms of quality as well. It is a kind of checklist and can be seen as exit criteria that help to create consensus and common understanding, which is crucial to a high-functioning Agile team. Defining and staying true to the definition of done is the responsibility of the whole team. The tester assists in defining the set of criteria. Agile development does not work if stories, iterations, or releases aren't "done." There are different levels of "doneness:"

- Task level
- User story level
- Iteration level
- Release level

Other aspects of the test approach are the Agile testing quadrants and automation pyramid, which help to define the approach for regression testing and the quality attributes to be considered in testing, and will be discussed in detail in the "Test Execution" activity section (3.2). Due to the incremental development, the risk of regression is increased in Agile projects. Therefore, it is important to manage this risk by defining the right test approach.

Test Team

In general, in an Agile environment, testers are part of a multi-functional team in which every team member should have the capabilities to prepare, execute, or automate tests. From an Agile point of view, the whole team has to contribute to testing activities. It is the responsibility of the entire team to ensure the expected quality of the software. The Agile test plan is a good place to document which test resources are part of the Agile teams, and which and when outside resources will be added to the project.

The allocation of the test resources must be done in close cooperation with other Agile teams, the project manager, and the test manager.

Infrastructural Needs

The benefit of iterative and incremental development is a short feedback loop. However, this requires continuous integration and deployment. It is self-evident that this implies the availability of tools.

Continuous integration is the practice in which integration happens early and often to avoid problems, and thus, reduce rework, cost, and time. The team frequently integrates new or changed code, often meaning that no intervening window remains between a code commit and a new build.

Normal practices using a scheduled build can cause developers to miss build and deploy slots, causing errors to arise without being noticed until after the deployment is completed.

The principle behind continuous integration is that a build can happen at any time or, put in another way, every commit can trigger a complete and functional build.

An automated build process that can be triggered at any time lies at the core of continuous integration. Many teams use a one-button-deploy practice that not only builds a new version of the software, but also does it very quickly so that the team doesn't have to wait to start new tasks until the build has completed. This goes for both coding and testing tasks. The practice that every commit should trigger a new build in order to verify that the changes integrate correctly can also be adhered to with an automated build.

Source control management and version control are key to continuous integration in the Agile development process. It's not only a matter of keeping track of all the code commits, but also making sure that all elements required to build the project are in place in the central repository so that the availability of additional (external) dependencies can never become an issue when building.

With the frequency of builds being deployed, it is obvious that the testers will never be able to keep the pace if they have to manually perform both regression and new test cases. Therefore, it is important to start automating sanity checks or regression tests at first.

This will not only strengthen the current build process, but will continue to give the testers more valuable time for testing new features and performing exploratory or session-based testing.

All of these activities rely heavily on the availability and maturity of tools. This includes automated tools, as it makes little sense to automate the build process, repository control, and testing if you need to put a large amount of manual labor into using the tools themselves.

The selection of tools should be made at the beginning of the project. It depends on the team and tool maturity, and also on the company policy for tools and vendors.

Mobile Application Testing Add-ons

Expectations

Mobile applications tend to be built for use under very specific conditions. Often, the target audience or the environment in which the app is to be used are very specific. Generally speaking, it seems that expectations of mobile applications are higher.

Therefore, testers should make an extra effort when studying the project context. It is crucial that testers fully understand the expectations of the mobile application under test. Asking the right questions about these expectations right from the start is an absolute must to be able to design an adequate test approach later on. When involved in an early stage, testers might even contribute to determining the exact expectations of the particular mobile application.

Policy Restrictions

When studying the project context, understanding relevant company policy regarding the use of mobile test devices is also very important. Such policies can have a big impact on which tests to plan. Examples of company policies that can restrict the testing activities you can deploy are:

- Authorization to leave the building for field testing
- Authorization to connect to the company's wireless network
- Authorization and budget to purchase mobile devices, SD cards, SIM cards, etc.
- Authorization to use a company car to perform road tests

Test Scope

Testers should reach beyond the boundaries of traditional thinking when determining the exact scope of a mobile application testing project. There are a number of features related to mobile applications that deserve testers' special attention, listed in the checklist below.

- **Network:** Think about how you want the app to work with Wi-Fi, 3G, 4G, switching between networks, switching between network strengths, etc. For instance, it is possible that Wi-Fi signals for a certain app are not as important, so you can decide to leave this out of scope.
- **Target audience:** Think of which type of users will use your app, which OS, which device, etc. Due to the large number of possible devices, it is impossible to test each possible configuration. To determine the test scope, you can put emphasis on the most frequently used devices of your target audience, while considering the full spectrum between low-end and high-end devices.
- **Movement and Location:** It is expected that the app will be used while moving (for instance, a running app or an app designed for use in vehicles), or will it be used in a static environment? Will the app be used inside or outside? If your application is used outdoors and it consumes a lot of energy, it might suffer from a battery drain, forcing its user to search for alternative solutions.
- **User Load and Time:** Is the expected user load spread more or less throughout the day? Or are there specific moments in the day when we can expect the user load clustering?
- **Interactions:** Will the application only be tested in isolation, or will its interaction with other applications be added to the scope? Think of the way the app will interact with other apps—how does your application handle interruptions (e.g., a phone call) and does your app continue to function while other existing

apps of your company are running? Be aware of the fact that simultaneously running applications might have an impact on each other. Consider testing your app while other frequently-used apps are running as well in the background.

- **Service Availability:** Think about the availability of external services and how this would impact the performance of the mobile application when the availability of these services is hindered (through network issues, server downtime, infrastructural changes, etc.). The tester can test this through extensive API testing, implementing stubs, and by using service virtualization and network virtualization to cover the various external dependencies of the mobile application.

Testing Staff

The competences needed from testing staff employed in mobile application testing projects also requires the test manager's extra attention. Generally speaking, it can be argued that a test manager might want to look for testers with a proper set of technical skills. To deal with the amplitude of possible devices and operating systems, tooling will inevitably be brought into the project to cover the most ground. Testers should be able to understand and operate the tools used.

The test manager can also consider bringing in end users from outside of the project team and organize some beta testing or hallway testing. Typically, beta testing will be organized with future end users of the mobile application under test. On the other hand, hallway usability testing will be executed by randomly-chosen people; people that happen to "pass by in the hallway" are asked to try using the mobile application under test. When feasible, organizing beta testing or hallway usability testing can have multiple advantages. Obviously, defects might be reported (and fixed), but maybe even more importantly, you can anticipate possible negative reviews on app stores. App stores give a lot of power to the end users; they can make or break the success of your app, so it is worth considering this at the very beginning of your project.

Infrastructural Needs

When performing mobile application testing, infrastructural needs will certainly be different, and most will probably be more demanding compared to those of more traditional testing.

There is a strong need for controlled test environments that cover the enormous range of conditions (device type, operating system, connection type, etc.) under which mobile applications can be used. However, one should be aware of the fact that it is simply impossible to create a test environment that covers it all. Solutions simulating some of these conditions and test automation tools that support recurring tests can be brought in, but it will never be possible to build an environment that fully covers the defined needs.

Instead of trying to build very complex and very expensive test environments that cover as many conditions as possible, mobile application testing projects can also consider testing in the real world to make up for the conditions that cannot be offered in a controlled test environment. In-the-wild testing, which involves testing in the real world, can be organized as a complement to the testing already planned and executed in the available test environments. This concept will be explained further in the next activity section, "Determine Test Approach."

1. Test Project Preparation (TPP)

1.2 Determine Test Approach

Goals

Determining the test approach is an iterative process that is an important aspect of creating the test plan. The test approach will be at the heart of your testing project. At first, a high-level approach is created that can be refined later as more details concerning the test project become available.

The result of this testing activity is the project's test approach matrix, which will be documented and maintained in the project's test plan.

Control over the testing process of **determining a test approach for the respective test project** that satisfies the business requirement for testing of **finding the most important defects in the object under test as soon as possible and at the lowest possible cost** by focusing on:

- Identifying the scope of the test project
- Identifying test requirements and/or product risks
- Composing the features to test tree (FTT)
- Determining test levels and test types
- Determining priorities
- Determining test depth
- Assigning test techniques
- Creating the test approach matrix

and is achieved by

- Identifying the object under test's features to test
- Prioritizing the object under test's features to test
- Determining the quality profile of the object under test
- Designing a differentiated test approach
- Distributing the differentiated test approach

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

TPP 1.2.1	TPP 1.2.2	TPP 1.2.3	TPP 1.2.4	TPP 1.2.5
Identify Features to Test	Prioritize Features to Test	Determine Quality Profile	Design Differentiated Test Approach	Distribute Test Approach

TPP 1.2.1 Identify Features to Test

An inventory of project requirements and product risks that need to be tested is created. Defined items to test are organized in an FTT.

TPP 1.2.2 Prioritize Features to Test

A relative priority is given to each feature to test that is listed in the FTT.

TPP 1.2.3 Determine Quality Profile

The quality profile is defined when a feature to test can be approved. The quality measures for the features to test are determined. Standards like ISO 25010, but also other commonly used heuristics, can be used as a source for this.

TPP 1.2.4 Design Differentiated Test Approach

Based on the priorities and the quality profile, it is decided when and how (extensively) the features to test will be tested. The test approach is documented in the test approach matrix, which summarizes the test project's features to test and their quality measures, applicable test levels, and test design techniques.

TPP 1.2.5 Distribute Test Approach

The differentiated test approach for the project (documented in the test approach matrix) is distributed to the involved stakeholders.

Maturity

The testing activity **"Determine Test Approach"** that satisfies the business requirement of **finding the most important defects in the object under test as soon as possible and at the lowest possible cost** is at:

Initial level

The "Determine Test Approach" testing activity is executed in an uncontrolled and chaotic way.

- A test approach is not defined for all projects.
- If a test approach is defined, it is done in an intuitive way. Business requirements and priorities for testing are determined based on experience and "gut feeling."

Starter level

The test manager takes control of the "Determine Test Approach" testing activity by creating a test approach matrix. A high-level risk analysis, mainly based on the defined test requirements, is executed.

- The test manager breaks down the business requirements into more manageable test requirements, or features to test, and lists them in an FTT.
- The test manager adds apparent product risks to the FTT. Rudimentary risk identification techniques (e.g., checklists, questionnaires, expert interviews) are applied for this.
- The test manager assigns a priority number to each feature to test by applying a qualitative appreciation of the impact and likelihood of the risk level associated with that particular feature to test.
- The test manager determines the quality profile of all features to test using a standard set of quality attributes. Relevant test types are determined and the FTT is organized accordingly.
- The test manager determines the different test levels at which the features to test will be tested and documents this in the test approach matrix. At this level, there is no real coordination between the different test levels.
- Selection of test design techniques to be used is left to the test analyst or tester at hand.
- The test manager distributes the test approach matrix to the principal stakeholders. This can be done as a part of or addendum to the project's test plan.

Experienced level

The test manager involves the principal stakeholders when executing the "Determine Test Approach" testing activity. More attention is given to product risks when creating the test approach matrix.

- Together with the principal stakeholders, the test manager breaks down the business requirements into more manageable test requirements, or features to test. All features to test are listed in an FTT.
- The test manager organizes facilitated workshops (e.g., risk workshops or risk brainstorming) with the principal stakeholders to identify product risks that are also being added to the FTT.
- In separate workshops, the test manager and principal stakeholders assign a priority number to each feature to test by applying a qualitative appreciation of the impact and likelihood of the risk level associated with that particular feature to test.
- Together with the principal stakeholders, the test manager determines the quality profile of all features to test using a standard set of quality attributes (and their relative importance level). Relevant test types are determined and the FTT is organized accordingly.

- Together with the principal stakeholders, the test manager determines the different test levels at which the features to test will be tested and documents this in the test approach matrix. Gaps and overlaps between the different test levels are discussed with the principal stakeholders.
- Taking into account the priority number of the different features to test and the importance level of the different test types (or quality attributes), the test manager determines the test depth needed and assigns test design techniques accordingly. The test design techniques to be applied are documented in the test approach matrix.
- The test manager distributes the test approach matrix to the principal stakeholders. Approval is implicit.

Master level

The test manager involves all stakeholders when executing the "Determine Test Approach" testing activity. There is a clear shift toward product risks as the basis for the test approach matrix. Detailed, cooperative risk identification techniques are applied to identify these product risks.

- Features to test are determined mainly based on identified product risks.
- The test manager applies cooperative risk identification techniques (e.g., PRISMA, FMEA) to identify product risks or features to test in a joint effort with all stakeholders involved. All features to test are listed in an FTT.
- For reasons of completeness, business requirements can be broken into more manageable test requirements, which will be added to the FTT.
- In separate workshops, the test manager and all stakeholders assign a priority number to each feature to test by applying a qualitative appreciation of the impact and likelihood of the risk level associated with that particular feature to test.
- In the case of high-risk areas, the priority number is calculated using quantitative risk analysis methods.
- All stakeholders are involved in determining the quality profile for all features to test using a standard set of quality attributes (and their relative importance level). Relevant test types are determined and the FTT is organized accordingly.
- Quality metrics are defined for every feature to test, making an objective evaluation of respective features to test possible.
- Together with the stakeholders, the test manager determines the different test levels at which the features to test will be tested and documents this in the test approach matrix. Gaps and overlaps between the different test levels are discussed with the stakeholders.
- Taking into account the priority number and the quality metrics of the different features to test, the test manager determines the test depth needed and assigns test design techniques accordingly. The test design techniques to be applied are documented in the test approach matrix.
- The test manager distributes the test approach matrix to all stakeholders. Approval is implicit, but can be formally organized if needed.

Agile Testing Add-ons

Agile Test Approach

In an Agile environment, it is important to determine the test approach in a way that reduces the risk of introducing regression and ensures the quality of the product. Therefore, it is recommended to think about the high-level approach at the very beginning of the project and to define the detailed test approach during the sprint planning meetings, depending on the user stories of the sprint backlog.

The purpose of creating the test approach is to clarify the test activities, test depth, and the level of regression testing to be performed.

Acceptance Criteria and Definition of Done

Acceptance criteria and definition of done are two important aspects that support teams in defining the appropriate test approach, and are defined in close cooperation with the Agile team(s) and stakeholders. Once the definition of done is agreed upon, the team should stick to the agreement and make sure that "done" really means that the developed features are tested and shippable.

Backlog

The product backlog contains the features to test and is ordered by the priority that is defined by the product owner based on business value, risks, and dependencies. From the testing point of view, the priorities of the features to test are highly dependent on the order of the product backlog items. Testers assist in prioritizing the backlog, but testing activities are only limited to the sprint backlog.

Agile Testing Quadrants

In Agile projects, different types of testing are performed to accomplish different goals. The Agile testing quadrants, a matrix originally developed by Brian Marick and later refined by Lisa Crispin and Janet Gregory, are a good starting point in building the test approach.[2] The four quadrants are numbered, but they should not be looked at sequentially because they do not represent any order or timeline of test execution.

The four quadrants reflect the different reasons for Agile testing. On one axis, the matrix is divided into tests that support the team and tests that critique the product. The quadrants on the left include tests that support the team as it develops the product. The concept of testing to help the developers is new to many testers and is one of the main differences between testing in a traditional project and testing in an Agile project. The other axis divides them into business-facing tests and technology-facing tests.

The Agile testing quadrants are a useful tool to identify the tests that need to be considered. On the other hand, the test automation pyramid introduced by Mike Cohn is a good model to identify the test automation strategy. Both models can be combined to design the high-level test strategy for a given Agile project. Continuous integration and deployment play an important role in setting up the test approach.

Keys to Success

- Testing is a whole-team approach and responsibility
- Testing is performed early and often
- Close collaboration with product owner and customer
- Provide and obtain feedback fast and constantly
- Apply Agile practices:
  - Test-driven development
  - Behavior-driven development
  - Acceptance-test-driven development
  - Exploratory testing
  - Continuous integration and deployment
  - Automated regression testing
  - Refactoring
  - Continuous improvement

[2] Crispin, Lisa, and Janet Gregory. *Agile Testing: A Practical Guide for Testers and Agile Teams.*Crawfordsville: Addison-Wesley, 2009

Mobile Application Testing Add-ons

Heuristics for Testing Mobile Applications

When identifying the features to test, specific heuristics with regard to testing mobile applications can be used.

One example of such heuristics is given by Jonathan Kohl and referred to by the acronym I SLICEDUP FUN:[3]

- **I**nputs: These represent the way you interact and control the device
- **S**tore submission: Check store submissions, make sure your app is approved to be distributed to the masses, regardless of the platform for which you are aiming
- **L**ocation: Some can be simulated, but never neglect the real world
- **I**nterruptions: Identify all possible interruptions that can happen while using your app
- **C**ommunication: What happens when your app is interrupted for a communication action
- **E**rgonomic: As apps are intended for tablets and smartphones, using apps should be easy and comfortable
- **D**ata in- and output: Different text inputs, overflow, invalid data, SQL injections
- **U**sability: Is it easy to use? Is it intuitive? Does it feel user friendly?
- **P**latforms: The variety is extensive given that the variety of devices keeps on expanding. Think about the lot of distinct manufactures, types of mobile devices, operating systems, and different versions of these operating systems. Do not forget about differences in internal and additional memory capacity, different possible screen sizes, built-in keyboard, touch pens, etc. Backwards compatibility should also be considered
- **F**unctional Properties: There are several settings that can be changed on a mobile device. Determine the ones that influence your app
- **U**ser Scenarios: Determine the users that will be using your app and work out user-based scenarios as tests
- **N**etwork Conditions: Test for signal strengths and variety of operator networks; explore changes and transitions

Quality Attributes

When considering the quality profile for mobile applications, there's a number of quality attributes that immediately come to mind:

Functionality	• Suitability • Security
Usability	• Understandability • Learnability • Operability
Efficiency	• Time-based • Resource-based
Maintainability	• Changeability • Stability
Portability	• Adaptability

However, the below table is not exhaustive and should by no means stop you from exploring other quality attributes. Be sure to check quality standards (e.g., ISO 25010) and/or quality heuristics to verify whether other quality attributes are relevant for the mobile application you are testing. Finally, make sure to use metrics where possible, which help to make selected quality attributes measureable.

Test Design Techniques

When creating tests, make sure to apply test design techniques that match the quality profile created earlier. For example, exploratory testing will help you in assessing the functionality and usability.

Emulation and the Cloud

Testing all the devices that are available on the market is merely impossible, both from a feasibility and budgetary point of view. An emulation tool has several advantages because it provides a cost-effective way to test in a close-to-real-life environment. These advantages add flexibility to performing tests on several devices quickly. There is emulation possible which supports test automation as well.

Cloud testing is another approach to take. Testing on remote live networks and/or simulated hardware provides the advantage of having your devices available at all times.

[3] Kohl, Jonathan. *Tap Into Mobile Application Testing*. Victoria, British Columbia: Leanpub, 2013. https://leanpub.com/testmobileapps

Beta Testing

Beta testing is a testing phase that can be organized after a more conventional testing phase is complete. The idea is to release a version of the mobile application under test to a selected group of identified future end users (closed beta testing) or even to the full user community (open beta testing), and to ask them to try the mobile application under real-world conditions.

The goal is to discover defects and/or issues in the mobile application under test from a user point of view that remained undiscovered during testing by the test team. It will offer the advantage of being able to fix the defects and/or issues before the mobile application is officially released to the full user community and prevent user dissatisfaction and bad publicity.

Beta testing, however, might not always be an option. It is okay to have a new game tested in beta, but having future end users test a new app that monitors vital functions is probably not an option.

Hallway Testing

Hallway usability testing is another way to involve future end users in testing (mainly) usability of the mobile application under test.[4] However, with hallway testing, these future end users are randomly selected from the "hallway." Randomly-chosen individuals are asked to try the mobile application under real-world conditions.

Working with randomly-chosen testers doesn't mean that this type of usability testing happens without preparation. On the contrary, the success of hallway testing depends on good preparation. Proper planning, choosing the right location for test execution, proper explanation of the purpose (assignment) to the chosen individuals, and strict timing are only a few things to consider when preparing hallway testing.

Advantages and disadvantages linked to hallway testing are more or less alike to those of beta testing.

In-the-Wild Testing

When it is not possible (and this will be the case on many occasions) to create a controlled test environment, in-the-wild testing comes to the rescue.[5] In-the-wild testing will provide test teams with the opportunity to increase test coverage. Tests that cannot be executed in the available test environments may be executed in the real world. It will allow for testing on more and real devices, more versions of operating systems, more mobile operating platforms, more network providers, more languages, and more user profiles.

[4] Miller, Jessica. "5 Killer Hallway Usability Testing Tips." Usability Lab (blog). September 17, 2014. http://usabilitylab.walkme.com/5-killer-hallway-usability-testing-tips/

[5] Hand, Richard. "'In-The-Wild Testing': The Missing Link in the QA Chain." *Software Test Professionals*. January 26, 2012. http://www.softwaretestpro.com/Item/5422/"In-The-Wild-Testing"-The-Missing-Link-in-the-QA-Chain/Test-and-QA-Software-Test-Professionals-Conference

FINANCE REPORT
ACCOUNT
REPORT
DASHBOARD
INCOME
OUT GOING
DASHBOARD > INCOME
TOTAL INCOME
37.91%
31.86%
30.23%

2. Test Build (TB)

2.1 Design Tests

Goals

"Design Tests" is the activity during which the project's test model is created according to the defined test approach. The test model is the project's collection of test procedures testing the features to test. A test procedure consists of test cases (high-level and/or low-level test cases), checklists, and in some cases, test scripts (automated test cases). The created test procedures will also result in test data needs, which are defined and created in this phase. Traceability toward the features to test created earlier is guaranteed.

Control over the testing process of **designing tests for the respective testing project** that satisfies the business requirement for testing of **creating a documented and repeatable test suite which covers the test project's features to test according to the test project's test approach** by focusing on:

- Creating a test procedure for every feature to test
- Detailing test procedures in test cases, test checklists, and/or test scripts
- Defining preconditions, logical steps, and pass/fail criteria
- Linking test procedures to features to test
- Linking test cases, test checklists, and/or test scripts to test procedures
- Creating low-level tests cases with required test data
- Creating an execution schedule based on priorities of the features to test

and is achieved by

- Creating a test model with test procedures, high-level test cases, checklists, and test scripts
- Creating traceability toward the features to test
- Adding test data
- Creating a test execution schedule

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

TB 2.1.1	TB 2.1.2	TB 2.1.3	TB 2.1.4
Create a Test Model	Traceability	Add Test Data	Create Test Execution Schedule

❯ TB 2.1.1 Create a Test Model

In this step, an assembled set of test procedures, test cases, checklists, and test scripts is created.

TB 2.1.1a – Define the Test Procedure
For each particular feature to test that needs to be tested, a test procedure that defines how this goal will be accomplished is created. Each test procedure will be translated into test cases, test checklists, and/or test scripts later on.

TB 2.1.1b – Create Test Cases
High-level test cases define, on an abstract level, a list of actions to be executed in order to execute a test procedure. Each high-level test case contains an initial input, an action to be performed, and an expected outcome. Later on, these high-level test cases will be turned into low-level test cases by adding concrete test data.

TB 2.1.1c – Create Checklists
Some test procedures cannot be verified by executing test cases, or creating test cases is not efficient. In those cases, defining a test checklist is the best solution. A test checklist defines a list of control actions to test a test procedure.

TB 2.1.1d – Create Test Scripts
Test scripts are the automated form of manual test cases.

❯ TB 2.1.2 Traceability

The "Determine Test Approach" activity generates a list of features which need to be tested during the project. During the "Design Tests" activity, the features to test are translated into test procedures. Traceability between test procedures (test cases and checklists) and features to test needs to be established. Traceability will enable and support progress monitoring (test build coverage, test execution coverage) and quality reporting.

❯ TB 2.1.3 Add Test Data

Low-level test cases are created by filling in specific test data in the high-level test cases defined earlier. This step also ensures the availability of the required test data by defining the test data needs and generating the required data.

❯ TB 2.1.4 Create Test Execution Schedule

The "Determine Test Approach" testing activity generates a prioritized set of features to test in an FTT. The priorities listed in this tree define the schedule of testing activities during test execution. The detailed preparation of this test execution schedule is performed in this step by ordering the test procedures.

Maturity

The testing activity **"Design Tests"** that satisfies the business requirement of **creating a documented and repeatable test suite covering the test project's features to test according to the test project's test approach** is at:

Initial level

The "Design Tests" testing activity is executed in an uncontrolled and chaotic way.

- Test cases are only created for the most important requirements.
- There is no traceability toward the FTT defined during the "Determine Test Approach" activity.
- Test data is created at the moment it is needed, generally during the test execution phase, and not in advance during this phase.

Starter level

Test procedures are created for each feature to test in every project.

- A high-level qualitative risk assessment is performed to define which features to test need to be tested.
- The decision to choose a test case or test checklist is made by the tester based on what seems to be the best solution.
- The decision to automate test procedures is made by testers based on when it seems to be profitable.
- A test specialist creates test scripts in the case that test automation is requested.
- A traceability matrix is created to have a clear view of the coverage of all requirements and risks.
- Queries are used to search for test data.
- A test execution schedule is created for the test procedures.

Experienced level

For every project, one test procedure is created for each feature to test.

- Creating a test case or checklist for a test condition is based on what seems to be the best choice.
- New test scripts are created for new functionalities; legacy test scripts need only be adapted.
- The traceability matrix gives an overview of the requirements and risks covered. It also provides a representation of the percentages of covered requirements and risks, which are displayed in a report.
- Test data can be searched for by queries or created when not available.
- Test data is created manually or automatically.
- An execution schedule is created for the pretest run and the test run. This execution schedule not only takes the prioritization of the FTT into account, but also the logical connection between test procedures, the release information, and the test environment.

Master level

For every project, a test procedure is created for each feature to test. The created test procedures are independent from each other.

- The decision to create test cases or checklists is made based on the knowledge of all of the guidelines.
- A library with test scripts is available within the organization. Newly-created scripts for new functionalities are added to this library.
- The traceability matrix gives an overview of the requirements and risks covered. It also provides a representation of the percentages of covered requirements and risks, which are displayed on a report.

- Test data can be requested from a specialized team.
- An execution schedule is created for the pretest run and the test run. This execution schedule not only takes the prioritization of the FTT into account, but also the logical connection between test procedures, the release information, and the test environment.
- The pretest run is automated.

Agile Testing Add-ons

Level of Design

In an Agile environment, test design is often replaced by high-level test cases, checklists, or even by exploratory testing.

The level of detail depends on the decision that the team has made together with the product owner and Scrum master. This decision can be based on:

- Available time for testing in the iteration
- Maturity of the testers
- Level of detail of the user stories
- Content of the definition of done

Exploratory Testing

In Agile projects, Lean principles are often applied in order to reduce and eliminate waste. This is one of the reasons why test documentation is not as extensive as in a Waterfall project. The focus is more on working software than exhaustive documentation. There is more time spent on creating automated tests than designing manual test cases that are only executed once during sprints.

Testing in an exploratory way is in fact simultaneous test design, test execution, and learning about the product. A common misconception is that these kinds of tests are not manageable or accountable. When using test charters (an up-front, defined, specific test mission), time-boxed test sessions, and debriefs, this can be a very powerful test approach, generating precious and quick feedback.

Test Basis

The creation of test cases is one of the daily test activities during the sprint. However, this task is limited to the user stories in the scope of the sprint that are committed by the team. Each iteration is focused on a few stories, taking the priority from the product backlog into consideration. All other product backlog items are out of scope.

A good user story focuses on the value a user gains from the system and uses the "INVEST" model[6]:

- **I**ndependent
- **N**egotiable
- **V**aluable
- **E**stimable
- **S**mall
- **T**estable

At the beginning of the sprint, the tester often identifies the high-level test cases based on the defined acceptance criteria. A good practice is to do this before the user story is implemented to clarify the scope with the product owner and to define the correct test cases.

Practices

Depending on the maturity level of the team, one or more of the following practices can be applied by the Agile team:

- Test-driven development
- Acceptance-test-driven development
- Behavior-driven development
- Story-test-driven development
- Specification by example

Traceability

Both detailed and high-level test cases can be written down in a tool, ranging from mind maps, to simple Excel files, to a test management tool. The link between features and test cases is not necessarily present in Agile projects, and depends on the tool in place to design the tests and the definition of done.

[6] Wake, William. "INVEST in Good Stories, and Smart Tasks." XP123 *Exploring Extreme Programming* (blog). August 17, 2003. xp123.com/articles/invest-in-good-stories-and-smart-tasks/

For tests that are only executed once and not automated, it does not make any sense to maintain traceability. In fact, in Agile projects, all stories accepted by the product owner have been tested and met the definition of done. So, it is obvious that all features were tested before delivery.

On the other hand, for automated tests, it is useful to know which user stories are covered by test automation as these tests are constantly running.

Mobile Application Testing Add-ons

Resources

Resources such as those below might not be self-evident or could even be unavailable:

- SIM cards
- SD cards
- Wi-Fi network
- Mobile devices (tablets, smartphones, etc.)
- Emulators
- Extra software of peripherals needed for your tests

When designing mobile application tests, you'll need to adapt to the choices made in the test approach and the resources made available in the test plan. Some tests should not be designed as it will not be possible to execute them anyway due to unavailability of the needed resources.

Mobile-specific Test Cases

Mobile devices are portable, which will result in location changes, network changes, and screen orientation changes while applications are being used. Mobile devices are also often controlled by a touchscreen or voice control. All of these factors will result in mobile-specific test cases.

Below, you can find a list that contains specific tests for mobile applications. Though this list is not exhaustive and cannot guarantee any test coverage completeness, it will give you a good understanding of what makes mobile test design unique. It is constructed around five main test categories:

- Installation
- Compatibility
- GUI (User Interface)
- Network
- Interrupts

Mobile App Checklist	
Installation	• Application must be installed via a computer link without errors occurring • Application must be installed via a download link without errors occurring • Application opening time is below 20 seconds • Application settings opening time is below three seconds • Application continues installing process even after the screen of mobile device has been locked • Application must be uninstalled without errors occurring
Compatibility	• Application runs on different OSs • Application runs on different devices • Application runs with different languages • Application runs with different time/date settings • Application functions properly even when other applications from same author are installed on the same mobile device

GUI	• There should be no overlapping images or buttons • The user interface should be user-friendly and easy to use • Changing from landscape to portrait orientation (rotating the device) should not affect the layout or cause any errors • Text should not be cut off/overlapped by images, borders, or buttons • All GUI tests should be performed on devices with different native screen sizes (DPI and screen size) • Once the app is installed, the app icon in the app drawer should be checked/verified to see if it meets expectations • All buttons integrated in the application should have a function/reaction after being pressed • All buttons should execute the function they describe after being pressed • Navigation within the application should work with both hard buttons (on the device) and soft buttons (in the application) • Pressing the back button (hard or soft) should not close the application without a confirmation message • Navigating to a new functionality should not take more than five user actions
Network	• Application runs with Wi-Fi connection • Application runs with cellular connection (3G/4G) • No errors should occur with weak or lost Wi-Fi connection during use of the application • No errors should occur with weak or lost cellular connection (3G/4G) during use of the application • No errors should occur when the device transitions from cellular to Wi-Fi connection (or the other way around) during use of the application • No errors should occur when the device loses GPS localization • No errors with Bluetooth transfer and connections • Application closes connections correctly
Interrupts	• Incoming phone calls, video calls, or text messages should not cause any errors • Launching and pausing a music/movie player should not cause any errors • Launching and pausing the camera should not cause any errors • Launching and pausing other applications should not cause any errors • Plug or unplug USB should not cause any errors • Changing the USB connection should not cause any errors • Lock and unlock should not cause any errors • Application should have no memory leaks or battery drain problems • In case the application emits sound, this sound should not be heard when interrupted by a phone call or another app that emits sound • Minimizing the application to status bar and launching it again should not cause any errors

The list is also a good starting point for creating your own checklist dedicated to your mobile application's specific needs. Simply expand the list with your own mobile-specific test cases and/or consolidate them with other existing checklists.

2. Test Build (TB)

2.2 Set Up Test Environment

Goals

Setting up a test environment for your testing project is performed in order to fulfill the test environment requirements and to make sure that all necessary test data is available to verify your testing project in each test level.

Control over the testing process of **setting up test environment(s) for the respective testing project** that satisfies the business requirement for testing of **having controlled test environment(s) available, guaranteeing full support and unhindered execution of the test project's test execution schedule** by focusing on:

- Identification and installation of hardware and software components (computer, network, middleware, operating system, etc.) of the test environment(s), supporting the test project's test approach
- Identification and installation of process components (planning, scheduling, backup, restore, database management, authorization, security, release management, etc.) of the test environment(s) supporting the test project's test approach
- Identification and installation of development tools and test tools supporting the test project's test approach
- Installation of the correct version of the respective test object and relevant satellite systems and interfaces in the test environment(s)
- Artificially creating and/or copying from production the (initial) test data as described in the test project's test design

and is achieved by

- Determining the test environment requirements
- Setting up the test environment(s)
- Setting up the test tools
- Installing the test object
- Setting up test data

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

TB 2.2.1	TB 2.2.2	TB 2.2.3	TB 2.2.4	TB 2.2.5
Determine Test Environment Requirements	Set Up Test Environment	Set Up Test Tools	Install Test Object	Set Up Test Data

TB 2.2.1 Determine Test Environment Requirements

An inventory of test environment requirements is created for the test project. The possible limitations are listed as well.

TB 2.2.2 Set Up Test Environment

The test environment(s) is (are) installed. During execution of the planned tests, the environment(s) will be maintained.

TB 2.2.3 Set Up Test Tools

The necessary test tools are installed and maintained.

TB 2.2.4 Install Test Object

The test object is installed and maintained during the different test levels.

TB 2.2.5 Set Up Test Data

The necessary test data is determined. The test data is artificially created and directly entered into the test environment(s) and/or copied from the production environment (in this case, measures to protect data privacy should be considered).

Maturity

The testing activity **"Set Up Test Environment"** that satisfies the business requirement of **having the controlled test environment(s) available, guaranteeing full support and unhindered execution of the test project's test execution schedule** is at:

Initial level

The testing activity is executed in an uncontrolled and chaotic way.

- The activity is not organized for all projects.
- If the activity is executed, it is executed in a chaotic and ad hoc manner.

Starter level

The test manager takes control of the "Set Up Test Environment" activity by listing a set of requirements for the test environment(s) and formalizing an approval by the principal stakeholders and environment Single Point of Contact (SPOC).

- The test manager lists requirements for the test environment(s) (test environment, test tool, test object).
- The principal stakeholders and test environment SPOC approve test environment requirements.
- Respective suppliers provide the requested test environment(s), test tools, and test object.
- Respective suppliers informally inform the test manager when the test environment(s), test tools, and test object are ready.
- The project team (development/test) maintains the test environment(s) and test tools.
- Respective suppliers maintain the test object.
- Test data is determined for low-level test cases.
- Test data is supplied (newly created or copied from production) by the project team (development/test).

Experienced level

The test manager involves the principal stakeholders when executing the "Set Up Test Environment" activity. Different test environments per test level are available (Development, Testing, Acceptance, and Production (DTAP) model).

- For each project, the test approach is mapped on the DTAP model (see "Determine Test Approach" activity).
- The test manager and principal stakeholders list requirements for the test environment(s) (test environment, test tools, test object) per test level.
- The principal stakeholders and test environment SPOC approve test environment requirements.
- Respective suppliers provide requested the test environment(s), test tools, and test object.
- Respective suppliers formally inform the test manager when the test environment(s), test tools, and test object are ready.
- Limitations of the test environment(s) are being traced back to the test approach and communicated to the principal stakeholders.
- The test manager adapts the test approach and test schedule according to actual state of test environment(s), test tools, and test object.
- Respective suppliers maintain the test environment(s), test tools, and test object.
- Test data is determined for low-level test cases.
- Test data is supplied (newly created, selected and copied from test data repository, or selected and copied from production) by the project team (development/test).

Master level

The test manager involves all stakeholders when executing the "Set Up Test Environment" activity. There's a clear process for determining and requesting test environment requirements with clear involvement of all involved parties, i.e., test manager, all stakeholders, and environment SPOC.

- For each project, the test approach is mapped on the DTAP model (see "Determine Test Approach" activity).
- The test manager and all stakeholders list requirements for the test environment(s) (test environment, test tools, test object, test data) per test level through the request form.
- All stakeholders and the test environment SPOC approve the test environment requirements request form.
- Respective suppliers provide the requested test environment(s), test tools, test object, and test data (newly created, selected and copied from test data repository, or selected, copied and scrambled from production).
- Respective suppliers formally inform the test manager when the test environment(s), test tools, and test object are ready through the delivery form.
- Respective suppliers communicate the test environment(s), test tools, and test data gaps through the delivery form.
- Respective suppliers formally communicate possible flaws of the test object through release notes.
- Limitations of the test environment(s) are being traced back to the test approach and communicated to all of the stakeholders.
- The test manager adapts the test approach and test schedule according to actual state of test environment(s), test tools, test object, and test data.
- Respective suppliers maintain the test environment(s), test tools, test object, and test data.

Agile Testing Add-ons

Selection and Set Up of Tools

Selection of the tools needed is a team responsibility. During the initial planning, often defined as sprint zero, the whole team gathers the most important requirements for the tooling to be used. During the development life cycle, the team also continuously looks for improvements that might result in the introduction and usage of new tools. From a purely testing point of view, the defined test approach and the Agile testing quadrants will impact the final selection of test tools and test environment to be used in the project. In addition to classical test tools that support test management, test case management, defect management, etc., Agile projects in particular also require tools to support continuous integration and deployment.

Testers will assist in setting up the selected tools and will also be involved in setting up the continuous integration framework to follow up on quality of code and to monitor unit tests or other automated test scripts. The challenge for Agile teams is figuring out how to align testing with the speed of development and delivery without loss of quality. Continuous testing is becoming more important, but doesn't happen without having automation in place.

Test Data

During the creation of the product backlog, testers can already define the test data types and check how easily the test data can be obtained. If test data needs a long preparation time or requires extensive manipulation, testers can start the test data setup process during the early stages of the project in order to have all data ready when the team starts developing the stories.

It also might be useful to think about a mechanism to automatically generate test data. This way, test data can be easily re-created several times during the ongoing sprint, or even future sprints.

Supporting Tools

One of the main principles of an Agile software development process is the importance of communication between all people involved in the project. The main goal of communication is to share information. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. In the case that the whole team is co-located, the team should avoid using unnecessary tools for communication. However, when this is not the case, it is important to think about different tools to facilitate team communication, taking the organizational structure into account.

A tool that can support planning and tracking of the progress of user stories that have business value to customers is also very important for Agile teams. There are many tools on the market to plan, track, and manage Agile projects, iterations, user stories, and tasks. Virtual Scrum/task boards are typical examples of such tools.

It's up to the team to decide whether such a planning tool adds value, or if the user stories and tasks can be tracked with post-its on a whiteboard alone. Co-located teams may prefer working with a whiteboard instead of using a virtual task board. On the other hand, a virtual task board is a powerful tool to support distributed teams in order to ensure all team members are always up to date.

Keep in mind that tools should never replace the benefits achieved from daily stand-up meetings, retrospectives, or close cooperation with stakeholders.

Test Object

The test object is installed as soon as a testable piece of code is available in order to have short feedback loops. Continuous integration and deployment are concepts supporting the testers and other team members in rapidly deploying every change to a test, acceptance, or production environment.

Mobile Application Testing Add-ons

Self-managed Test Environment

One of the characteristics of a mobile test environment is that part of it is often managed by the testers themselves. The mobile devices such as smartphones and tablets can be managed by the testers in the following ways:

- Firmware/operating systems are flashed on the devices by the testers
- Software is installed on the devices by the testers
- Sim cards, SD cards, cables, chargers, etc., are managed by the testers
- Testers keep track of what software is installed on which devices
- Device emulators and credentials are managed by the testers
- Device cloud credentials and access rights are managed by the testers

Please note that flashing firmware on mobile devices is not without risk. When done incorrectly, upgrading or downgrading a mobile device's operating system might result in irreversible damage to the device. Dedicated flashing tools should be used for this and one should take greatest care in selecting a proven (custom) ROM to install (e.g., Cyanogenmod). So, testers downgrading and upgrading the devices' software should be trained before executing these actions and should not proceed lightly.

Hardware or Virtualization

Having every kind of device you want to test physically available is not feasible for every project. The purchase and maintenance of a hardware portfolio should not be underestimated. Mobile devices are also rapidly evolving, and thus, a device can become "outdated" after a limited period of time.

Virtualization through third-party-managed device clouds or through either third-party or self-managed emulators is a very viable alternative. The main target devices can be purchased and self-maintained while extra devices and older OS versions can be tested on the device clouds or emulators.

Managing your devices and other hardware such as SIM cards, SD cards, cables, chargers, etc., should also not be underestimated. All of these items are small, expensive, and easy to lose. Keeping a who-has-what list and keeping it updated should be considered best practice. A list with all of the info on the devices and what OS version is currently installed can also be very time saving.

Authorizations

While you may be perfectly aware of what test environment you want, you might not always get what you want. Company policies may interfere with the setup of your test environment in many different ways:

- Authorization to manage credentials and access rights for devices, emulators, and cloud services
- Authorization to leave the building for field testing
- Authorization to connect to the company's wireless network
- Authorization and budget to purchase mobile devices, SD cards, SIM cards, etc.
- Authorization to use a company car to perform road tests

Always check the respective company policies and/or with corresponding management to determine whether you have the needed authorization and budget to set up the test environment as you would like. If not, negotiate whether you can acquire authorizations on a temporary basis and/or part of the budget required.

3. Test Execution (TE)

3.1 Verify Test Environment

Goals

During the "Verify Test Environment" activity, it is verified that the test environment(s) is (are) correctly set up according to the test environment requirements list and that the test object is correctly installed and ready for the next phase of testing.

Control over the testing process of **verifying the test environment(s) for the respective testing project** that satisfies the business requirement for testing of **making sure that the test environment(s) is (are) correctly set up and the test object is correctly installed in the test environment and preventing waste of testing effort by not starting test execution in a test environment that is not compliant with the defined test environment requirements** by focusing on:

- Verification of availability, quality, and stability of the test environment(s), test tools, test object, and test data through static checking of the test environment requirements list
- Verification of availability, quality, and stability of the test environment(s), test tools, test object, and test data through dynamic execution of selected test cases (intake test)
- Summarizing the results of the static check and the intake test in order to evaluate the availability, quality, and stability of the test environment(s), test tools, test object, and the test data
- Providing advice to the stakeholders regarding starting execution of the next test iteration, test run, test level, etc., based on the results of the static check and the intake test

and is achieved by

- Performing a static check of the test environment(s)
- Performing an intake test of the test environment(s) and the test object

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

TE 3.1.1 Perform Static Check

TE 3.1.2 Perform Intake Test

❯ TE 3.1.1 Perform Static Check

Verification of the quality of the test environment(s) is performed through static checking of the test environment requirements list. This way, IT/Infrastructure can check whether the deployments are as expected.

❯ TE 3.1.2 Perform Intake Test

Verification of the quality and stability of the test environment(s) and the test object is performed through dynamic test execution of the prepared pretest execution schedule. The test team will be responsible for this step.

Maturity

The testing activity **"Verify Test Environment"** that satisfies the business requirement of **making sure that the test environment(s) is (are) correctly set up and the test object is correctly installed in the test environment and preventing waste of testing effort by not starting test execution in a test environment that is not compliant with the defined test environment requirements** is at:

Initial level

The "Verify Test Environment" testing activity is executed in an uncontrolled and chaotic way.

- The activity is not organized for all projects.
- If the activity is executed, it is executed in a chaotic and ad hoc manner.

Starter level

The test engineer or test specialist takes control of the "Verify Test Environment" testing activity and uses the test environment requirements list to verify the test environment(s). The test engineer or test specialist also executes a dynamic test(s) to verify the test environment(s) and test object, but it is not a dedicated test, rather it is a subset of the test cases that are to be executed on this test object.

- For each project, the test environment requirements list is used as a (static) checklist to check the availability and quality of the test environment(s) and test object as required.
- Dynamic test execution is performed to verify the quality and stability of the test environment(s) and the test object as required.
- Improvised manual test cases or, in the best case, a small selection of prepared test cases are run.
- Actual results and expected results of the dynamic test execution are compared.
- When expected and actual results do not match, defects are logged.

Experienced level

The test engineer or test specialist takes control of the "Verify Test Environment" testing activity and uses the test environment requirements list to verify the test environment(s). The test engineer or test specialist also executes the planned dynamic test(s) to verify the test environment(s) and test object. This test is a dedicated test for this purpose and can be automated.

- For each project, the test environment requirements list is used as a (static) checklist to check the availability and quality of the test environment(s), test tools, test object, and test data as required.
- Dynamic test execution of the prepared pretest execution schedule (see "Design Tests" activity) is performed to verify the quality and stability of the test environment(s) and the test object as required.
- Test cases (preferably automated), as planned in the pretest execution schedule, are run.
- Actual results and expected results are compared.
- When expected and actual results do not match, defects are logged.

Master level

The test engineer or test specialist takes control of "Verify Test Environment" testing activity and uses the test environment requirements list to verify the test environment(s). The test engineer or test specialist also executes the planned dynamic test(s) to verify the test environment(s) and test object. This test is a dedicated test for this purpose and is automated.

- For each project, the test environment requirements list is used as a (static) checklist to check the availability and quality of the test environment(s), test tools, test object, and test data as required.
- Dynamic test execution of the prepared pretest execution schedule (see "Design Tests" activity) is performed to verify the quality and stability of the test environment(s) and the test object as required.
- Automated test cases, as planned in the pretest execution schedule, are run.
- Actual results and expected results are compared.
- When expected and actual results do not match, defects are logged.

Agile Testing Add-ons

Sprint-ready Checklist

During sprint zero, when the initial planning is done, a sprint-ready checklist is created, consisting of a set of tasks that must be completed before a sprint can start and run smoothly. If there is major incompletion, it would be risky to start a sprint as it will experience obstacles, delays, and impediments. Therefore, at the end of sprint zero, checks are performed to prove that the environment is available according to the specified requirements.

Sanity Checks

Short feedback loops are the strength of working in an Agile environment. This requires continuous integration as well as deployment. To ensure that during iterative development the test environment is ready to be used, automated test scripts are often written to replace manual sanity checks.

Test-driven development and acceptance-test-driven development are some of the key Agile principles used to validate that the test object is completely installed and the developed features are correctly deployed.

Mobile Application Testing Add-ons

Self-managed Test Environment

In mobile application testing projects, part of the test environment is often self-managed by the test team. As a result, at least for the part that they are managing themselves, test engineers or test specialists will be responsible for checking the state of the test environment. The static check of this part of the test environment cannot be delegated to others.

Keeping track of (the requirements of) the many devices, hardware, and software, therefore, is of utmost importance. Some measures that can help attain this are:

- Maintain an inventory of all devices, hardware, and software
- Keep track of physical location of the devices ("who-has-what"list)
- Use the devices strictly for testing
- Set up a mechanism that will automatically reset the devices on a regular basis (i.e., every night, week, etc.)

Resetting Devices

Resetting guidelines are recommended when several testers are working with a shared portfolio of test devices. For testing consistency and to ensure that every tester can start his tests in the same conditions as any other tester, it is recommended to regularly reset the test devices. This can be done in many different ways: before tests, after tests, on planned intervals, or even automatically when you are working with emulators or cloud devices.

Being able to start in a "clean" environment might prevent errors that would be caused by previous tests that were performed. Do note that not all tests require a reset; some tests actually require the application to already be installed and used for a certain period of time.

When the devices are regularly reset, the device will have to be set up again, test data has to be re-imported, and the application will have to be installed again every time.

Because resetting before testing is good for quality and consistency but does take some time, you will have to assess the frequency of resetting (per test/daily/weekly, etc.) to make sure the process is beneficial for you.

Service Level Agreements (SLAs) and Key Performance Indicators (KPIs)

When the test environment is managed by external parties, make sure that SLAs and/or KPIs addressing the availability and verification of the test environment are part of the agreement made with the supplier.

Intake Test

Dynamic tests executed for purposes of verification of the test environment will have to be executed on a regular basis, and probably also on many different devices (simulations, physical devices, or devices in the cloud). Hence, these tests are the ideal candidates for test automation. The following points should be taken into consideration when selecting the test cases for the intake test:

- Try to minimize the amount of test cases in the test set
- Create the test cases in the test set to be as generic as possible so they can be reused on multiple test devices
- Work with parameters to easily adapt the test set for different devices with different features
- Limit the number of devices that will actually be tested to a small group of devices representing all devices that are part of the scope

3. Test Execution (TE)

3.2 Execute Tests

Goals

The testing activity "Execute Tests" involves executing tests to verify that the test object meets the requirements, making use of a tool for executing tests, logging defects, and reporting on test execution progress and found anomalies.

Control over the testing process of **executing the designed tests for the respective testing project** that satisfies the business requirement for testing of **controlled test execution according to the test project's test execution schedule and formal reporting of observed defects** by focusing on:

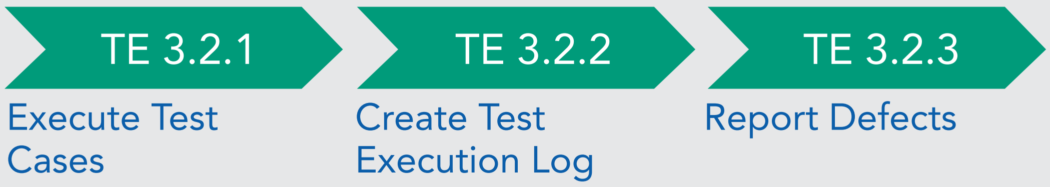
- Execution of manual test cases according to the test project's test execution schedule
- Execution of automated test scripts according to the test project's test execution schedule
- Verifying checklists according to the test project's test execution schedule
- Logging test execution details in test execution log
- Comparing actual test results with expected results
- Analyzing discrepancies between actual results and expected results
- Reporting defects in the case of confirmed discrepancies between actual results and expected results, and adding their severity

and is achieved by

- Executing the test project's test execution schedule
- Creating detailed test execution log
- Reporting defects in the case of anomalies

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.



TE 3.2.1 Execute Test Cases

Execution of the designed test cases to verify and validate that the test object meets the requirement specifications. An appropriate test automation tool will be required when considering automating part of test execution.

TE 3.2.2 Create Test Execution Log

Reporting on necessary information regarding the executed tests. This can also be done in a semi-automatic or automatic way using an appropriate test management tool.

TE 3.2.3 Report Defects

The possible discrepancies found between actual and expected results will be reported during this step. A defect management tool can be used to support this step. Reporting defects should not be done until the defect has been carefully checked to determine whether it is a real defect.

Maturity

The testing activity **"Execute Tests"** that satisfies the business requirement of **controlled test execution according to the test project's test execution schedule and formal reporting of observed defects** is at:

Initial level

The "Execute Tests" testing activity is executed in an uncontrolled and chaotic way.

- Some tests are executed.
- A test execution log is not always created.

Starter level

For every test project, the "Execute Tests" testing activity is performed to verify that the test object meets the requirements.

- The test cases/scripts/checklists are executed based on the test procedure and the test schedule.
- The test execution log, with relevant information (tester, start time, end time, result), is created manually during test execution by using a simple word processor or spreadsheet.
- Discrepancies between the expected results and the actual results are logged in a tracking tool.

Experienced level

For every test project, the "Execute Tests" testing activity is performed to verify that the test object meets the requirements. Defects are analyzed and priorities are determined on a project level.

- The test cases/scripts/checklists are executed based on the test procedure and the test schedule.
- The test execution log, with relevant information (tester, start time, end time, result), is created and maintained during test execution by using some sort of test management tool.
- The discrepancies between the expected results and the actual results are analyzed to establish their root cause.
- The severity of the defects is determined.
- Defects and/or remarks are reported in the defect template. The defect template is determined on the project level.

Master level

For every test project, the "Execute Tests" testing activity is performed to verify that the test object meets the requirements, defects are analyzed, and priorities are determined on company level.

- The test cases/scripts/checklists are executed based on the test procedure and the test schedule.
- The test execution log, with relevant information (tester, start time, end time, result), is created and maintained during test execution by using a test management tool.
- The discrepancies between the expected results and the actual results are analyzed to establish their root cause.
- The severity of the defects is determined.
- Defects and/or remarks are reported in the defect template. The defect template is determined on the company level.

Agile Testing Add-ons

Agile Testing Quadrants

Test execution involves carrying out the manual or automated test cases that have been created, or in case of exploratory testing, test cases that are designed on the spot.

During an iteration, different types of testing are performed to accomplish different goals. These different types of testing are listed in the Agile testing quadrants, a matrix originally created by Brian Marick and later refined by Lisa Crispin and Janet Gregory.[7]

The four quadrants reflect the different aspects of testing. On one axis, the matrix divides tests that support the team and tests that critique the product. The other axis divides them into business-facing tests and technology-facing tests.

Technology-facing Tests Supporting the Team

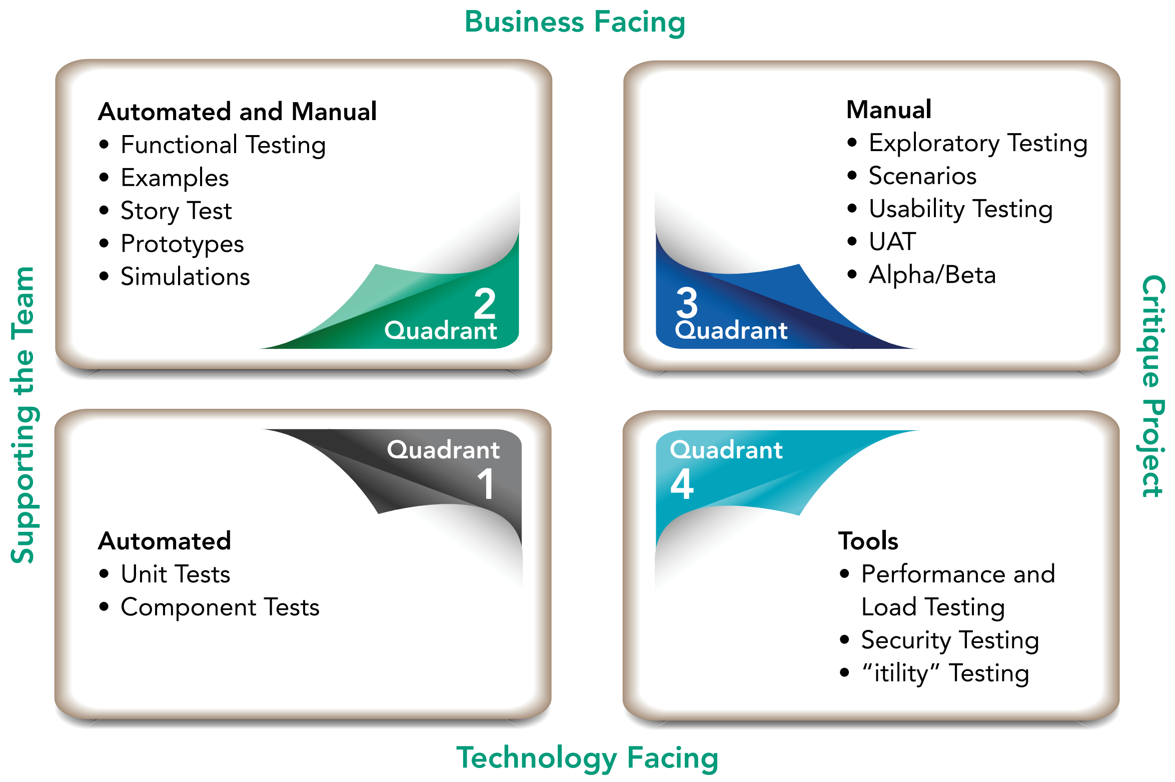
The tests in quadrant 1 function as a sort of guide for development, and are often part of test-driven development (TDD). The purpose of these tests is to verify that the code carries out what the business has described on a functional level. They are not intended for any form of customer use. They are related to the "internal quality" of the software.

Unit and component tests are a big part of this quadrant, but it is important to remember that they need a solid basis of TDD and a continuous integration process in place. Unit and component testing should result in a higher quality of code handed over which, in turn, will improve morale for all team members.

Business-facing Tests Supporting the Team

The purpose of quadrant 2 tests is to drive development on a higher level with business-facing tests. These tests are created to ensure the high-level system behavior responds in the way the business user intended. The tests in this quadrant are mainly confirmatory tests and their main purpose is to provide early feedback. Quadrant 2 tests ensure "external quality."

Key to successfully defining business-facing tests is making sure that they express requirements based on examples in a language and format that both the



[7] Crispin, Lisa, and Janet Gregory. *Agile Testing: A Practical Guide for Testers and Agile Teams*.Crawfordsville: Addison-Wesley, 2009

customer and development teams can understand. Usually, user stories are at the heart of these requirements. User stories are written by the business experts and, at first, usually do not contain a lot of detail. It is a starting point for an ongoing conversation between business experts and the development team. Testers can help elicit examples and context for each story. The combination of story, conversation, context, and examples finally makes up the requirement.

Quadrant 2 tests are written before coding is started. This is also known as acceptance-test-driven development (ATDD). Testers write tests in close cooperation with developers and the business as a first step in design and development because the tests help the developers understand what to write. The tests are typically automated, using automated acceptance testing frameworks like Fitnesse, Cucumber, Selenium, or other test tools.

Typical tests in this quadrant are examples, functional tests, story tests, prototypes, mock-ups, and simulations.

Business-facing Tests that Critique the Application

Quadrant 3 encompasses tests that critique the software, but do so in such a way that executing these tests will review the software from a user point of view and consider ways to further improve the product. Even business experts can overlook features and elements that real users expect, or express needs in an incorrect way. Therefore, the outcome of quadrant 3 tests will often guide the creation of new or additional stories or requirements. In general, these tests give confidence to the team that they are delivering working software, and that both functional and non-functional criteria have been met.

Typical quadrant 3 tests are exploratory testing, user scenario testing, alpha and beta testing, user acceptance testing, and usability testing. All of these tests explore the application under test in a way similar to how an end user would use the product. As a result, application failures under normal consumer use will be exposed and not-yet-identified requirements will surface. Depending on the particular test, actual end users can and will be involved during test execution.

Tests in quadrant 3 are usually executed manually. By automating quadrant 1 and quadrant 2 tests to the highest level possible, resources will be available for the manual text execution in quadrant 3.

Technology-facing Tests that Critique the Application

Quadrant 4 tests are concentrated more on non-functional requirements. Since most of the tests performed in this quadrant can only be executed by a tool or application experts, this quadrant is supported more by the technology than by the business, and its tests are often executed by resources with the necessary skills and tool knowledge.

Because testers often focus on testing the functionality and pay less attention to reliability or performance, explicitly adding these to different stories might be a good idea. Before executing these kinds of tests, however, the team should identify the expectations so they are able to measure unsatisfactory behavior. Working in close cooperation with the customer may provide clarity in case of any doubts.

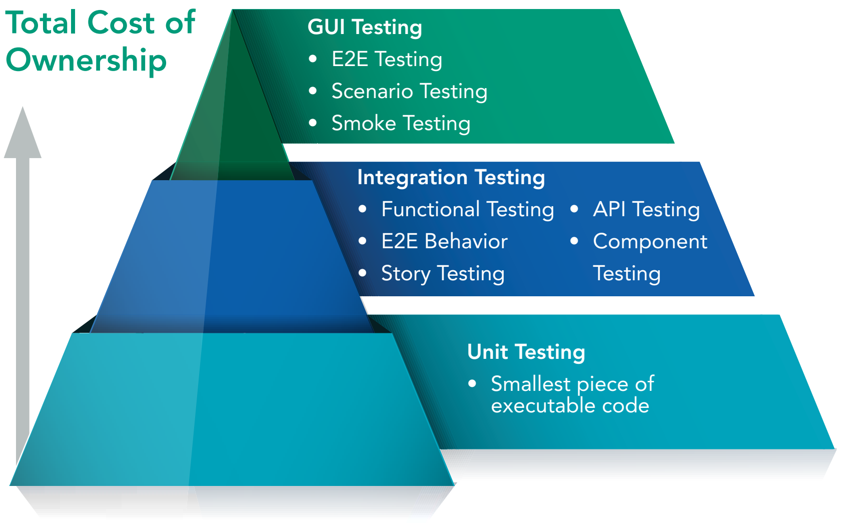
Typical quadrant 4 tests are performance and load testing, security testing, and installability testing.

Agile Testing Pyramid

Automated test execution is a must in an Agile development process, and is key to success in Agile testing. Test automation enables testers to focus on the newly-developed stories rather than having to manually execute regression tests over and over again to prove stability. Without adequate automation, there's not enough time for exploratory testing in an Agile iteration. The need for automated testing almost becomes essential to generate the velocity needed to ship working software in short iterations.

The Agile testing pyramid, a concept developed by Mike Cohn, is often used by Agile teams to guide test automation efforts.[8] The cost of ownership increases as you move up through the different layers of the test automation pyramid: the cost of creating and running automated high-level, end-to-end tests is considerably higher than creating and running automated low-level unit tests. However, when moving up through the different layers of the pyramid, business relevance and coverage also increases.

[8] Cohn, Mike. *Succeeding with Agile: Software Development Using Scrum.* Ann Arbor: Addison-Wesley, 2009



The three layers of test automation reflect this balance between costs associated with the automation of different test types and associated business relevance. Automation of unit tests is fast and cheap, but has low end-user relevance. Automation of E2E tests provides great coverage and has high end-user relevance, but is hard, time consuming, and expensive.

Report Defects

Defects found during test execution are reported immediately to the developers by using a defect management tool or face-to-face communication. The approach to managing Agile defects is to discuss, almost immediately after discovery, a possible solution for each defect. This does not mean that all defects have to be fixed immediately. It can still be a business decision whether a defect needs to be fixed or not.

Mobile Application Testing Add-ons

Build and Platform information

When testing mobile applications, exactly the same rules apply for creating execution logs and reporting defects as when testing non-mobile applications. There should not be any difference.

However, testers should be aware that proper creation of test execution logs and defect reporting becomes even more complex as the need for precise and detailed information about the test object and environment rises. In order to guide their efforts in determining the root cause of reported defects, developers (and other stakeholders) will require precise information about:

- **Build:** component and version of the objects under test
- **Platform:** device type, operating system, and hardware and software components of the mobile test device

This information can be found in the logs created by special apps made for this purpose (e.g., Catlog for Android OS). These logs grow to be very large when using the device for a long time, so it is best practice to clear the logs regularly.

Generalizing Defects

When a defect is being detected, the tester should apply reasonable effort to determine whether this defect also occurs under different conditions. Again, this is not different from defect reporting when testing non-mobile applications.

However, when testing mobile applications, evaluating "reasonable effort" to further investigate whether a defect is more general than it first appears can quickly become rather complex. More aspects need consideration, such as determining whether the same defect is also showing on other device types, with other OSs, and with other connection types; the availability of the right test environment; etc.

When there is a clear risk that additional investigation effort will have too much impact on the project's test schedule and test budget, the project's test manager should be consulted to jointly determine what can or cannot be done.

4. Test Project Closure (TPC)

4.1 Evaluate Test Project

Goals

The testing activity "Evaluate Test Project" involves writing and distributing a final test evaluation report that clearly evaluates the test project itself. It gives an overview of the challenges faced during the test project and lessons learned.

Control over the testing process of **evaluating the respective test project** that satisfies the business requirement for testing of **performing a clear evaluation of the test project, identifying the weak and strong points, and using the lessons learned to improve future projects** by focusing on:

- The baseline and evolution of the test plan
- The feedback of the test team
- The conclusions and findings of the regular test reports issued during the project (e.g., test status reports, test summary reports)

and is achieved by

- Holding a lessons-learned meeting
- Writing a test evaluation report
- Distributing this test evaluation report

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

TPC 4.1.1	TPC 4.1.2	TPC 4.1.3
Gather Test Summary and Test Status Reports	Create Test Evaluation Report	Distribute Test Evaluation Report

❯ TPC 4.1.1 Gather Test Summary and Test Status Reports

The first step is to gather all relevant documentation (test plan, test summary reports, test status reports, etc.). A lessons-learned meeting can be organized with the test team in order to fully understand the challenges that were faced during the test project.

❯ TPC 4.1.2 Create Test Evaluation Report

With all relevant information at hand, the goal in this step is to write the test evaluation report according to the relevant project or company guidelines for writing test evaluation reports.

❯ TPC 4.1.3 Distribute Test Evaluation Report

Once the test evaluation report is written, it needs to be distributed to all involved stakeholders. A formal signoff might be required.

Maturity

The testing activity **"Evaluate Test Project"** that satisfies the business requirement of **performing a clear evaluation of the test project, identifying the weak and strong points, and using the lessons learned to improve future projects** is at:

Initial level

The testing activity is executed in an uncontrolled and chaotic way.

- The activity is not organized for all projects.
- If there is an evaluation of the test project, it takes place in a chaotic and uncontrolled way.

Starter level

The test manager gathers all relevant information, writes the test evaluation report, and distributes it in an informal way.

- For each project, the activity is organized.
- All project information is gathered: project test plan, detailed test plan, test status reports, and test summary reports.
- A test evaluation report is written at the end of each project.
- An informal handover to inform principal stakeholders and the project manager about the test evaluation report is set up.

Experienced level

The test manager gathers all relevant information, writes the test evaluation report, and distributes it in a formal way to all principal stakeholders.

- For each project, the evaluation of the test project is organized. Lessons-learned meetings with the test team take place and practical details are written for the test archive.
- Various types of project information are gathered: project test plan, detailed test plan, test status reports, and test summary reports.
- A test evaluation report with the consolidation of the lessons-learned meetings is written.
- Company guidelines for writing the test evaluation report are met.
- A formal distribution of the test evaluation report is set up and the report is stored in a central project archive.

Master level

The test manager gathers all relevant information, writes the test evaluation report, and distributes it to all stakeholders in a formal way according to company guidelines.

- For each project, the evaluation of the test project is organized. Lessons-learned meetings with the test team and the stakeholders take place. Practical details are written for the test archive.
- Various types of project information are gathered: project test plan, detailed test plan, test status reports, and test summary reports.
- A test evaluation report with the consolidation of the lessons-learned meetings is written. A template from the test strategy is used.
- Formal distribution and archiving of the test evaluation report is set up according to company policy and formal approval is received.

Agile Testing Add-ons

Project Retrospective

The project retrospective is an important part of the project closure. It is a retrospective that concentrates on a complete project or release. It is useful to use the Agile manifesto as a starting point and to reflect on the team's performance on the main principles and identify possible improvements. This will lead to valuable insight and improvement to the overall process. In comparison to the sprint retrospective that is only meant for the team, in this meeting, the project manager, test manager, or other stakeholders are involved.

The goal and scope of a project retrospective differs slightly from a sprint retrospective. During the project retrospective, the attendees will address the following points:

- Look back at the whole project/release and talk about what went well, what didn't go well, and what they would like to improve on in the next release/project.
- Analyze the outcome and improvements defined during the different sprint retrospectives.
- Evaluate definition of done and definition of ready and, if necessary, adapt them for upcoming projects.
- Analyze the evolution of the velocity and check if there is any room for changes to the organizational structure, improving future productivity.

All suggested improvement actions are discussed and their added value is evaluated. The project team finally decides which of the defined actions will be applied to future projects. All of the improvement actions identified during the project retrospective must be addressed during the first iteration of a new project/release.

Evaluation Report

The added value of an evaluation report is to learn from the past and to improve future project success and quality. It is less valuable to focus too much on statistics, like number of executed test cases or found defects. It is more interesting to summarize the delivered items, the definition of done that was applied to ensure high-quality working software, and improvements defined during the project retrospective. The project closure takes all aspects of the project into account and does not only focus on testing.

From a testing point of view, the following information is shared with the stakeholders:

- Overview of delivered product backlog items
- Open defects
- Quality of automated unit tests
- Quality of automated regression tests
- Applied definition of done
- Improvement actions

Mobile Application Testing Add-ons

Quality

At the end of the test project, a final evaluation of the quality of the mobile application under test should be given. This final evaluation can be supported by traditional metrics like test specification coverage, test execution coverage, number of test cases passed/failed, number of open defects, etc. It is also important to have a more qualitative evaluation of the mobile application under test by addressing quality attributes like functionality, usability, performance, etc.

Based on this final evaluation, a comparison can be made with initial expectations. These are expressed as acceptance criteria or as the definition of done, depending on the development methodology that is being applied. This way, formal acceptance advice to relevant stakeholders can be determined. Associated risks, workarounds, and mitigation actions should be reported as well.

Lessons Learned

Mobile application development is very demanding. Due to fierce market competition and intense innovation, development cycles usually are very short. This is reflected in testing as well. Testing has to keep up with the pace without compromising application quality. As a result, all testing activities should be as efficient and as effective as possible, and proper support should be available at all times. During the project, continuous improvement should be a concern for everyone involved.

Additionally, lessons learned in the project should also be lifted across project borders. How can future projects profit from the lessons learned during the project that is about to be closed? This is a key question to be answered during evaluation of the test project. Different topics can be skimmed when looking for lessons learned:

- Test process
- Test approach
- Test environment
- Test team

The following questions can be helpful:

Lessons Learned	
Test Process	• Was the test process aligned with the software development life cycle? • Was the test process clear for all stakeholders in the project? • How were defects reported/managed? • How was test progress reported/managed?
Test Approach	• Did we cover the right devices/operating systems/network connections? • Did we cover the right test levels? • Did we cover the right test types/quality attributes? • Were the testing priorities clear? • Was the effort spent on the right things/activities? • What test (design) techniques were used? • Was the test approach flexible enough to cope with project changes (scope/planning/budget)?
Test Environment	• Was the test environment available as required/expected? • Were there tests that could not be executed due to missing parts in the test environment? • Did we have the right tools available? • Did the tools perform as expected? • Was test data available as required/expected?
Test Team	• Were the right skill set and competences available in the team? • Was the test team co-located (shared workspace, same office building) or geographically distributed (different office buildings, different towns or countries, different time zones, etc.)? • How did the test team cooperate? Was communication between test team members effective and efficient? • How did the test team cooperate/interact with analysts, developers, other stakeholders, etc.?

4. Test Project Closure (TPC)

4.2 Consolidate Test Deliverables

Goals

"Consolidate Test Deliverables" is a process in which all test deliverables are gathered and archived in a controlled way for each test project, and is described in the test strategy of the test project. This includes a formal handover of the test deliverables.

Control over the testing process of **consolidating test deliverables for the respective testing project** that satisfies the business requirement for testing of **collecting and archiving test deliverables for purposes of future reuse and compliance with relevant regulation frameworks and quality standards** by focusing on:

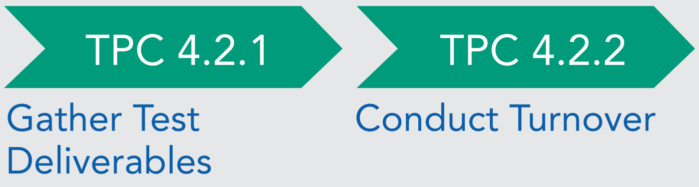
- Collecting all test deliverables
- Identifying the reusable test deliverables
- Identifying the non-reusable test deliverables
- Archiving selected test deliverables in an agreed location
- Organizing the actual handover of selected test deliverables to the test object's owner

and is achieved by

- Gathering test deliverables
- Conducting turnover of test deliverables to the test object's owner

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.



› TPC 4.2.1 Gather Test Deliverables

For the test project, all present test deliverables are gathered during this step.

› TPC 4.2.2 Conduct Turnover

The gathered test deliverables are handed over to the future application owner.

Maturity

The testing activity **"Consolidate Test Deliverables"** that satisfies the business requirement of **collecting and archiving test deliverables for purposes of future reuse and compliance with relevant regulation frameworks and quality standards** is at:

Initial level

The testing activity is executed in an uncontrolled and chaotic way.

- The activity is not organized for all projects.
- If test deliverables are gathered, it takes place in a chaotic and uncontrolled way.

Starter level

The test manager takes control of the "Consolidate Test Deliverables" testing activity, consolidates the test deliverables, and conducts a turnover.

- Test deliverables are gathered and archived for each project.
- There is no distinction made between reusable and non-reusable deliverables.
- An informal handover is set up to inform the stakeholders and the project manager about the repository.

Experienced level

The test manager takes control of the "Consolidate Test Deliverables" testing activity, consolidates the test deliverables, and archives them in a central project location. There's a formal handover in which principal stakeholders and the project manager are consulted.

- Test deliverables are gathered and archived for each project.
- Reusable deliverables and non-reusable deliverables are defined and stored separately in one project repository.
- Company guidelines for archiving have been met.
- During a formal meeting with the project manager, the test archive guidelines and repository are handed over.

Master Level

The test manager takes control of the "Consolidate Test Deliverables" testing activity, consolidates the test deliverables, and archives them in a central corporate location. There is a formal handover in which all of the stakeholders and the project manager are consulted.

- Test deliverables are gathered and archived for each project.
- A corporate repository has been created and communicated.
- Reusable deliverables and non-reusable deliverables are defined and stored separately in one corporate repository.
- Quality management has been consulted for approval; the process of archiving and setting up of the guidelines has been performed according to the test strategy.
- A formal meeting takes place with the project manager and all stakeholders involved to hand over the test archive guidelines and repository.

Agile Testing Add-ons

Project Closure

The main goal of Agile project closure is to hand over the product to the operational team. The project will be closed when all product backlog items in scope have been developed, accepted by the product owner, and handed over to the operational team. In Agile terms, this means that only features that have met the definition of done are ready for delivery and are shippable. All other items will either be archived or planned for an upcoming release.

Automated Test Scripts

The Agile manifesto states, "Working software over comprehensive documentation." Generally, Agile seeks to minimize waste, which also applies to documentation. This is reflected in the test deliverables that are being consolidated at the end of the project. The main focus is on ensuring that the future application owner has a good overview of the delivered features and quality.

In a world of continuous integration, deployment, and delivery, it is quite obvious that from a testing point of view, reusable test deliverables mainly consist of automated test scripts. These are used as living documentation and are part of the regression test set to ensure that the quality of the product remains stable during the maintenance phase. This requires the transfer of knowledge concerning the setup, monitoring, and creation of automated builds.

The most efficient and effective method of conveying information is face-to-face conversations, as described in the Agile principles. To ensure that the operational team is up to speed and familiar with the product when the system is handed over to them, it is important that they already participate in the review meetings at the end of each iteration. This way, they already gain the knowledge and are less dependent on documentation that might already be outdated when the product is shipped to production.

Mobile Application Testing Add-ons

Automated Test Scripts

Chances are very high that test cases need to be repeated many times, not only as a consequence of incremental/iterative development cycles that are often applied when developing mobile applications, but also due to the many platforms (device, operating system, network connection) that need to be tested.

In this context, automated test scripts are the first and most important test deliverables to be consolidated and handed over to the application owner for future use. Automated test scripts are used as living documentation and are part of the regression test set that will help ensure the quality of the mobile application during consecutive releases.

Test Environment

Test environment management is usually a very challenging activity when developing mobile applications. Therefore, future application owners will also welcome all test deliverables related to this subject. When feasible, the entire test environment will be handed over.

This includes:

- Hardware
  - Devices
  - Cables
  - SD cards
  - Touch pen
  - Etc.
- Installed software
  - Operating system
  - Mobile application
  - Monitoring tools
  - Simulation tools
  - Etc.
- Related documentation
  - Test environment requirements
  - Inventory list
  - Installation procedures
  - Resetting procedures
  - Contracts with external suppliers (SLA/KPI)
  - Network setup
  - Etc.

5. Test Management (TM)

5.1 Staff and Manage Test Team

Goals

During the "Staff and Manage Test Team" activity, the test manager ensures the test team's availability and performance.

Depending on the test project's needs (i.e., defined roles and responsibilities), the test manager will assemble the test team. Team members with the right set of skills, experience, and attitude are selected. In doing so, the test manager will not only consider the individual's ability to perform, but also the team's ability to perform. Building a good and performant team is more than just bringing individuals together.

If needed, the test manager will organize proper training for the team members. Test team members should have the opportunity to gain or add to their knowledge and experience required to execute the responsibilities assigned to them.

Once the test team has been assembled, the team members' technical capabilities, commitment, and motivation are continuously monitored and evaluated. If needed, appropriate measures are taken by the test manager.

This is an ongoing activity throughout the complete test project's life cycle.

Control over the testing process of **staffing and managing the test team to ensure the test team's availability and performance** that satisfies the business requirement for testing of **wanting to have the test project tested in the best way possible; making sure that people are fully trained to do the job in the best way possible; and making sure that a lack of resources will not slow down the project** by focusing on:

- The number of resources needed according to the test plan
- The skill set needed according to the test plan
- The evolution and availability of the test team

and is achieved by

- Selecting the test team members at the beginning of the project
- Training the team members if needed
- Managing the test team
- Releasing the test team at the end of the project

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

TM 5.1.1	TM 5.1.2	TM 5.1.3	TM 5.1.4
Select Test Team	Train Test Team	Manage Test Team	Release Test Team

TM 5.1.1 Select Test Team

At the beginning of the project, the test manager builds the test team needed to complete the project. In the worst case scenario, he assigns the most appropriate roles to the people assigned to him. In the best case scenario, he can select the best team member for each role.

TM 5.1.2 Train Test Team

If needed, the test manager will make sure that the test team is trained for the task at the beginning of or during the project.

TM 5.1.3 Manage Test Team

During the project, it is possible that team members will need to be added, replaced, or removed due to external (budget, time, scope, etc.) or internal (resignation, sickness, etc.) factors. The test manager anticipates or reacts to those factors and manages the team accordingly.

TM 5.1.4 Release Test Team

At the end of the project, the test team is released. This is done after the handover is complete and is usually done informally.

Maturity

The testing activity **"Staff and Manage Test Team"** that satisfies the business requirement of **wanting to have the test project tested in the best way possible; making sure that people are fully trained to do the job in the best way possible; and making sure that a lack of resources will not slow down the project** is at:

Initial level

The "Staff and Manage Test Team" testing activity is executed in an uncontrolled and chaotic way.

- For some projects, there is a test team and a test manager; for other projects, the tester acts as a test manager in addition to his testing responsibilities.

Starter level

The test manager gets a team assigned, tries to assign the correct role to each team member, and manages the test team in a reactive way.

- The test manager gets assigned a test team.
- Training is given "on the go" and when needed.
- Management of test team is done" on the go" and when needed.
- The test team is managed in a reactive way.

Experienced level

The test manager can select his test team at the beginning of the project. He can select the test team members internally from a pool of testers, but it's possible that he can even hire externally. The test manager can identify the training needs up front and reacts accordingly. Management of the test team is done proactively.

- The test manager selects (test) staff that can take on the defined test project roles—selection can happen both internally and externally (hiring extra/ temporary resources).
- The test manager identifies training needs for test team members.
- The test manager arranges a training program for the test team members.
- On a regular basis, the test plan and the planning are evaluated, and actions to manage the test team are taken.
- The test team is managed in a proactive way.

Master Level

The test manager staffs his team, taking into account the level of independence wanted. Training needs are identified up front and the test manager acts accordingly. The test manager manages his team with a risk-based strategy.

- Depending on the level of independence wanted (dictated by the test approach) and the level of centralization of functions (dictated by the company's test policy), the test manager will make agreements with principal stakeholders, line managers, and/or external parties to staff the test team.
- The test manager identifies training needs for test team members.
- The test manager arranges a training program for the test team members.
- The test team is managed by taking a project risk-based strategy into account.

Agile Testing Add-ons

Staffing

Before diving deep into the iterations, it is important to determine the number of testers and/or (domain) specialists needed during the release. Therefore, it is important to have a good understanding of the testing types and test effort required. Plan ahead to make sure the needed resources are allocated for the period of the project.

The amount and type of people in a testing role and the time needed to prepare the testing depend on several factors:

- Maturity level of the testers
- Maturity level of the team
- Complexity of the system under test
- Number of test types in the scope of the project
- Maturity level of test automation
- Number of test cases that are automated
- Product and project risks
- Availability of the testers
- Agile maturity level of the team

For some iterations and stories, especially when faced with non-functional aspects of the project such as performance, usability, acceptance, or security testing, it might be necessary to bring in outside testing roles to temporarily support the team.

Since the allocation of the test resources must be done in close cooperation with other Agile teams and the project manager, the Agile test plan is a good place to document which test resources are part of the project, and which and when outside resources will be added to the project.

Skills

Test-driven development and acceptance-test-driven development are often applied by Agile teams. Therefore, it will be important to recruit resources, not mandatory testers, who have the appropriate skills and experience. Next to technical and business skills, it is important to have the adequate soft skills to achieve the goal by closely collaborating with each other to deliver business value.

Below are some of the key traits that are important for an Agile tester:

- Open mindset
- Commitment
- Results-oriented
- Provide continuous feedback
- Close collaboration and communication with team members and stakeholders
- Practice continuous improvement
- Respond to change
- Self-organizing

One of the strengths of a tester is knowledge on both testing and business levels of what needs to be delivered. Paired with the requirements specified by the product owner, this combination makes a strong team that is able to describe exactly what is expected and what is not. Testers are also in an excellent position to critique and question requirements if they are unclear or ambiguous.

Training Needs

One way to satisfy training needs is by organizing formal training. The test manager plays an important role in providing input about the content and duration of training and assessing the impact of the training (needs) on the velocity of the team. This will be important to take into account once the team has started iterative development to ensure that time for training is allocated and taken into account when setting velocity and sprint targets in order to avoid having uncompleted stories at the end of a sprint.

Another way to satisfy the training needs is through on-the-job coaching. For this, pair testing can be applied. Pair testing is one of the most common approaches used in Agile environments to share knowledge and get new or less experienced team members up to speed.

Test Manager

A main difference between classical and Agile projects is that in an Agile environment, testers are part of a self-organizing team and are less dependent on a test manager or team lead. It is a team's responsibility to complete the committed work, identify impediments or bottlenecks, and escalate these to the Scrum master.

The role of the test manager is less focused on managing the testers; the role is changing to a facilitating and consulting role, including following responsibilities:

- Function as the communication chain to senior management
- Increase the control on an Agile project and provide vision
- Provide guidance and leadership for both new and experienced testers and specialists

5. Test Management (TM)

5.2 Monitor and Adjust Test Plan

Goals

“Monitor and Adjust Test Plan” is a recurrent process during the project life cycle, which makes it possible to manage the test project and have a view of the possible impacts on the testing activities.

Control over the testing process of **monitoring and adjusting the test plan for the respective testing project** that satisfies the business requirement for testing of **being responsive to changes in the test project’s context and adjusting the test project governance in order to deliver agreed-upon test deliverables within time and budget and according to the agreed level of quality** by focusing on:

- Continuously monitoring and re-evaluating the test project’s context
- Adjusting the test project’s scope according to the changed project context
- Adjusting the test project’s test approach matrix (features to test, priorities, test techniques, etc.)
- Re-estimating the workload associated with the adjusted test project’s scope and test approach
- Adjusting the test project’s organization according to the adjusted test project’s scope, test approach, and associated workload
- Adjusting the test project’s schedule according to the adjusted test project’s scope, test approach, and associated workload

and is achieved by

- Monitoring and adjusting the project context
- Monitoring and adjusting the test scope
- Monitoring and adjusting the workload
- Monitoring and adjusting the test organization
- Monitoring and adjusting the test schedule

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

TM 5.2.1	TM 5.2.2	TM 5.2.3	TM 5.2.4	TM 5.2.5
Monitor and Adjust Project Context	Monitor and Adjust Test Scope	Monitor and Adjust Workload	Monitor and Adjust Test Organization	Monitor and Adjust Test Schedule

TM 5.2.1 Monitor and Adjust Project Context

The project context is verified to ensure that the test plan always has the correct project scope.

TM 5.2.2 Monitor and Adjust Test Scope

The test scope will be updated if necessary. Think about updating test levels, test types, test items, and the out-of-scope items.

TM 5.2.3 Monitor and Adjust Workload

Once testing has started, the workload is monitored, the actual and planned workloads are compared, and re-estimations can be done.

TM 5.2.4 Monitor and Adjust Test Organization

The test organization is monitored to ensure that the test team is still capable of executing all tasks as described within the test plan.

TM 5.2.5 Monitor and Adjust Test Schedule

The team ensures that the test schedule is still up-to-date. A comparison between the actual and planned test schedules, as well a comparison between the actual and planned budgets, is performed.

Maturity

The testing activity **"Monitor and Adjust Test Plan"** that satisfies the business requirement of **being responsive to changes in the test project's context and adjusting the test project governance in order to deliver agreed-upon test deliverables within time and budget and according to the agreed level of quality** is at:

Initial level

The "Monitor and Adjust Test Plan" testing activity is executed in an uncontrolled and chaotic way.

- The test plan is not monitored and adjusted for all projects.
- If the test plan is monitored and adjusted for a project, it is done in a chaotic and ad hoc way.

Starter level

The test manager takes control of the "Monitor and Adjust Test Plan" testing activity and decides when and if a test plan needs to be adjusted. The test manager can consult the principal stakeholders, but it's not mandatory.

- For each project, the test plan is monitored and adjusted.
- This activity is done in an ad hoc way whenever changes are needed.
- The test manager decides when and if the test plan needs to be adjusted.
- Principal stakeholders might be consulted in this activity, but they don't take an active role.
- Once the test plan has been adjusted, a notification can be sent to the principal stakeholders.

Experienced level

The test manager takes control of the "Monitor and Adjust Test Plan" testing activity and performs a regular activity to adjust the test plan. This activity happens in a controlled way and principal stakeholders are informed in the case of changes.

- For each project, the test plan is monitored in a controlled way.
- The test manager performs a regular activity that monitors the test plan.
- If changes are needed, principal stakeholders are informed and together with the test manager, they decide what changes are needed.
- Once the changes are decided upon, the test manager adjusts the test plan (or test plans, one for each test level).
- Once the test plan is adjusted, the principal stakeholders receive a notification.

Master level

The test manager and all stakeholders take control of the "Monitor and Adjust Test Plan" testing activity; a regular monitoring activity is set up. This activity happens in a controlled way, all stakeholders are informed, and a meeting is organized to discuss the necessary actions to be taken.

- For each project and test plan, a monitoring activity is set up with the test manager and all stakeholders.
- If an action is needed to adjust the test plan, all stakeholders are informed that action is needed and a meeting is planned to discuss what action needs to be taken.
- The necessary action is agreed upon by all stakeholders involved and all other stakeholders are informed about the action to be taken.
- The test manager adjusts the test plan to reflect the action taken.
- Once the test plan is adjusted, all stakeholders receive a notification.
- The test plan is placed under version management and with each update, the version is adapted.

Agile Testing Add-ons

Adjust Test Scope

One of the core Agile values is a quick response to change. Short feedback loops and close cooperation with the customer can result in changes to the project scope that is monitored and updated by the product owner. Such a scope change can also possibly impact the prioritization of the features in the product backlog.

In Agile projects, it is the team's responsibility to monitor the progress of committed activities. The work to be done is often visualized using a task, Scrum, or Kanban board, and followed up on during the daily stand-up meeting. The remaining workload is made visible by a release burndown/up chart or a dashboard keeping track of remaining epics or user stories.

Test Team

Normally the team monitors the amount of completed features and adjusts the velocity for upcoming sprints in order to meet definition of done for all features of the sprint backlog at all times. However, when for some reason, in a particular sprint, testing does become a bottleneck and features cannot be completely tested anymore, other team members will support the testers in completing testing activities rather than adding additional test resources to the team. Only in the event that there is no other option, the team will be scaled up to increase the number of resources to complete the work.

5. Test Management (TM)

5.3 Manage Anomalies

Goals

This activity specifies how anomalies are handled during the project. Anomalies can be defects, events, issues or changes:

- Defects are differences between expected and actual results of the system under test detected during the Test Execution phase.
- Events are situations during the project which can cause a delay if no appropriate actions are taken.
- An issue is a situation occurring during the test process which will definitely cause a delay.
- A change can occur during the test project lifetime and will result in an alternation of the initial test plan.

Control over the testing process of **managing all anomalies observed during the test project to a closure** that satisfies the business requirement for testing of **resolving all anomalies that might impact the quality of the test object in particular, and the test project in general,** by focusing on:

- Gaining knowledge about the content of the detected anomaly
- Assigning the anomaly to the correct stakeholder in order to have it investigated
- Defining the solution of the anomaly
- Implementing the solution of the anomaly
- Verifying and closing the anomaly

and is achieved by

- Identifying the anomaly
- Investigating the anomaly
- Determining a solution to the anomaly
- Resolving the anomaly
- Closing the anomaly

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

TM 5.3.1 Identify Anomaly

TM 5.3.2 Investigate Anomaly

TM 5.3.3 Determine Solution to Anomaly

TM 5.3.4 Resolve Anomaly

TM 5.3.5 Close Anomaly

TM 5.3.1 Identify Anomaly

Once an anomaly is detected in the test project, the first step is to make sure it is correctly defined. Information about the anomaly is gathered, its impact on the project and/or product is defined, and the anomaly is logged. All stakeholders involved in the project can and should identify the anomalies they are detecting.

TM 5.3.2 Investigate Anomaly

The type of anomaly is determined and a responsible stakeholder is assigned to the anomaly. The root cause of the problem is defined, together with an impact analysis which defines the priority of the anomaly. Alternative solutions and/or workarounds are determined to mitigate the anomaly. Additional documentation is added to the anomaly that was logged earlier.

TM 5.3.3 Determine Solution to Anomaly

The final solution to the anomaly is defined in this step. The solution is approved and the anomaly which was logged earlier is updated with information on what the solution will be, what the effort is, when it can be implemented, what the cost is, and what the potential risk is.

TM 5.3.4 Resolve Anomaly

The defined solution is implemented. Any updates about the anomaly on the level of actual effort, schedule, cost, and risk are documented with the anomaly that was logged earlier.

TM 5.3.5 Close Anomaly

The implemented solution is verified, resulting in two different possibilities: the solution is accepted or not accepted. In case of acceptance, the anomaly is closed. In case of a non-acceptance, the anomaly remains open and steps 2, 3, 4, and 5 are executed again. The verification result is documented with the anomaly that was logged earlier, along with the close date and close type.

Maturity

The testing activity **"Manage Anomalies"** that satisfies the business requirement of **resolving all anomalies that might impact the quality of the test object in particular, and the test project in general** is at:

Initial level

The "Manage Anomalies" testing activity is executed in an uncontrolled and chaotic way.

- Only defects are taken into account.
- Every defect is at least tracked by email for a project.
- Solutions of defects are not always verified by a retest.

Starter level

All types of anomalies (defects, events, issues, and changes) are managed. When deemed necessary/useful, relevant principal stakeholders are consulted during the process.

- All anomaly types are managed to a final closure following a predefined life cycle, which is uniform for the whole project.
- Anomalies are tracked in a project repository.
- Occasional meetings are organized to discuss the solution of particular anomalies with relevant (based on their knowledge and experience) principal stakeholders.
- A final solution is applied to every relevant anomaly.
- The solution of an anomaly is always verified.

Experienced Level

All types of anomalies (defects, events, issues, and changes) are managed. During the process, the principal stakeholders are consulted on a regular basis.

- All anomaly types are managed to a final closure following a predefined life cycle, which is uniform for the whole project.
- Anomalies are tracked in a project repository.
- Regular meetings are organized to discuss alternative solutions for every relevant anomaly with the principal stakeholders.
- Different possible solutions to solve an anomaly are considered and evaluated.
- A final solution is applied to every relevant anomaly.
- The solution of an anomaly is always verified.

Master level

All types of anomalies (defects, events, issues, and changes) are managed. During the process, all stakeholders are consulted on a regular basis.

- All anomaly types are managed to a final closure following a predefined life cycle, which is uniform for the whole organization.
- A life cycle to manage anomalies is documented in the corporate test strategy or test policy.
- Anomalies are tracked in a corporate repository.
- Regular meetings are organized to discuss alternative solutions for every relevant anomaly with all principal stakeholders.
- Different possible solutions to solve an anomaly are considered and evaluated. A detailed cost/benefit evaluation for each solution is conducted.
- A final solution is applied to every relevant anomaly.
- The solution of an anomaly is always planned and verified.
- A causal analysis study is conducted in order to find new related anomalies and/ or to put preventive measures in place.

Agile Testing Add-ons

Defects

It is best to agree upon the defect management approach in the initial planning phase.

Usually, defects related to new functionality have to be fixed during the sprint to fulfill definition of done. Most teams do have a "zero tolerance," meaning that all known bugs need to be solved in order to get stories accepted. The main reason for this is to keep the code maintainable and avoid technical debt.

It is not uncommon that new defects are not immediately entered in a defect tracking tool. Instead, testers might address defects to the developers immediately. Together, they deal with them quickly and effectively and work together to reproduce and solve the defect. Defects that can be solved within the same sprint will not be entered in the project's defect tracking tool, while those that cannot be solved in time will be.

The decision to investigate and solve defects is based on the priority of related user stories. This means that blocking defects concerning lower-prioritized user stories may get lower priority and have to wait until the more important features have been completed. Priorities are determined together with the product owner.

In the event that the team has agreed with the product owner that low-priority defects will not be solved in the same sprint, or a defect has been spotted after a story was accepted, the team needs to make that visible and add the defect to the product backlog. The decision regarding if and when (what sprint) a defect will be fixed depends on the business value and the priority of the defect.

In order to avoid regression, automated tests are often created to cover identified defects. These tests are added to the continuous integration framework to ensure that the code doesn't break in the future.

Impediments

Impediments are events or issues that can cause a delay or prevent Agile teams from completing their stories, impacting the team's velocity. Therefore, impediments should be raised as early as possible. Proper understanding of the impact on the team's productivity is very important.

It can be useful to differentiate between impediments that slow down the team and those that completely block progress on features. Some examples are:

- Sickness of team members
- Unforeseen changes in team composition
- Issues with tools
- Unavailability of hardware
- Technical debt
- Missing dependencies
- Miscommunication
- Unavailability of the product owner

First, the Scrum master visualizes the impediments on an impediment board to improve transparency. The Scrum master follows up to ensure the impediment gets resolved, but it is the whole team's responsibility to address the impediment. Before taking action, it can be useful to analyze the root cause using tools like the 5 Whys or the Fishbone diagram.

Ideally, the Scrum team can remove any impediment itself, but this is not always the case. When this happens, the Scrum master will find an external party that can help to remove the impediment.

The impediment removal process should not stop when the impediment has been successfully removed. The team should practice continuous improvement and learn from the impediments and define actions to prevent similar impediments in the future. Especially when dealing with reoccurring impediments, the team should discuss how such issues can be avoided in the future and define action points for the upcoming iterations during the retrospective meeting.

Changes

Scope creep is handled differently in Agile projects than in traditional projects. As described in the Agile manifesto, it is more important to respond to change and to collaborate with the customer instead of following a plan and negotiating contracts.

So, if a change request is identified by one of the team members or stakeholders during the sprint or review meeting, the change will be analyzed and discussed with the product owner or customer.

If the change has added value to customer, the feature is added to product backlog, prioritized, and estimated by the team, taking all testing effort into account.

5. Test Management (TM)

5.4 Reporting

Goals

During this activity, the test manager will provide regular feedback to the test project's stakeholders. Depending on the stakeholder's information needs, different reports will be created and communicated.

Control over the testing process of **reporting on the test project** that satisfies the business requirement for testing of **wanting to have a clear view of the progress of the test process; the planning versus the actuals; the number of anomalies; and the actions to be taken to keep the project in line with the test plan** by focusing on:

- The evolution of the anomalies
- The actual progress of the project versus the planned progress
- Regular meetings with the test team to discuss the reported anomalies and the progress made

and is achieved by

- Gathering the anomalies
- Gathering the test progress
- Analyzing the gathered data
- Writing the test reports
- Distributing the test reports

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

TM 5.4.1 Gather Anomalies › TM 5.4.2 Gather Test Progress › TM 5.4.3 Analyze the Gathered Data › TM 5.4.4 Write Test Report › TM 5.4.5 Distribute Test Report

TM 5.4.1 Gather Anomalies

The test manager will gather all relevant anomalies from the anomaly database, sort them by priority, and verify that all anomalies are updated to the latest status.

TM 5.4.2 Gather Test Progress

The test manager will gather the actual progress made on the different planned testing activities based on the test cases repository, the test execution log, meetings with test team staff, etc. The number of test cases that have been created and executed since the last test reporting are counted and collected. Also, test cases still to be created/executed are estimated.

TM 5.4.3 Analyze the Gathered Data

In order to start writing the test report, the test manager has to analyze the data gathered in the first two steps. By applying dedicated metrics, the data gathered is now transferred into information worth reporting to the different stakeholders (e.g., test preparation coverage, test execution coverage, traceability, etc.). The evolution of the anomalies and the evolution of the test progress will be reported and future actions will be determined.

TM 5.4.4 Write Test Report

Following the project or company guidelines or templates for the respective test reports (e.g., test status report, test summary report), the test manager writes the test report based on the conclusions made in step 3.

TM 5.4.5 Distribute Test Report

After the test report is written, the test manager distributes the test report to the stakeholders involved.

Maturity

The testing activity **"Reporting"** that satisfies the business requirement of **wanting to have a clear view of the progress of the test process; the planning versus the actuals; the number of anomalies; and the actions to be taken to keep the project in line with the test plan** is at:

Initial level

The "Reporting" testing activity is executed in an uncontrolled and chaotic way.

- There is some test reporting, but it is informal, infrequent, and limited to defects and progress made.
- Not all interested stakeholders are reported to.
- The test report is sent by mail or by oral communication in an infrequent way.

Starter level

The test manager writes a test report for each test project and distributes it to principal stakeholders on an infrequent or on-demand basis

- Defects and anomalies are gathered from the tracking tool.
- Test progress versus planned progress is gathered via informal meetings with the test team and the test management tool.
- The test report in is writing, but not in a fixed format.
- The test report is sent infrequently to the principal stakeholders.

Experienced level

The test manager writes a test report for each test project and distributes it to principal stakeholders on a planned basis.

- Defects and anomalies are gathered from the tracking tool.
- Test progress versus planned progress is gathered via formal and frequently-planned meetings with the test team.
- The test report is in writing and in a fixed template described in the test strategy.
- The test report is sent to all principal stakeholders on a frequent and planned basis.

Master level

The test manager writes a test report for each test project and distributes it to all stakeholders on a planned basis.

- Defects and anomalies are gathered from the tracking tool.
- Test progress is gathered in an automatic way (with the use of a test management tool).
- The test report is in writing and in a fixed template described in the test policy.
- The test report is sent to all stakeholders on a frequent and planned basis.

Agile Testing Add-ons

Agile Reporting

Following the Agile principles, it is more important to focus on open communication and working software. Stakeholders are mainly interested in an overview of features that are ready to be delivered and not in stories that haven't been completely developed or tested. In summary, only "done" counts in Agile projects.

Hence, there is less need for detailed and extended reports mentioning metrics like the number of test cases created, the number of test cases executed, or the number of found issues during sprints. Instead, it is sufficient to provide a dashboard to the stakeholders containing the following information:

- Burndown charts on sprint and release level
- Completed features/user stories
- Change requests added to product backlog
- Open defects
- Impediments
- Team's velocity

Within an Agile context, testing activities are part of the tasks that have to be completed in order to finish features and to deliver working software. Therefore, the reporting is mostly done in terms of user stories of features and value to the customer rather than development or testing activities. As a result, there is no need for a separate test status report. The status of testing is reflected in the progress as a whole.

Daily Stand-up Meeting

In most Agile approaches, teams use daily stand-up meetings to track the progress, impediments, risks, and challenges that arise. The status of user stories and the corresponding (test) activities is made visible on the task board that is constantly kept up-to-date by all team members.

Progress

Progress can be reported on different levels: the status of a specific user story, the status of the current sprint, or the status of the entire release might be points of interest.

- User story dashboards can be used to visualize the features that are completed and the user stories that still need to be delivered.
- Sprint burndown charts reflect the progress during a sprint and are updated daily. The Y axis indicates the remaining effort and the X axis indicates the length of iteration.
- Release burndown charts show implementation progress during the release. This chart is the key information radiator for the project's overall status. It provides answers to questions like:
  - When could release be completed based on previous progress?
  - What progress has been made in previous iterations?
  - Is the velocity of the team sufficient to complete the release on time?

Quality

Also, quality reporting can be done on different levels:

- Results of automated tests give a good overview of the stability of the code and can be shared with stakeholders. Test automation on several levels supports the team in delivering good quality, working software. It is key to apply continuous testing within a more global approach of continuous integration, continuous deployment, and continuous delivery.
- A demo of working software at the end of the sprint is the most efficient and reliable way to report quality. In Scrum, during the review meeting at the end of the sprint, a live demonstration of the developed features is given to the product owner and the stakeholders to show all stories completed by the team and to prove the definition of done is satisfied.

6. Quality Management (QM)

6.1 Monitor Organization

Goals

The test manager or (independent) quality manager will assess the quality of the test organization.

The goal of the activity is to detect possible deficiencies in the test project's organization (e.g., communication between different sub-teams, outsourcing practices, hierarchical structure, level of professionalism, etc.) and to make recommendations to solve these deficiencies.

Implementing the recommendations can also be part of this activity, when all relevant stakeholders agree to do so.

Control over the testing process of **monitoring quality of the test organization** that satisfies the business requirement for testing of **making sure that the test organization that is in place is able to execute the testing activities at hand and in a proper way (properly executing the test process and delivering quality test products)** by focusing on:

- Comparing compliance of the actual test organization with the intended or documented test organization
- Determining and reporting non-compliance (deviations from intended or documented test organization) of the actual test organization
- Taking necessary measures to remediate all reported non-compliance of the project's test organization
- Detecting and reporting shortcomings and insufficiencies of intended or documented test organization
- Taking necessary measures to remediate all reported deficiencies of the intended or documented test organization

and is achieved by

- Assessing organizational quality
- Defining organizational improvements
- Planning organizational improvements
- Implementing organizational improvements
- Verifying organizational improvements

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

QM 6.1.1	QM 6.1.2	QM 6.1.3	QM 6.1.4	QM 6.1.5
Assess Organizational Quality	Define Organizational Improvements	Plan Organizational Improvements	Implement Organizational Improvements	Verify Organizational Improvements

› QM 6.1.1 Assess Organizational Quality

The quality level of the project's test organization is checked based on the intended (not documented) or expected (documented in the project's test plan or the corporate's organizational model) level of quality. This happens on a daily basis and can be considered part of the "Staff and Manage Test Team" activity.

In the event that organizational non-quality is identified, applying a more formal management review or (external) audits can deliver more detailed information on the organizational non-quality. Organizational non-quality can originate from non-compliance of the actual test organization with the intended or expected test organization, or from deficiencies in the intended or expected test organization. Regardless of the origin, all organizational non-quality is reported.

› QM 6.1.2 Define Organizational Improvements

Actions to remediate reported organizational non-quality are defined. At the same time, if not already available from the project's test plan or the corporate's organizational model, metrics that can be used to measure the test organization's quality level will be defined.

› QM 6.1.3 Plan Organizational Improvements

Based on the available resources (time, budget, people), defined actions to remediate organizational non-quality are planned for execution. Also, activities for collecting data measures and calculating defined metrics to measure organizational improvements (before/after) are planned. All organizational non-quality, due to non-conformance with the intended or expected test organization, should be addressed during the test project. Actions to resolve deficiencies of the test organization can be postponed or referred to a higher level of responsibility.

› QM 6.1.4 Implement Organizational Improvements

A "before picture" of the respective organizational quality is created. Planned actions for collecting data measures and calculating "before" metrics are executed. Once the before picture is created, actions remediating respective organizational quality are executed as planned.

› QM 6.1.5 Verify Organizational Improvements

Once the organizational improvements have been implemented, planned actions for collecting data measures and calculating "after" metrics are executed. Based on the before and after metrics, actual organizational improvement is verified. When the improvement proves to be insufficient, the improvement cycle might be repeated.

Maturity

The testing activity **"Monitor Organization"** that satisfies the business requirement of **making sure that the test organization that is in place is able to execute the testing activities at hand in a proper way (properly executing the test process and delivering quality test products)** is at:

Initial level

The "Monitor Organization" testing activity is executed in an uncontrolled and chaotic way.

- The activity is not organized for all projects.
- If the activity is executed, it is executed in a chaotic and ad hoc manner.

Starter level

Determining and improving test organization non-quality is incorporated into the daily activity of managing the test team. The test manager responds to the determined organizational non-quality on an ad hoc basis. The test manager identifies what organizational non-quality is addressed by taking into account feasibility (available resources) and plans test organization improvement actions accordingly.

- For each project, the test organization is monitored.
- The organization monitoring is merely part of the "Staff and Manage Test Team" activity.
- Organization monitoring is executed in an informal way; the test manager reacts to organizational weaknesses that are observed during daily follow-up of the test team.
- It is the test manager who, in the end, decides whether the organizational weakness will be addressed and how this will be done.
- The test manager escalates organizational improvement actions that are beyond the scope of his mandate to the project manager.
- Defined organizational improvement measures are implemented in an ad hoc manner.

Experienced level

Determining and improving test organization non-quality is incorporated into the daily activity of managing the test team. Together with principal stakeholders, the test manager and quality manager identify what organizational non-quality is addressed by taking into account feasibility (available resources) and incorporate organizational improvement actions into the global test project.

- For each project, the test organization is monitored.
- The organization monitoring is part of the "Staff and Manage Test Team" activity.
- When possible organization weaknesses are identified, the test manager can decide to organize a satisfaction survey, a collaborative cooperation scan, or even a management review focusing on the testing organization.
- The test manager and/or quality manager discuss possible organizational improvement measures with principal stakeholders. Together, they decide on the organizational improvements that will be addressed in the ongoing test project (given time, resource, and quality constraints).
- Defined and agreed-upon organizational improvement measures are incorporated into the global test plan (see "Plan Test Project" and "Monitor and Adjust Test Plan" activities).
- Organizational improvement actions that cannot be implemented in the scope of the (current) test project are drafted in a lessons-learned report.
- The impact of organizational improvement actions is monitored.

Master level

Determining and improving test organization non-quality is incorporated into the daily activity of managing the test team. However, external (not organized by the project) audits can be a secondary source of test organization non-quality determination. Together with relevant stakeholders, the test manager and quality manager identify what organizational non-quality is addressed by taking into account feasibility (available resources) and incorporate organizational improvement actions into the global test project.

- For each project, the test organization is monitored.
- The organization monitoring is part of the "Staff and Manage Test Team" activity.
- When possible organization weaknesses are identified, the test manager can decide to organize a satisfaction survey, a collaborative cooperation scan, a management review, or even an (external) audit focusing on the testing organization.
- In addition to daily follow-up and in line with the corporate quality policy, an (external) test process assessment can also be organized.
- The test manager and/or quality manager discuss possible organizational improvement measures with all stakeholders. Together, they decide on the organizational improvements that will be addressed in the ongoing test project (given time, resource, and quality constraints).
- Defined and agreed-upon organizational improvement measures are incorporated into the global test plan (see "Plan Test Project" and "Monitor and Adjust Test Plan" activities).
- Organizational improvement actions that cannot be implemented in the scope of the (current) test project are drafted in a lessons-learned report.
- The impact of organizational improvement actions is measured.

Agile Testing Add-ons

Individuals and Interactions

The first principle of the Agile manifesto states, "Individuals and interactions over processes and tools." This should come as no surprise, since the most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Retrospective

In a self-organizing team, all team members have to monitor the quality, productivity, and efficiency of the team. For example, if the team is not able to accomplish the committed work several times in a row, it should consider if it is due to the current organizational or team structure.

During the retrospective, the team carefully examines the composition of the team on the one hand and the structure of the team on the other hand. First, the team identifies the weaknesses of the team and analyzes the root causes before defining actions. Some possible questions that have to be answered are:

- Is the team's velocity not sufficient?
- Is there any bottleneck?
- Are communication and collaboration as expected?
- Is the team structure slowing down the team's progress?
- Should the team size be adapted?
- Lack of knowledge?
- Does everybody have an open mindset?
- Are technical skills in place to automate test scripts?

Co-located Teams

Having co-located teams is key to improved (face-to-face) communication, teamwork, team spirit, and knowledge sharing. There are several kinds of co-located teams. People may be located in the same building, but in different offices. Others can sit together on the same floor or in the same office, but they are separated by walls. The most productive and efficient constellation of a co-located team is all team members sitting together in a shared workspace where each member of the team is able to see the other team members, including the Scrum master and product owner.

From a testing point of view, the ideal situation is having all of the testers located in the same office next to the product owner and developers. Co-location makes life easier for testers. They can convince the other team members that their contribution adds value to the team much faster. The relationship between the developers and testers improves and doubts about testers' participation are quickly gone. Testers are no longer seen as troublemakers, but as a real part of the team. They all pull together to achieve the goals.

Distributed Teams

Effective collaboration and good communication between team members and stakeholders is key to success. Unfortunately, having co-located teams is not always possible and distributed teams are set up. There are several types of distributed teams. Some teams are split between two locations in the same city or country, some team members might be working from home, while other teams are much more geographically dispersed, even working in different time zones.

Often, one of the main reasons why companies are working with distributed Agile teams is cost reduction. However, the effort and time needed to get a distributed team performing at the same level as co-located teams is often highly underestimated. In order to build a successful team, it is necessary to run through the four stages of the Tuckman[9] model:

- Forming
- Storming
- Norming
- Performing

It can be a challenge to build the relationship and trust between distributed team members. The distance and possible time differences make effective collaboration and face-to-face communication difficult. Due to this, the Forming and Storming stages of a distributed Agile team can take considerably longer in comparison to co-located Agile teams. As a consequence, distributed teams will reach the Norming stage (in which it effectively works together) and the Performing stage

[9] Tuckman, Bruce. "Developmental Sequence in Small Groups." *Psychological Bulletin* Vol 63, no. 6 (June 1965): 384-399. http://dx.doi.org/10.1037/h0022100

(in which the team is functioning completely) much later. But with clear agreements and discipline, it is possible to have distributed teams performing at the same level as co-located teams.

From a testing point of view, the least preferable situation is when the testers are not sitting next to the product owner and developers, as is often the case when testing is outsourced or done offshore. If this happens, testers will be slowed down in their investigation to check out how things are working. Close cooperation with the product owner or developer will be more difficult and challenging due to the distance. Testers will need to put much more effort into communication in order to share knowledge, explain their position, or set the record straight. The strength of a tester—a critical mindset and interaction with all people involved—cannot be used to its fullest advantage anymore.

Improvements

The team is only as strong as the weakest link in the chain. So, it is important to get the weakest link to the same level as the rest of the team. This is achieved by doing an assessment of the skills, strengths, and weaknesses of each individual. Based on this information, the team identifies possible improvement actions to support the weakest team member in order to increase the velocity of the team.

If the team structure is causing an issue, then scaling out, scaling up, reducing team size, or reducing number of teams are some possibilities to improve the organization.

- In the event that there are distributed teams, it might be better to reduce the team size and to establish two smaller, but co-located teams instead of having one large team distributed over several sites.
- In the event that there are co-located teams, it can be more efficient to reduce team size and to have more teams in parallel in order to increase the productivity.
- Another option is to restructure the teams by introducing component, feature, or functional teams.

6. Quality Management (QM)

6.2 Monitor Product

Goals

While the test project as a whole aims at monitoring the quality of the final project deliverables or test object (e.g., application software, procedures, documents, etc.), the goal of the "Monitor Product" activity is to monitor the quality of the intermediate products delivered during the project (e.g., requirement documents, technical design documents, code, test plan, test cases, test reporting, etc.).

The test manager or (independent) quality manager will assess the quality of the intermediate project deliverables and define and implement product improvements in the event that the quality standards set for the intermediate project deliverables are not met.

Control over the testing process of **monitoring quality of the product** that satisfies the business requirement for testing of **ensuring that all intermediate products delivered during the project will have the desired level of quality** by focusing on:

- Comparing quality of the actual intermediate project deliverables with expected or documented acceptance and/or quality criteria
- Determining and reporting product non-quality (deviations of actual quality from expected quality) of actual intermediate project deliverables
- Taking the necessary measures to improve all reported (and confirmed) product non-quality of the project's intermediate deliverables

and is achieved by

- Assessing product quality
- Defining product improvements
- Planning product improvements
- Implementing product improvements
- Verifying product improvements

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

QM 6.2.1	QM 6.2.2	QM 6.2.3	QM 6.2.4	QM 6.2.5
Assess Product Quality	Define Product Improvements	Plan Product Improvements	Implement Product Improvements	Verify Product Improvements

QM 6.2.1 Assess Product Quality

The quality level of intermediate project deliverables is assessed based on the expected level (documented as acceptance/quality criteria or not) of quality using review techniques. Product non-quality (discrepancy between actual and expected product quality) is reported and will be considered and treated as any other anomaly detected during the (test) project (see "Manage Anomalies" activity). The focus will be both on the intermediate project deliverables that serve as the test project's test basis, and on the test project's deliverables.

QM 6.2.2 Define Product Improvements

Actions to remediate reported product non-quality are defined. At the same time, if not already available from the project's test approach or the corporate's test strategy, metrics that can be used to measure the product quality level will be defined.

QM 6.2.3 Plan Product Improvements

Defined actions to remediate product non-quality are planned for execution. In addition to this, activities for collecting data measures and calculating defined metrics to measure product improvements (before/after) are planned.

QM 6.2.4 Implement Product Improvements

A "before picture" of the respective product quality is created. Planned actions for collecting data measures and calculating "before" metrics are executed. Once the before picture is created, actions for remediating respective product quality are executed as planned.

QM 6.2.5 Verify Product Improvements

Once the product improvements have been implemented, planned actions for collecting data measures and calculating "after" metrics are executed. Based on the before and after metrics, the actual product improvement is verified. When the improvements prove to be insufficient, the improvement cycle might be repeated.

Maturity

The testing activity **"Monitor Product"** that satisfies the business requirement of **ensuring that all intermediate products delivered during the project will have the desired level of quality** is at:

Initial level

The "Monitor Product" testing activity is executed in an uncontrolled and chaotic way.

- The activity is not organized for all projects.
- If the activity is executed, it is executed in a chaotic and ad hoc manner.

Starter level

The test manager selects part of the project's test deliverables for review. Principal stakeholders informally review the selected deliverables and address product non-quality as deemed necessary.

- For each project, the main test deliverables are reviewed by principal stakeholders, e.g.,
  - The test plan is reviewed by principal stakeholders.
  - The test basis is reviewed by the test team.
  - The test design is reviewed by the test manager, developers, or analysts.
- Informal reviewing techniques are used to identify product non-quality.
- Original authors decide what product non-quality will be addressed.
- Original authors define the improvement actions to be implemented.
- There is no formal follow-up process that addresses the final closure of the identified product non-quality.

Experienced level

Principal stakeholders review intermediate project deliverables as defined in the (test) project's review plan that is created by the test manager or a quality manager dedicated to the project. All defined product non-quality is treated through the "Manage Anomalies" activity.

- For each project, the test manager or quality manager (dedicated to the project) will draw up a review plan indicating quality gates at which the project's intermediate deliverables will be reviewed, by whom, and what techniques should be used to do so.
- Reviewing techniques can range from informal reviews, to walkthroughs, to technical reviews.
- Formal acceptance or quality criteria are defined for each of the project's intermediate deliverables that should be reviewed.
- Identified product non-quality is logged in the project's anomalies repository.
- Identified product non-quality is managed to a final closure through the "Manage Anomalies" activity.
- The impact of the product improvement actions is monitored.

Master level

All stakeholders review intermediate project deliverables as defined in the project's test approach that is created by the test manager (with the help of the quality manager of the test project). Also, external (not organized by the project) audits can add to the detection of product non-quality. All defined product non-quality is treated through the "Manage Anomalies" activity.

- For each project, quality gates and corresponding reviews of the project's intermediate deliverables are part of the global project's test approach. The approach doesn't differentiate between intermediate and end products.
- All types of reviewing techniques (informal review, walkthrough, technical review, inspection, expert opinion, demo, witnessing) are applied.

- Formal acceptance or quality criteria are defined for each of the test deliverables that should be reviewed.
- Identified product weaknesses are added to the corporate anomalies repository.
- Identified product weaknesses are managed to final closure through the "Manage Anomalies" activity.
- The impact of the product improvement actions is measured.

Agile Testing Add-ons

Product Quality Assessment

The first principle described in the Agile manifesto is, "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."

The expected quality level is defined up front and formalized in the definition of done on the feature, iteration, and release levels. The definition of done is reviewed and, if needed, adapted in order to improve the quality on a regular basis.

Incremental and iterative development allows the team members to improve constantly and to deliver the expected quality. The quality of the product is continuously monitored and product improvements can be suggested during the complete life cycle, but especially during the review meeting.

In general, team members can suggest product improvements to the product owner during the whole iteration.

Review Meeting

Customers and end users rarely know what they want until they see working software. In Scrum, at the end of each sprint, the new features of the product are demonstrated to the product owner, stakeholders, and customer during the review meeting.

The review meeting provides the opportunity to inspect, adapt, and optimize the product based on the gained experience and understanding. The product owner decides if proposed improvements have any added value to the customer. In the event that a suggested improvement is accepted by the product owner, a new backlog item is added to the product backlog, prioritized, and estimated by the team.

The implementation of accepted improvements is handled in the same way as other features from the product backlog.

6. Quality Management (QM)

6.3 Monitor Process

Goals

The test manager or (independent) quality manager will assess the quality of the test process that is followed throughout the test project.

Compliance with applicable standard testing processes or corporate test strategy and test policy (if available) is checked. Discrepancies are reported and possible improvement actions are suggested. Implementation of these improvement actions (when deemed both necessary and possible) can also be part of this activity. When, for some reason, it is decided not to implement the suggested improvements, these suggestions should be referred to the final test evaluation report.

Control over the testing process of **monitoring the quality of the test process** that satisfies the business requirement for testing of **making sure that the test process that is being used is effective (doing the right things) and efficient (doing the things right)** by focusing on:

- Comparing compliance of the actual test process with the intended or documented test process
- Determining and reporting non-compliance (deviations from intended or documented test process) of the actual test process
- Taking the necessary measures to remediate all reported non-compliancy of the project's test process
- Detecting and reporting shortcomings and insufficiencies of the intended or documented test process
- Taking the necessary measures to remediate all reported deficiencies of the intended or documented test process

and is achieved by

- Assessing process quality
- Defining process improvements
- Planning process improvements
- Implementing process improvements
- Verifying process improvements

Steps

During this testing activity, testing staff (and other relevant stakeholders) will proceed according to the following steps.

QM 6.3.1 Assess Process Quality

QM 6.3.2 Define Process Improvements

QM 6.3.3 Plan Process Improvements

QM 6.3.4 Implement Process Improvements

QM 6.3.5 Verify Process Improvements

QM 6.3.1 Assess Process Quality

The quality level of the project's test process is checked based on the intended (not documented) or expected (documented in the corporate test policy or test strategy) level of quality. This happens on a daily basis and can be considered to be part of the "Monitor and Adjust Test Plan" activity.

In the event that process non-quality is identified, applying more formal (test) process assessment methods can deliver more detailed information on the process non-quality. Process non-quality can originate from non-compliance of the actual test process with the intended or expected test process, or from deficiencies in the intended or expected test process. Regardless of the origin, all process non-quality is reported.

QM 6.3.2 Define Process Improvements

Actions to remediate reported process non-quality are defined. At the same time, if not already available from the corporate's test policy or test strategy, metrics that can be used to measure the test process quality level will be defined.

QM 6.3.3 Plan Process Improvements

Based on available resources (time, budget, people), defined actions to remediate process non-quality are planned for execution. Activities for collecting data measures and calculating defined metrics to measure process improvements (before/after) are also planned. All process non-quality due to non-conformance with the intended or expected test process should be addressed during the test project. Actions to resolve deficiencies of the test process can be postponed or referred to a higher level of responsibility.

QM 6.3.4 Implement Process Improvements

A "before picture" of the respective process quality is created. Planned actions for collecting data measures and calculating "before" metrics are executed. Once the before picture is created, actions for remediating respective process quality are executed as planned.

QM 6.3.5 Verify Process Improvements

Once the process improvements have been implemented, planned actions for collecting data measures and calculating "after" metrics are executed. Based on the before and after metrics, actual process improvement is verified. When the improvement proves to be insufficient, the improvement cycle might be repeated.

Maturity

The testing activity **"Monitor Process"** that satisfies the business requirement of **making sure that the test process that is being used is effective (doing the right things) and efficient (doing the things right)** is at:

Initial level

The "Monitor Process" testing activity is executed in an uncontrolled and chaotic way.

- The activity is not organized for all projects.
- If the activity is executed, it is executed in a chaotic and ad hoc manner.

Starter level

Determining and improving test process non-quality is incorporated into the daily activity of following up on the test project. The test manager responds to the determined process non-quality on an ad hoc basis. The test manager identifies which process non-quality is addressed by taking into account feasibility (available resources) and plans test process improvement actions accordingly.

- For each project, the test process is monitored.
- The process monitoring is merely part of the "Monitor and Adjust Test Plan" activity.
- Process monitoring is executed in a rather informal way; the test manager reacts to process weaknesses that are observed during daily follow up of the test project.
- It is the test manager who, in the end, decides whether the process weaknesses will be addressed and how this will be done.
- The test manager escalates test process improvement actions that are beyond the scope of his mandate to the project manager.
- Defined test process improvement measures are implemented in an ad hoc manner.

Experienced level

Determining and improving test process non-quality is incorporated into the daily activity of following up the test project. Together with principal stakeholders, the test manager and quality manager identify what process non-quality will be addressed by taking into account feasibility (available resources) and incorporate test process improvement actions into the global test project.

- For each project, the test process is monitored.
- The process monitoring is part of the "Monitor and Adjust Test Plan" activity.
- When possible process non-quality is identified, the test manager can decide to organize a formal (self-) evaluation of the test process (by the test team and principal stakeholders) or a formal test process assessment guided by the project's or company's quality manager or an external test consultant.
- The test manager and/or quality manager discuss possible process improvement measures with principal stakeholders. Together, they decide on the process improvements that will be addressed in the ongoing test project (given available resources and quality constraints).
- Defined and agreed-upon test process improvement actions are incorporated into the global test plan (see "Plan Test Project" and "Monitor and Adjust Test Plan" activities).
- Test process improvement actions that cannot be implemented in the scope of the (current) test project are drafted in a lessons-learned report.
- The impact of test process improvement actions is monitored.

Master level

Determining and improving test process non-quality is incorporated into the daily activity of following up on the test project. However, external (not organized by the project) audits can be a secondary source of test process non-quality determination. Together with relevant stakeholders, the test manager and quality manager identify what process non-quality is addressed by taking into account feasibility (available resources) and incorporate test process improvement actions into the global test project.

- For each project, the test process is monitored.
- The process monitoring is part of the "Monitor and Adjust Test Plan" activity.
- When possible process weaknesses are identified, the test manager can decide to organize a formal (self-) evaluation of the test process (by the test team and all stakeholders) or a formal test process assessment guided by the project's or company's quality manager or an external test consultant.
- In addition to the daily follow up, and in line with the corporate quality policy, an (external) test process assessment can also be organized.
- The test manager and/or quality manager discuss possible process improvement measures with all stakeholders. Together, they decide on the test process improvements that will be addressed in the ongoing test project (given time, resource, and quality constraints).
- Defined and agreed-upon test process improvement actions are incorporated into the global test plan (see "Plan Test Project" and "Monitor and Adjust Test Plan" activities).
- Test process improvement actions that cannot be implemented in the scope of the (current) test project are drafted in a lessons-learned report.
- The impact of test process improvement actions is measured.

Agile Testing Add-ons

Retrospective

Agile promotes continuous improvement, as described in one of the 12 Agile principles, "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly."
In Scrum, the forum to define regular process improvements is the sprint retrospective. The Scrum master facilitates the retrospective.

The main purpose of the retrospective is to inspect how the last iteration went. The team members look for improvements regarding the current process they are following (not only with regard to process, but also to people and technology). They will identify what went well and what could be improved on in the next sprint, taking the strengths and weaknesses of the process into account. The focus of these improvements is often limited to the team or sprint achievements and less on cross-project learning.

From a testing point of view, any improvement that enables the team to deliver higher quality during the sprint is welcome. Process improvements are generally planned for the next iteration, monitored daily, and followed up on. During the next retrospective meeting, the team verifies if the improvements had any added value to the team and decides to keep, adapt, or stop the action(s).

Retrospective Tools

There are several tools available to keep retrospective meetings interesting and to collect as much information as possible:

- Happiness Histogram
- Six Hats Thinking
- Retrospective questions
- 5 Whys and the 5 Whys Retrospective
- One Word Retrospective
- Retrospective Dialogue Sheets
- Retrospectives with Lego
- Perfection Game

Acknowledgments

The STBoX 3.0 generic software testing process framework is the result of a joint effort by many CTG colleagues, each adding their bit to the whole. Without them, it would never have been possible to create this solid, practical, and experience-based framework. Thank you to all of these colleagues! STBoX 3.0 is an achievement made possible by all of you.

Operating under the umbrella of "CTG Labs," CTG's Research and Development solution, STBoX 3.0 was created by CTG's "STBoX Center of Excellence." Thank you to all of the CTG test consultants who participated in the Center of Excellence during the last few years: **Nicky Andries**, **Michael Arefi**, **Sven Borghers**, **Yannick Brabants**, **Pascal Collard**, **Stefan De Cap**, **Timothy De Luyck**, **Sven Du Four**, **Jochen Gyssels**, **Kevin Herinckx**, **Gunther Leonhardt**, **Stijn Mullie**, **Christian Mwizerwa**, **Kaj Schittecat**, **Alexander Siccard**, and **Olaf Skwara**. Thank you for taking the time to share your experiences, perform research, create new ideas, and put it all on paper. It was your contributions in the first place that made it possible to create this framework.

A special thanks to **Alessandra Glista**, **Emilie Laubacker**, and **Christine Vermeiren**, our colleagues from the Marketing department, who helped edit and lay out the booklet. Also, a special thanks to **Kevin Boutsen** for developing the STBoX 3.0 website.

Last but not least, a big thank you to the members of our CTG management for their support: **Pieter Vanhaecke**, **Bob Daelman**, **Amanda LeBlanc**, and **Filip Gydé**.

STBoX 3.0: The Road to Testing Maturity

Since CTG released its first version of STBoX (Software Testing Based on CTG eXperience), software testing has come a long way. Most companies have software testing in place as one of their mainstream development activities and software testing is often being executed by skilled and trained testing professionals. It's no longer a question of introducing software testing, but a question of how software testing can be improved to face new challenges. DevOps and shortened time to market, the internet of things and the multitude of mobile devices, and the ever-increasing importance of quality characteristics like security, performance, and usability are only a few examples of current software testing challenges that need to be conquered.

STBoX 3.0 will help in dealing with these challenges. It represents the evolution of STBoX from a methodology for software testing (released in 2006), to a software testing knowledge base for different environments (released in 2009), to this generic software testing process framework, applicable to many different situations and contexts. STBoX 3.0 offers project managers, test managers, and quality managers a generic software testing process that can be tailored to the particular needs of their project. STBoX 3.0 also offers policy makers a generic testing maturity model to measure and improve their company's testing processes as needed. Finally, STBoX 3.0 also offers all test professionals a source of inspiration to get their testing started quickly.

STBoX 3.0 was created by CTG's "STBoX Center of Excellence" collecting and consolidating the experience of its 120 test consultants who are active in many test assignments and deal with current software testing challenges on a daily basis. The Center of Excellence is led by Sven Borghers and Michael Arefi. Together, they share nearly 35 years of software testing experience.

Sven Borghers is a Senior Test Consultant with nearly 20 years of experience with all aspects of testing. He has contributed to many different assignments involving test execution, test design, test coordination, test management, and test process improvement. In addition to his consultancy work, he is also a well-appreciated trainer and coach due to his experience and ability to provide real-life testing examples. Sven has been the driving force behind the STBoX Center of Excellence for many years.

Michael Arefi is a Senior Test Manager with more than 15 years of experience, built during the many projects in which he has held various roles, both in classical and Agile environments. Michael specializes in leading Agile projects. As a Scrum master, he successfully proved that this role can be perfectly fulfilled by a tester. Apart from his contribution to STBoX Center of Excellence, Michael is also the lead of CTG's Agile Center of Excellence.

STBoX 3.0 is also available on **www.stbox.eu**.

Bibliography

Bach, James, and Michael Bolton. "Rapid Software Testing – Appendices." Published class lecture. Satisfice, Inc., 2015. http://www.satisfice.com/rst-appendices.pdf.

Cohn, Mike. *Succeeding with Agile: Software Development Using Scrum*. Ann Arbor: Addison-Wesley, 2009.

Crispin, Lisa, and Janet Gregory. *Agile Testing: A Practical Guide for Testers and Agile Teams*. Crawfordsville: Addison-Wesley, 2009.

Hand, Richard. " 'In-The-Wild Testing': The Missing Link in the QA Chain." *Software Test Professionals*. January 26, 2012. http://www.softwaretestpro.com/Item/5422/"In-The-Wild-Testing"-The-Missing-Link-in-the-QA-Chain/Test-and-QA-Software-Test-Professionals-Conference.

Kohl, Jonathan. *Tap Into Mobile Application Testing*. Victoria, British Columbia: Leanpub, 2013. https://leanpub.com/testmobileapps.

Miller, Jessica. "5 Killer Hallway Usability Testing Tips." *Usability Lab* (blog). September 17, 2014. http://usabilitylab.walkme.com/5-killer-hallway-usability-testing-tips/.

Tuckman, Bruce. "*Developmental Sequence in Small Groups*." Psychological Bulletin Vol 63, no. 6 (June 1965): 384-399. http://dx.doi.org/10.1037/h0022100.

Wake, William. "INVEST in Good Stories, and Smart Tasks." *XP123 Exploring Extreme Programming* (blog). August 17, 2003. xp123.com/articles/invest-in-good-stories-and-smart-tasks/.

Reliability
Matters

CTG provides industry-specific IT services and solutions that address the business needs and challenges of clients in high-growth industries in North America and Western Europe. CTG also provides strategic staffing services for major technology companies and large corporations. Backed by 50 years of experience and proprietary methodologies, CTG has a proven track record of reliably delivering high-value, industry-specific staffing services and solutions to its clients. CTG has operations in North America, Western Europe, and India. The company regularly posts news and other important information online at www.ctg.com.