TMMi in the Agile world

Version 1.4

Produced by the TMMi Foundation

Editor: Erik van Veenendaal

Contributors

Asim Ali	(United Arab Emirates)
Katalin Balla	(Hungary)
Clive Bates	(UK)
Jan Jaap Cannegieter	(The Netherlands)
Vahid Garousi	(The Netherlands)
Alon Linetzki	(Israel)
Fran O'Hara	(Ireland)
Jurian van de Laar	(The Netherlands)
Leanne Howard	(Australia)
Poonam Jain	(India)
Tim Moore	(UK)
Alfonsina Morgavi	(Argentina)
Meile Posthuma	(The Netherlands)
Matthias Rasking	(Germany)
Chaobo Shang	(China)
Erik van Veenendaal	(Bonaire – Caribbean Netherlands)
Blaine Webb	(UK)
Karolina Zmitrowicz	(Poland)

Version history

This section is provided for information purposes only.

Version	Date	Comment
1.0	30/06/2017	Introduction Chapter and all TMMi Level 2 process areas addressed
1.1	16/05/2018	The interpretation in an Agile context of all TMMi level 3 process areas added.
1.2	10/12/2018	The interpretation in an Agile context of all TMMi level 4 process areas added.
1.3	03/07/2019	The interpretation in an Agile context of all TMMi level 5 process areas added.
1.4	24/12/2019	Added sections 6.4 and 6.5 to explicitly indicate that all of TMMi levels 4 and 5 is applicable.

Contents

1 Introduction

1.1 Objectives

This document explains how combining an Agile approach for software development with the TMMi test process improvement model is a possible route to achieve your business objectives. It explains how TMMi can be used and applied beneficially in an Agile context. Proven alternatives are provided to traditional testing approaches that implement TMMi practices while maintaining and perhaps even increasing agility. This document covers how TMMi can help Agile organizations, irrespective of how mature the organization is on their journey to increasing levels of agility, by providing reminders of critical testing practices that frequently lose visibility as organizations grow and project pressures increases. Each TMMi process area and its specific goals are discussed one by one. It is stated how these relate to Agile and what the practices will typically look like. If a practice is not expected to be performed in an Agile context, since it has no added value, this is also explicitly stated.

Many (small) Agile organizations are successful and growing; but they may have few documented processes and little formal training program for the people involved. To maintain the success as the organization grows, they will require some more process discipline. Organizations then typically have the fear of losing the Agile culture that has led to the current success. This is one of the challenges that lies ahead when starting a TMMi test process improvement initiative in an Agile environment.

This document has many target audiences, the two main groups being:

- Mature traditional organizations that want to move to Agile while keeping their process maturity.
- Agile organizations that are successful and are growing. As a result they need some kind of process maturity while at the same time they want to keep the benefits of being Agile.

This document is not intended to be used independently of the original TMMi model. It is intended to provide guidance to those performing test process improvement in an Agile environment on the interpretation and meaning of the various TMMi goals and practices. This document supplements the original TMMi model and should be used as complementary document.

1.2 TMMi and Agile

The mistaken belief is that the TMMi and Agile approaches are at odds. Agile approaches and TMMi can not only co-exist, but when successfully integrated will bring substantial benefits. There is also a challenge of looking at testing differently, being fully integrated within Agile development and what that means in the context of a “test” improvement programme. Note that the “i” of TMMi refers to the fact that testing should be an integrated part of software development, and not be treated as something that is totally separate. Literature and presentations on testing in Agile projects tend to focus on unit testing, test automation and exploratory testing, but there is more! Using the TMMi model in an Agile context provides reminders of critical testing practices that are often “forgotten”. This document will show with examples that TMMi and Agile methods can effectively work together. The challenge is to apply lean principles to empower Agile practices and facilitate TMMi practices.

When implementing TMMi one must take into account that the intent of the TMMi model is not to “impose” a set of practices on an organization, nor is it to be applied as a standard to which one must “prove compliance”. Used appropriately, TMMi can help you locate the specific testing areas where change can provide value given the business objectives. This is true regardless of the lifecycle model that is being applied. It is important to always remember that TMMi practices are an expected component, but can also be achieved by what is referred to as “alternative” practice with respect to a defined TMMi practice. Always think, what is the intent of the practice, what is the rationale and how does it add value to the business? Often in an Agile culture the intent is already achieved but through an alternative practice. Typically “any” solution is compliant as long as its driven by business needs! When using TMMi don’t be too prescriptive, this was not how TMMi is intended in the first place. Always interpret the TMMi goals and practices to the context of your situation. In general by first establishing the process needs within your specific business context, decisions can be taken on how to focus and drive process improvement priorities.

Most of the conflicts that arise between the TMMi and Agile are based in either a historical TMMi view of what a "good practice" should look like when implemented, or a misunderstanding of the rationale for agile practices based on how they should support Agile values. TMMi experts, including (lead) assessors, will need to re-consider and potentially re-think messages that might be inadvertently shared related to what a "good TMMi compliant" practice should look like when implemented. When Agile approaches are implemented appropriately together with TMMi processes, this will result in the effective implementation of testing practices, not their deletion. Note that in addition to the specific (testing) practices within the TMMi, there are also generic practices. The purpose of the generic practices is to support the institutionalization of a process area, which effectively means ensuring the organization has an infrastructure in place to support the process area when new people come in or other changes happen within the organization.

Moving from traditional based software development to Agile can also bring out the initiative to prune and lean the processes as they are defined today. In this way TMMi based organizations will benefit from the Agile way of thinking. There has been a tendency for people to read things into the TMMi model that are not there and thus create unnecessary non-value-added process and work products. By going back to the roots and improvement goals and using the TMMi model as it is intended, one will support the alignment of real value-added processes with real process needs and objectives. Pruning and leaning processes with an Agile mindset will result in processes that reflect what people really do and will ensure that only data that is used is being collected. The Agile mindset will also bring a focus on keeping things as simple as possible, which is typically not easy but will bring a benefit for those practicing a TMMi implementation. Improvements within Agile will typically take place through small empowered teams that can take rapid action, which is another way where TMMi can benefit from being Agile. Also remember there is a natural break between TMMi level 3 and, TMMi levels 4 and 5. Especially Agile organizations are recommended to critically choose and do the practices at TMMi levels 4 and 5 that matter and have added value. Whilst TMMi is comprehensive, to be successful organizations must identify the key testing practices and improvements requiring focus.

1.3 Test Maturity Model integration (TMMi)

The TMMi framework has been developed by the TMMi Foundation as a guideline and reference framework for test process improvement, addressing those issues important to test managers, test engineers, developers and software quality professionals. Testing as defined within TMMi in its broadest sense to encompass all software product quality-related activities.

TMMi uses the concept of maturity levels for process evaluation and improvement. Furthermore process areas, goals and practices are identified. Applying the TMMi maturity criteria will improve the test process and has shown to have a positive impact on product quality, test engineering productivity, and cycle-time effort. TMMi has been developed to support organizations with evaluating and improving their test processes.

TMMi has a staged architecture for process improvement. It contains stages or levels through which an organization passes as its testing process evolves from one that is ad hoc and unmanaged to one that is managed, defined, measured, and optimized. Achieving each stage ensures that all goals of that stage have been achieved and the improvements form the foundation for the next stage.

The internal structure of TMMi is rich in testing practices that can be learned and applied in a systematic way to support a quality testing process that improves in incremental steps. There are five levels in TMMi that prescribe the maturity hierarchy and the evolutionary path to test process improvement. Each level has a set of process areas that an organization must implement to achieve maturity at that level. The process areas for each maturity level of TMMi are shown in Figure 1.

A main underlying principle of the TMMi is that it is a generic model applicable to various life cycle models and environments. Most goals and practices as defined by the TMMi have shown to be applicable with both sequential and iterative life-cycle models, including Agile. However at the lowest level of the model, many of the sub-practices and examples provided are (very) different depending on the life cycle model being applied. Note that within TMMi, only the goals are mandatory, the practices are not.

TMMi is freely available on the web site of the TMMi Foundation. The model has been translated in Spanish, French and Chinese. TMMi is also available in published book format.

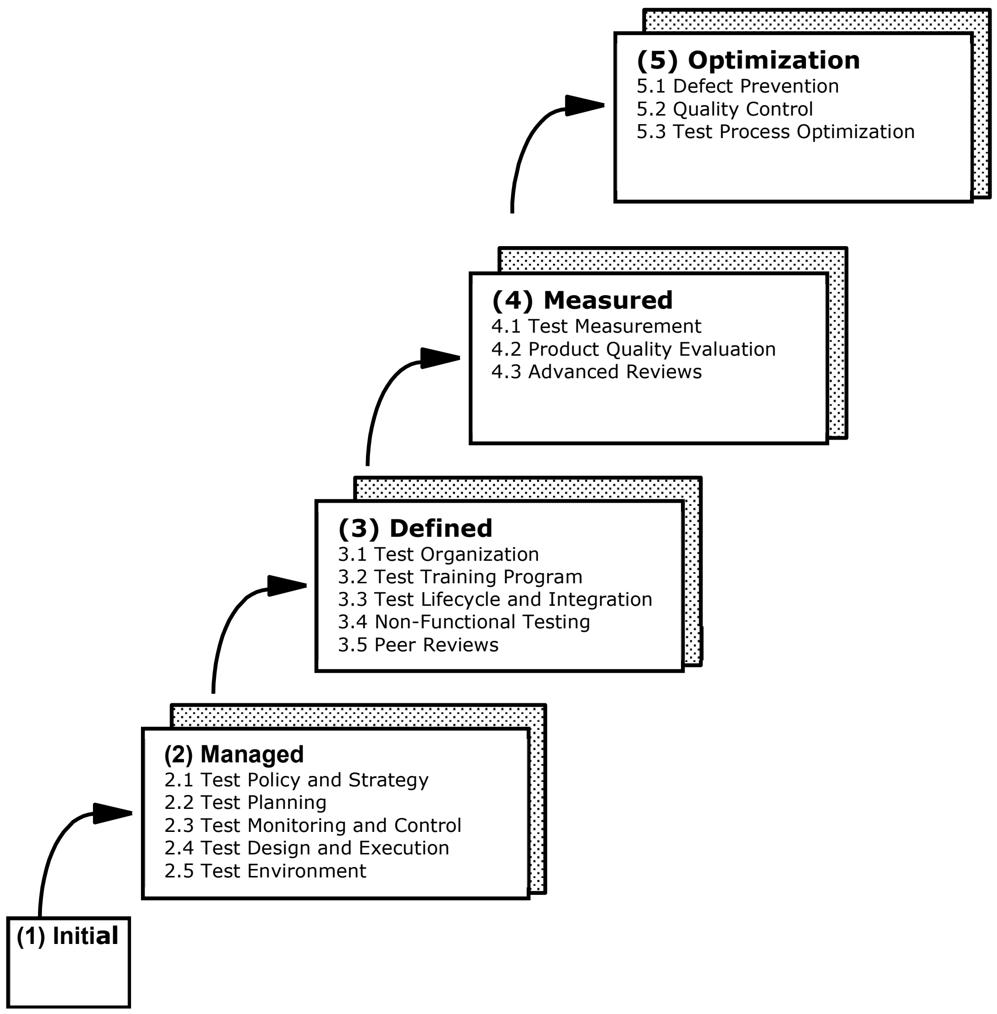

(5) Optimization
5.1 Defect Prevention
5.2 Quality Control
5.3 Test Process Optimization
(4) Measured
4.1 Test Measurement
4.2 Product Quality Evaluation
4.3 Advanced Reviews
(3) Defined
3.1 Test Organization
3.2 Test Training Program
3.3 Test Lifecycle and Integration
3.4 Non-Functional Testing
3.5 Peer Reviews
(2) Managed
2.1 Test Policy and Strategy
2.2 Test Planning
2.3 Test Monitoring and Control
2.4 Test Design and Execution
2.5 Test Environment
(1) Initial

Figure 1: TMMi maturity levels and process areas

1.4 Agile

In 2001, a group of individuals, representing the most widely used lightweight software development methodologies, agreed on a common set of values and principles which became known as the Manifesto for Agile Software Development or the Agile Manifesto. The Agile Manifesto contains four statements of values:

- Individuals and interactions *over* processes and tools
- Working software *over* comprehensive documentation
- Customer collaboration *over* contract negotiation
- Responding to change *over* following a plan

The Agile Manifesto argues that although the concepts on the right have value, those on the left have greater value. Agile itself is not a methodology, but different methodologies have meanwhile been developed that provide practices to put these values into action. Three important representatives of methodologies that support Agile are Extreme Programming (XP), Scrum, and Kanban.

1.5 Test process improvement in an Agile context

Agile has a strong focus on empowered teams performing their own continuous improvement locally also through frequent retrospectives. Some of these improvements may be test focused. These may be directed by a TMMi based improvement project at a higher level but may also be addressing local team process issues relating to testing. It is very important that the action in the context of test process improvement are not interpreted and/or used as taking this local ownership away from Agile teams.

Some of the principal aspects to consider, when looking at the influence of Agile on improvement context are:

- Improvement cycle frequency
- Organizational aspects
- Scope of improvements
- Source of improvements
- Level of (test) documentation
- Improvement methods.

Within projects which use Agile, improvements generally take place in frequent feedback loops which enable test process improvements to be considered frequently (e.g., at the end of a sprint when using Scrum). Because the scope is often limited to the previous cycle or iteration, small but frequent improvements are made which focus mainly on solving specific project problems. The focus of these improvements is often not on cross-project learning and institutionalization of improvements.

Looking at how test process improvement is organized and managed, there is likely to be less focus on a test process group at an organizational level and more emphasis on the self-management of teams within the project. These teams generally have the mandate to change the testing process within the project used to match their needs, resulting in highly tailored processes. However, some organizations also use weekly test stand-up meetings to bring things to a higher and cross-project level.

Since there is a more project-specific focus on (test) process improvement, less emphasis is likely to be placed on broader issues affecting testing across the organization. This could mean, for example, that fundamental testing problems may not be fully addressed because they are beyond this project-centric context. A typical example here is the approach taken to testing certain quality attributes, such as performance and reliability. These issues may become deferred from iteration to iteration because they often require more skills and resources than the project team has available. It is hard in these areas to make a substantial next step without major investments. Solving problems only on a project level could also easily lead to sub-optimization and losing touch with the bigger picture.

In Agile context, the range and number of alternative improvement ideas to be considered may be significantly more than compared to non-Agile life cycle models. Since all members perform some testing within the project, these ideas can come from any team member. This places a stronger emphasis on evaluating and prioritizing improvement suggestions, which may be more of a team effort than a task assigned to a test process improver. Since this may require the specific testing knowledge of a test process improver, they can also act as a consultant to the team on request.

In projects using an Agile methodology, don't expect to find the level of test documentation you would expect from projects using a sequential lifecycle. There may be a single combined "test document" covering the essential elements of a test policy, test strategy and even high-level test plan. Test process improvers should avoid making "improvement" suggestions which call for more rigorous and thorough test documentation. One of the main Agile principles is of course that documentation is only created when there is a clear and unambiguous need for it. Not only will the test documentation be less detailed, the same goes for the process documentation. Often a strategy referred to as "formalizing informality" is successfully applied in Agile environments. If something is working well, there is no need to change this for the sake of TMMi. However, documented it can be taught and shared with others which is often beneficial. What is meant here with the "formalizing informality" strategy, is that if there is a process that works, but it is informal in certain ways, one can teach and document it just like it is being performed. Being lightweight as a process means that not every possible usage scenario is addressed in its description. These processes must therefore be supported by mentoring and on-the-job assistance especially during the period of initial deployment. As a result, bringing test process maturity to an Agile organization while maintaining the Agile culture, one also needs more – not less – training. Likewise during an assessment the focus to gather evidence will shift toward doing more interviews instead of studying artifacts.

The methods used to propose test process improvements when using Agile will tend to focus on analytical methods for evaluating the root-causes of problems, such as cause-effect diagrams. These are particularly useful methods for the problem-solving mind-set which is important at the end of an iteration.

2 TMMi Level 2 Managed

2.1 Process Area 2.1 Test Policy and Strategy

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

2.1.1 SG1 Establish a Test Policy

Any organization that embarks on a test improvement project should start by defining a test policy. The test policy defines the organization’s overall test objectives, goals and strategic views regarding testing and test professionals. It is important for the test policy to be aligned with the overall business (quality) policy of the organization. Test improvements should be driven by clear business goals, which in turn should be documented in the test (improvement) policy. A test policy is necessary to attain a common view of testing and its objectives between all stakeholders within an organization. This common view is required to align test (process improvement) activities throughout the organization. Note that test objectives should never be an objective by themselves, they are derived from the higher level goal to establish working software and product quality.

The above is also true in an organization that practices Agile software development. Indeed within many organizations there is much discussion on the changing role of testing, independence of testing, test automation and professional testers in Agile software development. These items and others are typically issues that should be addressed in a discussion with management and other stakeholders and documented in a test policy. Any organization, including those that practice Agile, that wants to start a test improvement project needs to identify and define the business drivers and needs for such an initiative. Why else start an improvement project? By spending time to capture the true business needs, one can provide a context to make decision where to focus the (test) improvement priorities, e.g., on which process area. Note that a test policy is typically a one-page lean document, web page or wall chart at organizational level, and not a document at project level.

The TMMi specific goal Establish a Test Policy, including its specific practices, is fully applicable to organizations applying Agile software development. Of course the elements of a test policy can also be incorporated in a development policy. There is no specific TMMi requirement for it to be a separate document. The development policy in an organization applying Agile software development could for instance identify Scrum as its management framework, XP as a main method being used and adherence to the Agile values as a main principle. Another important principle that could be mentioned is that everyone in the team is responsible for everything; also product quality is a team-responsibility.

However, there is an important consideration for Agile in relation to the test policy and especially defined test (improvement) goals. Although there may be overall goals that relate to test process improvement in the organization, this needs to be balanced with individual projects and Agile teams being responsible for improving their own process. The Agile process improvement challenge is to guide and frame the improvement on an organization level while not reducing an individual Agile team’s sense of ownership of its own process.

2.1.2 SG2 Establish a Test Strategy

The test strategy follows the test policy and serves as a starting point for the testing activities within projects. A test strategy is typically defined either organization-wide or program-wide. A typical test strategy is based on a high-level product risk assessment and will include a description of the test types, test quadrants and test levels that are to be performed, for example: unit, acceptance, regression and performance test. It is not good enough to just state for example that within an iteration unit testing and acceptance testing will be carried out. We need to define what is meant by unit and acceptance testing; how these are typically carried out and what their main objectives are. Experience shows that when a test strategy is defined and followed, less overlap between the various testing activities is likely to occur, leading to a more efficient test process. Also, since the test objectives and approach of the various test types and levels are aligned, fewer holes are likely to remain, leading to a more effective test process and thus higher level of product quality. To establish this alignment it is highly recommend to cover both Agile testing and any non-Agile testing taking place in the same project under one umbrella test strategy. However, if there are separate Agile and non-Agile projects, there are typically two test strategies.

A test strategy is a vital document within an Agile environment. It defines on a high-level the testing to be done in the Agile teams (iteration teams); what test types, test quadrants and test levels are executed and on a high-level their approach. The document describes how testing is organized, e.g., what testing takes place inside Agile teams and what testing takes place outside. It will define the relationship from Agile teams to those test levels that are performed outside their scope, e.g., hardware/software integration testing, system integration testing or beta testing. A test strategy ensures that all those involved in testing understand the bigger testing picture.

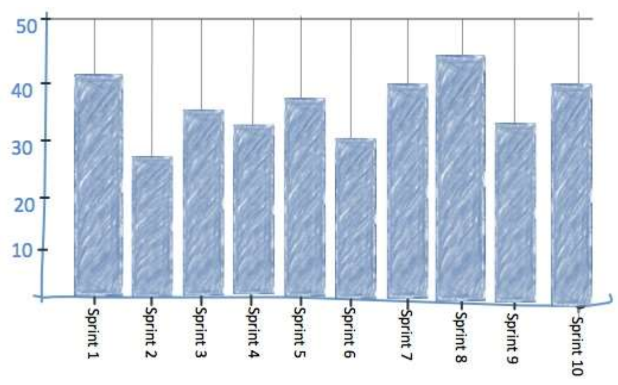
The lean test strategy document is often a good solution for Agile organization and a way out from detailed (project) test plans. During release planning the available organization-wide or program-wide test strategy is discussed and confirmed, or a derived test strategy is created specifically for the project. The test strategy confirmed or created provides a test framework spanning all iterations.

The TMMi specific goal Establish a Test Strategy, including its specific practices, is fully applicable to organizations applying Agile software development. Note that active “distribution” activities, e.g., for the test strategy document, may be less relevant in agile. Those who are a stakeholder should already have been involved in earlier discussions as part of the whole-team approach. However, a test strategy is a not at team-level but on a higher level. The whole team operates at the team level so an organizational or programme document still needs distribution to teams and indeed other stakeholders e.g., those performing testing activities outside the team if a hybrid model exists.

2.1.3 SG3 Establish Test Performance Indicators

The business objectives for test improvement, as defined in the test policy, need to be translated into a set of key test performance indicators. The test policy and the accompanying performance indicators provide a clear direction, and a means to communicate expected and achieved levels of test performance. The performance indicators must indicate the value of testing and test process improvement to the stakeholders. Since investments in process improvement need long term management support, it is crucial to quantitatively measure the benefits of an improvement program to keep them motivated. Beware that this TMMi specific goal is about defining a limited number (e.g., 2 or 3) test performance indicators. It is not about setting up and implementing a full measurement program but rather about defining a core set of indicators that tell you how the value of testing is changing over time and within different delivery environments.

Within Agile the focus will be more team-based and systems thinking. This may result in a corresponding broadening of the indicators to the team and overall system rather than being confined solely to the specifics of testing itself. Also indicators at TMMi level 2 are mainly related to the end-results of the iterations. Examples would include escaped defects, velocity, customer satisfaction ratings, effort/waste, test automation percentage etc. The challenge would be to define the appropriate blend of indicators relating to the team-based approach and systems thinking while giving a good indication of the performance achievements of a TMMi-based test improvement programme.



The TMMi specific goal Establish Test Performance Indicators, including its specific practices, is fully applicable although the performance indicators selected and applied may well have a larger scope and be more broader than being related to testing only. Of course the latter will make the analysis and interpretation of performance indicators more challenging. In fact the performance indicators being used may not be called test performance indicator, but rather a team performance indicator or system performance indicator. This is still ok in the context of TMMi as long as it has testing related elements and is being used to evaluate the progress being made doing test improvement.

2.2 Process Area 2.2 Test Planning

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

Beware that the key to successful test planning is in upfront thinking ("the activity"), not in defining the associated test plan ("the document").

For Agile lifecycles, two kinds of planning typically occur, release planning and iteration planning. The Test Planning process area at TMMi level 2 focuses on the testing related activities at both release and iteration planning. Release planning looks ahead to the release of a product at the start of a project. Release planning requires a defined product backlog and may involve refining larger user stories into a collection of smaller stories. Release planning provides the basis for a test approach and test plan spanning all iterations. Release plans are high-level. After release planning is done, iteration planning for the first iteration starts. Iteration planning looks ahead to the end of a single iteration and is concerned with the iteration backlog.

Note, the Test Planning process area has a number of specific practices. The TMMi doesn't state when or how to conduct these practices. It doesn't state you can't plan incrementally either. The traditional approach has been to solidify as many decisions as one can up front, so the related cost and schedule can also be solidified. The rationale for this approach has been to better estimate work and reduce the risk of scope creep. Agile approaches typically take the position that we gain greater value by continuous refinement of the plan based on the latest information and on-going collaboration with the customer.

2.2.1 SG1 Perform Risk Assessment

Exhaustive testing is impossible, and choices need always to be made and priorities need always to be set. This TMMi goal is therefore also applicable for Agile projects. For Agile projects a high-level product risk assessment shall be performed based on a product vision document or set of high-level user stories at release planning. For each iteration a more detailed product risk session shall be performed based on the user stories or other requirements for that iteration as part of the iteration planning session. The product risk assessment process in an Agile project will have a much more lightweight format compared to those applied in a traditional projects following a sequential lifecycle model. An example of a lightweight product risk technique to be used is risk poker [Van Veenendaal].

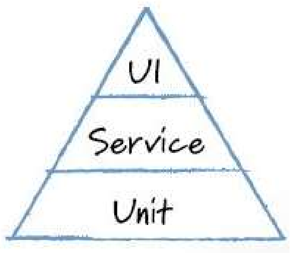
At release planning, business representatives who know the features in the release provide a high-level overview of the functionality to be developed, and the whole team, including the tester(s), will assist in the risk identification and assessment.

During iteration planning the Agile team identifies and analyzes product risks based on the user stories to be implemented in the upcoming iteration. Preferably all members of the Agile team and possibly some other stakeholders participate in the product risk session. The result is a prioritized list of product risk items identifying the critical areas for testing. This in turn will help in determining the appropriate amount of test effort to allocate in order to cover each risk with enough tests, and sequencing these tests in a way that optimizes the effectiveness and efficiency of the testing work to be done. The estimated tasks on the task board can be prioritized in part based on the level of product risk associated with them. Tasks associated with higher risks should start earlier and involve more testing effort. Tasks associated with lower risks should start later and involve less testing effort.

2.2.2 SG2 Establish a Test Approach

A test approach is defined to mitigate the identified and prioritized product risks. For a specific iteration, the items and features to be tested are identified during iteration planning. This activity is also based on the result of the product risk session. The prioritized list of items to be tested typically relates to the user stories to be tested in this iteration. The features typically relate, among others, to the various software quality characteristics to be tested. New products risk may become apparent during the iteration requiring additional testing. Issues like new product risks requiring additional testing are typically discussed at daily standup meetings.

The test approach that is defined at iteration level to mitigate the risks can cover for example additional reviewing of user stories and acceptance criteria, testing effort proportional to the level of risk, the selection of appropriate test technique(s) based on the level and type of risk. The test approach at release level will be at a much higher level and shall be based on defined test strategy at programme or organizational level. Often a test approach is held or displayed on the team/project wiki.



An important risk that is always apparent in iterative development, is the regression risk. The test approach needs to define how the regression risk is managed. Typically this will be done by creating a specific regression test set, which is preferably automated. In this context the test automation pyramid [Cohn] is helpful. It shows how to maximize the value of regression test automation, starting with unit tests at the lowest level of the pyramid and moving on to service level testing. User interface testing sits at the very top. Unit tests are fast and reliable. The service layer allows for testing business logic at the API or service level, where you're not encumbered by the user interface (UI). The higher the level, the slower and more brittle testing becomes.

Entry criteria, normally a part of a defined test approach (specific practice 2.3) are likely to be non-relevant to Agile development. Within Agile software development, testing is an integral part of the team process and an almost continuous activity. Consequently, there is no need for a specific checklist or gateway to determine if testing can, or cannot, be started. This also applies for a component going within an Agile team from one test stage (e.g., unit testing) to another (e.g., acceptance testing).

Testing exit criteria (specific practice 2.4) are part of the so-called Definition of Done (DoD). It is important that the DoD has specific test related criteria, e.g., for test coverage and product quality (defects). The iteration should result in the implementation of the agreed set of user stories and meet the (test) exit criteria as defined in the DoD. Typically stories that fail to meet the exit criteria are put into the backlog and may be dealt with within the next iteration. Of course within Agile there are no exit criteria for a component going from one test stage to another. Not only will a Definition of Done exist at iteration level, often there is also a DoD at release level spanning multiple iterations. The DoD at release level will again typically have coverage and product quality related criteria.

Specific practice 2.5 Define suspension and resumption criteria is likely to be non-relevant to Agile lifecycles. As testing is an integral part of the Agile software development process, it will of course not be treated as separate and independent activity from other iteration activities. When there are blocking issues that could be considered as potential or actual threats to the progress of testing, these are discussed in the daily standup meeting. In this forum the team will decide what actions, if any, need to be taken to solve the issues. Thus formal suspension and resumption criteria will not be required and defined, issues around this are dealt with as part of the normal Agile routine. The Agile routine thus serves as an alternative practice for this specific practice.

2.2.3 SG3 Establish Test Estimates

For Agile teams, detailed estimates for testing will be made during iteration planning. High-level (test) estimates are made during release planning and possibly also at backlog refinement sessions. All estimates are of course done as a team estimate which includes all effort required to deliver each story. It is important to ensure that testing activities are indeed taken into account during the estimation sessions. This can be done by having testing activities identified as separate tasks and subsequently estimating each individual test task, or by estimating each user story whereby the testing to be done is made explicit and thus taken into account. A tester, being part of the Agile team, shall participate in the estimation sessions. Planning Poker or shirt-size are typical estimation techniques being used at Agile software development.

The work to be done for the next iteration is typically defined by the user stories. The user stories need to small in size for them to be estimable. Estimable is one of the user story criteria defined by INVEST [Wake] and is applicable for user stories that are to be part of an iteration. During iteration planning Agile teams will typically identify and define test-related tasks. Test tasks will be captured on a task-board together with the other development tasks. This set of tasks is the basis for the estimation of the iteration. The set of defined tasks for the upcoming iteration serves as a sort of work-break structure (as used in sequential projects).

At release planning user stories or epic's will typically be more high level defined and not yet detailed into specific tasks. This will of course make the estimations harder and less accurate. As stated Agile projects will not establish a work-breakdown structure as a basis for estimations, but may at release level benefit from a simple diagram that visualizes the product to be developed and thereby adequately scopes the work.

Although an Agile team will estimate in a relatively informal way, the rationale for the estimations should be clear (i.e. what factors are being considered). Discussions based on the rationale promotes a higher level of accuracy of the estimations. Typically in Agile projects the estimation is focused on size (using story points) or effort (using ideal man days as an estimation unit). Costs are normally not addressed as part of estimation sessions in Agile projects.

This TMMi specific goal and its specific practices is therefore fully applicable with the exception of specific practice 3.2 Define test lifecycle. One of the basics of iterative and Agile software development is to work in small chunks. Therefore the tasks that have been identified are typically detailed enough to serve as a basis for (test) estimation. There is thus no need in Agile to also define a lifecycle for testing activities to serve as an additional basis for estimation.

2.2.4 SG4 Develop a Test Plan

Re-stating the comment that test planning is about upfront thinking ("the activity") and not about defining the associated test plan ("the document"). In Agile most of the test planning activities, as defined by this TMMi specific goal, will to be performed during release and iteration planning. However, the result of these activities will typically not be documented in a test plan, especially for iteration planning where they could be reflected on the task board.

As testing is performed as an integral part of the release and iteration planning, the resulting "schedules" will also include testing activities. Rather than a detailed schedule such as developed with sequential lifecycles, the schedule within an Agile project is much more like an ordering of user stories (backlog items) and tasks that reflects the release and iteration priorities, e.g., based on desired delivery of business value. Mind-maps are often used here as a supporting technique. The task board will reflect the iteration priorities. Thus no explicit (test) schedule is established; it is expected that clear release and iteration priorities are defined for the user stories, respectively the tasks to be performed, including the testing tasks.

On a project level, test staffing is part of building the team; the need for test or multi-skilled resources is identified upfront. As projects change or grow, one can easily forget the rationale for the initial selection of an individual to a given team/project, which often includes specific skills needs or experience specifically related to the team/project. This provides a good rationale for writing down the skill needs of people, providing back-up information related to why they were selected for the team/project. Once the Agile teams are defined test staffing is more or less fixed. During iteration planning the identification of the (additional) resources and skills, e.g., for non-functional testing, needed to perform the testing in an iteration can if necessary be discussed to ensure that the team has sufficient testing resources, knowledge and skills to perform the required tests.

The Scrum master should ensure that the product owner provides input to testing as required, e.g., by answering questions and having conversations on user stories. The product owner should be sufficiently available, which should again be organized upfront.

An initial project risk session should be part of release and iteration planning. Identification (and management) of further project risks during the iteration is part of the daily standup meetings and are typically documented by means of an impediment log. It is important that also testing issues are noted in the impediment log. The impediments should be discussed at the daily standup meeting, until they are resolved.

Although no specific and detailed test plan is developed and documented, specific practice 4.5 Establish the test plan is still relevant within an Agile context. The result of the discussions that take place in the context of test planning are however likely to be captured in a lightweight form, possibly a mind-map.

2.2.5 SG5 Obtain commitment to the Test Plan

Within Agile the process to develop and establish a test approach and test plan is a team-based exercise, possible lead by a test professional (being one of team members). Product-quality is a team responsibility. As such, provided the team follows the correct process, commitment to the (test) approach and (test) plan is already an implicit result of release and iteration planning as it‘s a team-effort. This of course is a huge difference with the way of working in a traditional environment where typically the tester prepares the test plan and then thereafter needs to obtain explicit commitment.

In Agile projects, the Agile team (including product owner) must understand and agree on the prioritized list of product risks and the test mitigation actions to be performed. The understanding and commitment can for example be achieved by means of a short presentation followed by a discussion during release or iteration planning explaining the product risks, test approach and its rationale to the team.

During estimation session (see SG 3 Establish Test Estimates) the work load is estimated. The (test) resources for a team are fixed by the setup of the Agile team. User stories that are estimated and selected to be developed take the available resources as a starting point (constraint). Thus again reconciling work and resource levels is not a meaningful activity. The specific practice 3.2 Reconcile work and resource levels is therefore a practice that is typically not relevant in an Agile context. Daily stand-up meetings will be used to address any resource issues during the iteration and to (re-)allocate appropriate resources immediately, or remove a deliverable from an iteration to be reconciled in future iteration or release planning.

2.3 Process Area 2.3 Test Monitoring and Control

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

Following the Agile manifesto and its accompanying principles, there are some things that are fundamentally different to monitoring and control in Agile projects compared to traditional projects. Although monitoring and control are essential elements of an Agile project it does not imply sticking to a rigid plan is the goal, in fact the opposite is true in that both the manifesto and principles talk about welcoming change. Test Monitoring and Control could in an Agile context be interpreted as providing best practices to continuously adjust the plan to keep it current, which is what Agile approaches recommend.

From a test monitoring and control perspective this means we are not plan driven, but constantly reviewing our progress and results from testing, and adapt our plan and approach as appropriate - new product risks may become apparent. The test plan, however it is captured, is a living entity and continuously needs to be reviewed and updated as new information is leant or feedback given. Agile projects also need to keep the 'bigger picture' in mind and monitor and control at higher (release) level as well as at iteration level.

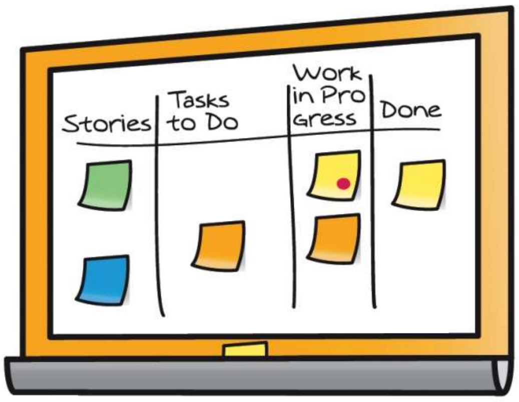
It is important to note that as testing is a process that is fully integrated into the overall process of the Agile team, test monitoring and control is also an integral part of the overall monitoring and control mechanisms of the Agile team. As a result, testers do not report to a test manager as with traditional projects, but to the team.

Since the Test Planning process area at TMMi level 2 focuses on both release and iteration planning, the same applies to test monitoring and control. It is expected to that test monitoring and control is performed at both planning and release level.

2.3.1 SG1 Monitor Test Progress Against Plan

Testers in Agile teams utilize various methods to monitor and record test progress, e.g., progression of test tasks and stories on the Agile task board, and burndown charts. These can then be communicated to the rest of the team using media such as wiki dashboards and dashboard-style emails, as well as verbally during stand-up meetings. Teams may use burndown charts both to track progress across the entire release and within each iteration. A burndown chart will typically show progress against expected velocity and backlog of features to be implemented (performance of the team). Sometimes when test environment resources are scarce and vital, e.g., for non-functional testing, specific burndown charts are used to monitor the test environment resources usage against those agreed during iteration planning. Another common practice is to track test environment issues via the task board, e.g., by using stickers marked 'blocked test environment' on the story cards or by creating a separate column on the board where all stories blocked by environments wait until unblocked. It's all about creating visibility of the impact of a test environment that is blocking progress.

To provide an instant, detailed visual representation of the whole team's current status, including the status of testing, teams may use Agile task boards. The story cards, development tasks, test tasks, and other tasks created during iteration planning are captured on the task board, often using color-coordinated cards to determine the task type. During the iteration, progress is managed via the movement of these tasks across the task board into columns such as *to do*, *work in progress* and *done*. Agile teams may use tools to maintain their story cards on Agile task boards, which can automate dashboards and status overview. However, some teams do not create specific tasks for the individual activities but may just use the story card and annotate comments in this card for tests, referencing Agile tools or wiki's where the testing may be documented. Testing tasks on the task board typically relate to the acceptance criteria defined for the user stories. As test automation scripts, manual scripted tests, and exploratory tests for a test task achieve the pass status, the task moves into the done column of the task board.



The Definition of Done serves as exit criteria against which progress is being measured. The DoD should also be related to testing activities and shows all criteria that need to be satisfied before testing of an user story can be called 'Done'. Note that testing criteria related to the test tasks form only a part of what the team agrees to complete as the Definition of Done. Definition of Done criteria are typically applied at multiple levels, e.g., at iteration level and release level. The whole team reviews the status of the task board regularly, often during the daily stand-up meetings, to ensure tasks are moving across the board at an acceptable rate. If any tasks (including testing tasks) are not moving or are moving too slowly, this than triggers a team-based discussion where the issues that may be blocking the progress of those tasks are analyzed.

The daily stand-up meeting includes all members of the Agile team including testers. At this meeting, they communicate their current status and actual progress to the rest of the team. Any issues that may block test progress are communicated during the daily stand-up meetings, so the whole team is aware of the issues and can act on them accordingly. In this way also risk management is integrated in these daily meetings. Any project risk, including those for testing, e.g., the lack of availability on test environments, can be communicated and addressed during the daily stand-up. (Note, monitoring product risks is part of monitoring product quality and thus discussed hereafter with the specific goal SP 2 Monitor Product Quality Against Plan and Expectations.) Daily standup meetings for daily task management, and Agile team practices related to task course correction are good proven techniques that meet the intent of TMMi specific practices in the Test Monitoring and Control process area.

The milestone review within Agile is at the completion of an iteration. The accomplishments of testing, e.g., against the Definition of Done, will be part of the iteration review. Demos are organized with stakeholders to discuss the business value and quality of the product being delivered.

Stakeholders are represented by the product owner in iteration planning, iteration reviews (demos) and retrospectives. The product owner is involved through discussions on the product backlog and will provide feedback and input to the elaboration of user stories and the design of tests throughout the iteration. Other stakeholders are involved at the end of each iteration during the iteration review. No specific monitoring of stakeholder involvement is required as the representation of stakeholders by a product owner is embedded in the Agile way of working. Note that active involvement of the product owner in Agile projects is a crucial success factor but is sometimes quite challenging to realize in practice. If the latter is the case, this will be reported and discussed at the daily stand-up and managed as a project risk (see above) that will affect the whole team's effectiveness and efficiency.

2.3.2 SG2 Monitor Product Quality Against Plan and Expectations

For product quality monitoring largely the same mechanisms are used as for progress monitoring (see SG1 above). In Agile, for product risk monitoring the focus lies on a review of list of product risks in regular meetings rather than the review of any detailed documentation of risks. Newly identified product risks or changed product risks, e.g., as a result of exploratory testing, will be discussed and required testing actions will be agreed upon. The status of the various product risks is typically shown by means of charts on the dashboard.

A best practice is that no feature is considered done until it has been integrated and tested successfully with the system. In some cases, hardening or stabilization iterations occur periodically to resolve any lingering defects and other forms of technical debt. Agile teams use defect-based metrics similar to those captured in traditional development methodologies, such as test pass/fail rates, defect discovery rates, defects found and fixed, to monitor and improve the product quality. The number of defects found and resolved during an iteration as well as the number of unresolved defects possibly becoming part of the backlog for the next iteration should be monitored during the daily stand-up meetings. To monitor and improve the overall product quality, many Agile teams also use customer satisfaction surveys to receive feedback on whether the product meets customer expectations.

Testing exit criteria, e.g., for test coverage and product quality (defects), are part of the Definition of Done (DoD). The adherence to agreed exit criteria is typically monitored through the task board, whereby a story can only be indicated as "done" if it complies to its DoD criteria.

The daily stand-up meeting is the mechanism used to almost continuously perform product quality reviews. The milestone product quality review within Agile is at the completion of an iteration. Demos are organized with stakeholders to discuss the business value and quality of the product being delivered. Product quality is verified and validated against the defined DoD quality criteria.

Following the process area Test Planning, specific practices on monitoring entry, suspension and resumption criteria are likely to be non-relevant. For more information refer to the specific goal 2 Establish a Test Approach of the Test Planning process area, where an explanation has been given why these criteria are typically likely to be non-relevant in an Agile environment.

2.3.3 SG3 Manage Corrective Actions to Closure

Agile teams would very quickly notice issues such as deviations from expectations in a burn-down chart and/or lack of progression of (test) tasks and stories on the Agile task board. These and other issues, e.g., issues blocking test progress, are communicated during the daily stand-up meetings (see above), so the whole team is aware of the issues. This than trigger a team-based discussion where the issues are analyzed. The outcome could be that the baseline needs an update (e.g., removal of one or more user stories from the iteration backlog), the Definition of Done is perhaps too strict or changes should to be made to the way of working. The team will decide on corrective actions to be taken together with the product owner. In cases where the iteration backlog is changed the product owner presents an updated iteration backlog to the team and removed items from the iteration backlog are taken to the next iteration and discussed during iteration planning.

Managing corrective actions in Agile projects is primarily a responsibility of the self-organizing team. The team can define and implement appropriate corrective actions, or escalate any issues as 'impediments' to the Scrum master. The Scrum master typically has the responsibility to support the team to manage issues to closure. Typical events to discuss and manage corrective actions are daily stand-ups and retrospective meetings. Corrective actions that have been agreed can be managed to closure as ‘tasks’ or ‘backlog items’ via the product backlog or (within the iteration) via the task board.

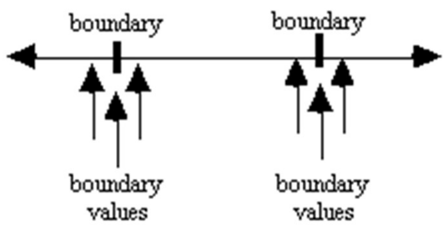
2.4 Process Area 2.4 Test Design and Execution

The purpose of Test Design and Execution is to improve the test process capability during test design and execution by establishing test design specification, using test design techniques, performing a structured test execution process and managing test incidents to closure.

Although the underlying objective (“mitigate the risks and test the software”) is the same for an Agile and a traditional sequential project, the approach taken on how to test is typically very different. In Agile, flexibility and being able to respond to change are important starting points. Also, test analysis and design, test implementation and test execution are not subsequent test phases, but rather are performed in parallel, overlapping and iteratively. The level of detail of test documentation established is another key difference. Typically more often experienced-based and defect-based techniques are used in Agile projects, although specification-based test design techniques may also still be applicable and used. With more emphasis on unit and integration testing, white-box techniques such as statement and decision testing are also much more popular. A final major difference is the level of the regression risk, which calls for more regression testing at the various test levels. Ideally regression testing is highly automated. There are many differences, but in the end in the context of the process area Test Design and Execution it’s always about creating tests, mitigation products risks, running tests and finding defects.

2.4.1 SG1 Perform Test Analysis and Design using Test Design Techniques

In Agile, test analysis and design, and test execution are mutually supporting activities that typically run in parallel throughout an iteration. In sequential lifecycle projects, test analysis takes place by testers reviewing the test basis, e.g., the requirements, and evaluating the testability of the test basis once it has been created. In Agile projects, testers are part of a team that collaboratively creates and refines user stories. Frequent informal reviews are performed while the requirements are being developed including for each user story the acceptance criteria. These criteria are defined in collaboration between business representatives, developers, and testers. Typically, the tester’s unique perspective will improve the user story by identifying missing details and making them testable. A tester can contribute by asking business representatives open-ended questions about the user story and its acceptance criteria and proposing ways to test the user story. Test analysis is thereby not an explicit separate activity, but rather an implicit activity that testers perform as part of their role within collaborative user story development.



Based on the analysis of the user stories, test conditions[1] are identified. From a testing perspective, the test basis is analyzed in order to see what could be tested – these are the test conditions. Test conditions (sometimes referred to as test situations) are basically an identification of “things” that need to tested/covered [Black, Van Veenendaal]. Provided the defined acceptance criteria are detailed and clear enough that may well take over the role of traditional test conditions. The test conditions are subsequently translated into tests, which are needed to provide coverage of the defined and agreed acceptance criteria. It is often also beneficial to perform test analysis at a higher level rather than just user stories. For example analyzing a feature or epic or a collection of stories to identify test conditions that are more abstracted than those at user story level and also span multiple user stories. Specification based test design techniques are typically helpful in deriving test

[1] The term test condition is used in this paragraph based on the TMMi practices although this is not a common Agile term. TMMi defines a test condition as “An item or event of a component or system that could be verified by one or more test cases, e.g. a function, transaction, feature, quality attribute, or structural element”. It is not intended here to impose an additional layer of test documentation to an Agile team. However, what is important is that there is a clear understanding by the team of what items (or events) need to be tested. In some projects there are many tests but without an understanding of what they are really covering and need to cover. In an Agile context the intend of the TMMi practices on test conditions is typically achieved alternative practices such as acceptance criteria (provided they specified to the required level of detail) and test charters in exploratory testing.

conditions from user stories and acceptance criteria. However, in Agile most often these test design techniques are used more implicitly than explicitly whereby the testers based on their experience have mastered the techniques and are able to use them with flexibility in context. Test conditions will be documented in a lightweight format contrary to the more traditional way of working where they are documented as part of a test design specification document. Sometimes, especially in projects where exploratory testing is widely used, test conditions are just identified by means of brainstorming and are documented as test ideas (to be part of test charters). Defining test conditions is also the basis for establishing horizontal traceability (see hereafter) and managing the coverage of the (automated) regression test set whereby automated tests are developed to cover specific test conditions.

With the test-first principle being applied with Agile, tests that cover the set of test conditions will be identified (and possibly automated) prior to, or at the least in parallel with, the development of the code. For automated unit testing an approach like Test-Driven Development can be considered. For higher test levels, Behavior-Driven Development (BDD) and Acceptance Test-Driven Development (ATDD) are popular Agile development approaches that also support testing. Both BDD and ATTD are also highly related to test automation.

For most manual testing, tests will be identified/refined as the team progresses with test execution. Tests are most often not documented in as much detail as in traditional projects, but rather in a format of test ideas using exploratory testing. For complex and critical areas a more traditional approach to test design and test case development using formal test design techniques may be the best way to cover the risks. However, also with this approach the amount of documentation will be limited compared to testing in a traditional sequential lifecycle environment. The prioritization of tests follow the prioritization of the user story they are covering. The prioritization of the user stories is based on business value; highest priority relates to highest business value. It's important that tests are established to cover functional and non-functional risks, but especially in Agile also specifically to cover the regression risk.

Specific test data necessary to support the test conditions and execution of tests is identified. However, in Agile in contrast to traditional environments, the test data needed is typically not first specified as part of a test specification document. Rather provided the necessary tools and/or functionalities are available, the test data is immediately created to allow an almost immediate start of the execution of manual tests. However, for automated tests, the data will typically be needed to be specified upfront.

Traceability between the requirements, test conditions and tests needs to be established and maintained. Teams need to make clear that they have covered the various user stories and acceptance criteria as part of their testing. While a formal requirements management tool might not be needed, the team does need support to have their requirements organized and managed in a way that supports the assignment of identifiers to each requirement so that those identifiers can be used to ensure testing is completed according the agreed criteria. In many Agile organizations, user stories are the basis for developing a set of related acceptance criteria and subsequently tests. Once these tests are established, the tests themselves often become the detailed agreed-to requirements. Using this approach may suffice to achieve the intent of the TMMi specific practice on requirements management related to horizontal traceability.

2.4.2 SG2 Perform Test Implementation

Test implementation is about getting everything in place that is needed to start the execution of tests. Typically the development of test documentation, e.g., test procedures, to support test execution is minimized. Rather automated (regression) test scripts are developed and prioritized. Also regression test preparation starts as soon as possible in parallel to other testing activities.

In an Agile environment detailed test procedures are not common practice. Working in a knowledgeable team, tests will most likely be documented at a much higher level of abstraction. This will suffice for those executing the tests since it is expected that they have the required level of domain and technical knowledge to perform the tests. Those executing the tests work within of the team, so they will have a better understanding of both what and how it is coded, and how the functionality is fit for purpose. Since the level of change within and across iterations is typically high, developing detailed test procedure would also create much maintenance work. Typically much testing is performed by using exploratory testing. In exploratory testing, no detailed test procedures are developed, but rather high level one-page test charters describing test ideas and guidance for the tester. However, specific test data that is required to perform tests needs of course to be created upfront.

Many Agile teams use automated testing. Typical approaches being used are Test-Driven Development (TDD), Behavior-Driven Development (BDD) and Acceptance Test-Driven Development (ATDD). Using one or more of these approaches automated test scripts will be created as part of the test implementation activity.

Continuous integration is a key practice for Agile projects. Testing is a parallel and fully integrated activity to be performed by the team. It is not an independent separate activity or phase. The specific practice 2.3 Specify intake test procedure therefore becomes irrelevant within an Agile team. There is no need and in fact it doesn't make sense to have a specific formal intake test being specified and performed. However, in case some testing, e.g., system integration test or hardware/software integration test, is being performed outside the scope of the Agile team, they may specify an intake test to evaluate the output quality of the Agile team and determine whether the product that has been delivered is ready for their testing.

In Agile environments no specific test execution schedule will be created and therefore specific practice 2.4 Develop test execution schedule is likely to be non-relevant. There is a high level of flexibility regarding the order in which the test will be executed throughout an iteration although this will be guided by the priority of the various user stories. The various tests to be executed will be identified during iteration planning and will be managed as test tasks through the Agile task board.

2.4.3 SG3 Perform Test Execution

The software items being developed in the iteration need of course to be tested. Following the Agile manifesto, as with the other goals in this process area, this will typically be done with much less documentation intensiveness than in a traditional sequential project. Often no detailed test procedures and test logs are produced. The tests, especially the regression tests, should as much as possible and feasible be automated.

In an Agile project, software is delivered, as a minimum, on a daily basis and testing is an integral part of the Agile development process whereby work is performed in mutual cooperation to deliver a quality product. There is no dedicated and separate development and test team. Therefore a specific intake test as defined in specific practice 3.1 Perform intake test is likely to be non-relevant in a pure Agile environment. However, when some testing is done outside the Agile team (as is often the case), an intake test will typically be performed by the test team that is responsible for testing outside the Agile team. Bear in mind there are activities expected to be completed before acceptance testing on a specific user story within the Agile team starts. These activities are typically part of the Definition of Done, e.g., agreed amount of unit testing has been completed and passed.

Test Execution is done in line with the priorities defined during iteration planning. Some tests may be executed using a documented test procedure, but typically many tests will be executed using exploratory and session-based testing as their framework. In exploratory testing, test design and test execution occur at the same time, guided by a prepared test charter. A test charter provides test objectives and test conditions to cover during a time-boxed testing session. During exploratory testing, the results of the most recent tests guide the next tests.

With iterative development there is an increased need for organize and structure regression testing. This is sometimes done manually but will preferably be done using automated regression tests and supporting tools. Regression testing at especially unit and integration testing is often part of a continuous integration process. Continuous integration is important practice within Agile software development. It's basically an automated build and test process that takes place on at least a daily basis and detects early and quickly. Continuous integration allows for running automated regression tests regularly and sending quick feedback to the team on the quality of the code and code coverage achieved.

The incidents that are found during testing may be logged and reported. There is typically a discussion within Agile projects whether all incidents found should indeed be logged. Particularly in the case where the team is co-located, testers need to talk with developers and incidents/defects that can be fixed immediately and included within the next build may not need to be logged, they just need be fixed. In Agile generally not all incidents found will be logged and managed accordingly. As stated many are just raised informally by a member of the team (who may be a tester) and are fixed immediately by the developer who injected the defect. Some data is lost, but the overhead at the same time is reduced. Some teams only log incidents that escape iterations, some log them if it can't be fixed today, some only log high priority incidents. If not all incidents found are logged, criteria must be available to determine which incidents should be logged and which ones shouldn't. Remember, the intent is to get the defect correctly fixed, not to log an incident. An example of such a criterion: “If a defect can be resolved before the next daily standup and daily build, it does not need to be documented. Any defect raised that cannot be closed within one day, or affects a stakeholder outside the team, should be documented and captured.”

Sometimes incidents are logged on the Agile task board, either as a sticker on a task or as a separate task, visualizing that it’s blocking a story and its tasks from getting completed. It is acceptable to use post-it notes on a task board in order to manage your defects and this is often the chosen method for co-located teams. The Agile team will subsequently work on the incidents and manage them to closure (see specific goal 4 of this process area hereafter). Some choose to log and document incidents found in a tool, e.g., defect management or defect tracking tool. Such a tool should than be used as lean as possible and not force any elements of test incident reports to be submitted that have no or too little added value.

It is considered a good practice also in Agile teams to log data during test execution to determine whether the item(s) tested meet their defined acceptance criteria and can indeed be labeled as “done”. This is also true when applying experienced-based techniques such as exploratory testing. Examples of information that may be useful to document are test coverage (how much has been covered, and how much remains to be tested), observations during testing, e.g., do the system and user story under test seem to be stable, risk list (which risks have been covered and which ones remain among the most important ones), defects found and other issues and possible open questions.

The information logged should be captured and/or summarized into some form of status management tool (e.g., test management tools, task management tools, task board), in a way that makes it easy for the team and stakeholders to understand the current status for all testing that was performed. Of course, it’s not just the status of testing tasks in isolation that the team needs a status on, rather it needs to know the overall status of the user story. Nothing is complete from a story perspective unless all related activities are complete, not just testing.

2.4.4 SG4 Manage Test Incidents to Closure

As stated above, in Agile environments often not all incidents found will be logged. This specific goal only applies to those incidents that are logged and thus need to be managed to closure. In principle managing incidents in Agile is simple. In case where the incident is logged on the Agile task board, either as a sticker on a task or as a separate task, it is visualized as a defect blocking a story and its tasks from getting completed. The standard Agile way-of-working will be applied as with any other impediment or task that is blocking progress. It will be discussed during stand-up meetings and assigned to be resolved by the team at which moment in time it will disappear from the task board.

In case the decision is made to defer the incident found to another iteration, they become merely another desired option (or change) for the product. Add them to the backlog and prioritize accordingly. When the priority is set high enough they will be picked up by the team in the next iteration. Note that priority is typically defined by the business but at times some priority needs to be given to incidents that relate to so-called technical debt. Some teams have dedicated iterations to clean up technical debt related incidents/defects. However, this is not a standard Agile recommended practice as debt should be paid back on a regular basis.

As an example for how incidents can be handled in an Agile environment, first imagine that the team is in the middle of an iteration. While working on one of the user stories, the team determines that it can't get that to “done”, because one of the acceptance criteria is not fulfilled, e.g., because of a defect introduced during the development. Agile teams tend to fix these sort of defects as soon as possible in the current iteration, otherwise they would carry on undone work (equivalent to lean inventory). These kind of incidents will be visualized on - and managed through the task board.

Secondly, suppose all the stories for the iteration are completed according to the Definition of Done by the end of the iteration. If during the review meeting though, while demonstrating a story, there is something which doesn't behave the way the customer would want it to. Consider that the unexpected behavior wasn't actually discussed together with the business before the iteration started. In this case, the software functionality that is produced is working, and what probably needs to be done is translate the incident into an additional story and add it to the product backlog.

It's a good practice to monitor the number of defects found and resolved during an iteration as well as the number of unresolved defects possibly becoming part of the backlog for the next iteration during the daily (stand-up) meeting and also at retrospective meetings.

In a traditional sequential lifecycle environment there is often a specific Configuration Control Board (CCB) that meets, reviews and decides on the disposition of the incident. This role in Agile is taken over by the empowered team. Of course the business representative, e.g., product owner, has an important role to play when it comes to deciding on the disposition of an incident. Handling incidents is an overhead, it's taking away focus and effort from other things. The overhead increases the more we try to formalize a process to list them, qualify them, order them and ultimately plan them. A complex incident management process like this is plain waste. Better to outline some basic principles on how incidents are handled and managed by Agile teams and act accordingly.

2.5 Process Area 2.5 Test Environment

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

With the process area Test Environment an adequate test environment is established and maintained, including generic test data, in which it is possible to execute the tests in a manageable and repeatable way. Of course also in a software development project using the Agile lifecycle, an adequate test environment is indispensable. Because of the short cycle times of an iteration the test environment needs to be highly stable and available. Problems in the test environment will always immediately have an impact on the progress and results of the iteration. Properly managing the configuration and changes to test environment and test data is therefore also of utmost importance.

Test environments come in many ways depending among others on the system under test, but typically specification, implementation and management of the test environment is done in parallel to the Agile software development project. The implementation of a test environment often takes much time and can be highly complex, making it almost impossible to perform this task within the limited cycle time of an iteration [Van der Aalst and Davis]. It also involves different stakeholders and engineers that typically make up an Agile team. The Agile methodologies have been developed to support software development, not for test environment or infrastructure development and management. It's just less suitable and the way test environment or infrastructure development and management is performed remains largely the same as within a traditional environment although there may be some changes to the process when Agile is being practiced.

2.5.1 SG1 Develop Test Environment Requirements

Specification of test environment requirements is performed early in the project. The requirements specification is reviewed to ensure its correctness, suitability, feasibility and accurate representation of a 'real-life' operational environment. Early requirements specification has the advantage of providing more time to acquire and/or develop the required test environment and components such as simulators, stubs or drivers.

In some Agile development projects a so-called initial iteration (iteration 0) is applied where the elicitation and specification of test environment requirements is performed.

2.5.2 SG2 Perform Test Environment Implementation

Sometimes (part of) the implementation takes place in the initial iteration in addition to other activities, e.g., training, high level product risk assessment and establishing definition of done. In this way one aims to achieve that even the first software development iterations already have a (partly) working test environment. Technical environmental actions or issues can be part of the product backlog.

Defining the test environment (including generic test data) could as with a sequential lifecycle start by means of a full plan, but it is often better to start with the implementation as soon as possible, and to have an initial version of the test environment available upon the start of the first iteration.

Indeed it is difficult to be generic on test environments. In some domains stubs and drivers are rapidly being replaced by service virtualization, typically allowing for a much faster test environment implementation. In some Agile projects provisioning of test environments is highly automated. Test environments can be commissioned in minutes or hours, rather than in days. There is also a lot more use nowadays of virtualization of machines, servers and services. Also the cloud can assist to provide the required environments, again speeding up the access.

2.5.3 SG3 Manage and Control Test Environments

Management and control of the test environment will take place throughout an Agile project. Sometimes the Agile team needs to set up and maintain the test environment themselves, but most often this is done by a separate unit outside the Agile team. When the activities are done by the Agile team themselves this of course implies that the team has sufficient technical knowledge to be able to perform these tasks. It also means that the tasks become part of the Agile process, e.g., they need to be addressed in a planning session, put onto the task board and possibly be discussed during daily stand-up meetings in case of any issues.

As a main conclusion for the process area Test Environment, the specific goals and practices are all still applicable and do not change in essence. The thing that changes is their timing in the lifecycle.

3 TMMi Level 3 Defined

3.1 Process Area 3.1 Test Organization

The purpose of the Test Organization process area is to identify and organize a group of highly skilled people that is responsible for testing. In addition to testing, the test group also manages improvements to the organization's test process and test process assets based on a thorough understanding of the strengths and weaknesses of the organization's current test process and test process assets.

3.1.1 SG 1 Establish a Test Organization

Test Organization is an often misunderstood process area. Many read this as the TMMi requires an independent test group or even department that does independent testing. As much as this is a possibility, there are also other organizational models that comply to the TMMi requirements. In addition to the independent test organization that prepares and executes tests, the test competence centre is also an option. Typical tasks and responsibility of a so-called test competence centre are establishing, managing and improving test processes, and providing resources to projects with testing experience, knowledge and skills. A test competence centre commonly has ownership regarding test methodologies and processes. Note that a tester on a day-to-day basis is typically part of the Agile team, but on top of that is it common for a tester to be belong to a test organization, like the test competence center or test guild.

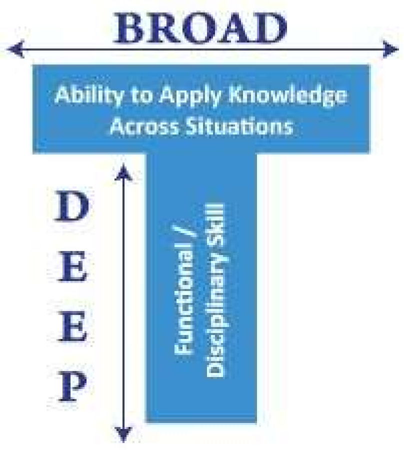
A test organization requires leadership in areas that relate to testing and quality issues. The staff members of such a group are called test specialists. The test competence centre option fits well with Agile and ensures that testing is preserved as a discipline and treated seriously as such. Of course there are Agile thought leaders that are not in favor of keeping special disciplines. While this may be a preferred option for some, doing TMMi requires treating and keeping testing as a discipline. Needless to say this should only be done if it makes sense from a business perspective and thus when this doesn't make sense TMMi is probably not the preferred improvement path.

In Agile, the test organization may take the format of a test guild. A test guild is typically an informal, self-managing group of testers, who are member of different Agile teams. Important activities of a test guild can be knowledge sharing, group wise learning, trend watching etc. Membership of a guild is on a voluntarily basis. In practice, a test guild often meet on a bi-weekly basis. A test guild can also be assigned more formal tasks like establishing test career paths, organize training and determine, plan and implement test process improvements. It is important to ensure the test guild remains largely informal and self-managing. An important constraint for a test guild to be successful is that its members are assigned time to be spend on the various test guild activities to be performed.

Independent testers are often more effective at finding defects. Balancing independence with Agile is a challenge. Some Agile teams retain fully independent, separate test teams, and assign testers on-demand during the final days of each sprint. This can preserve independence, and these testers can provide an objective, unbiased evaluation of the software. However, time pressures, lack of understanding of the new features in the product, and relationship issues with business stakeholders and developers often lead to problems with this approach. Another option is to have an independent, separate test organization where testers are assigned to Agile teams on a long-term basis, at the beginning of the project, allowing them to maintain their independence while gaining a good understanding of the product and strong relationships with other team members. In addition, the independent test organization can have specialized testers outside of the Agile teams to work on long-term and/or iteration-independent activities, such as developing automated test tools, carrying out non-functional testing, creating and supporting test environments and data, and carrying out test levels that might not fit well within a sprint (e.g., system integration testing). Note that there are also possibilities to practice independent testing within an Agile team. As long as someone else other than the creator of assets or code also tests the product then this is still independence. Therefore developers can test each other's code via code reviews and/or unit test, or business analysts that contribute to testing from an user's perspective using for example different personas. Last but not least, independence can be organized through an organizational and responsibility structure, but is probably above all having the right critical mindset.

3.1.2 SG 2 Establish Test Functions for Test Specialists

Testing is regarded as a valued profession and discipline. Test knowledge and skills are needed by the Agile team. The test specialist will provide these, and also coach other team members while performing test activities. Test functions and test career paths are defined, and supported by a test training program. The required knowledge and skills for a typical Agile tester are different than in a traditional organization. The tester will of course still need "traditional" test knowledge and skills, but in addition also knowledge and skills for test automation and to be able to perform tasks outside of testing, e.g., scripting or requirements engineering. Since an Agile tester is working in a team, soft skills are equally important as increasing technical skills. The Agile tester is a so-called T-shape tester which shall be reflected in the description of the test functions. The test organization is staffed by people who have the skills and motivation to be good tester in an Agile context. They are assigned to a specific test function. Especially in an Agile context it is important, since the whole team will perform test activities, to elaborate other non-test specific functions descriptions with expectation regarding test activities, roles and required testing knowledge and skills.


BROAD
Ability to Apply Knowledge Across Situations
DEEP
Functional / Disciplinary Skill

3.1.3 SG 3 Establish Test Career Paths

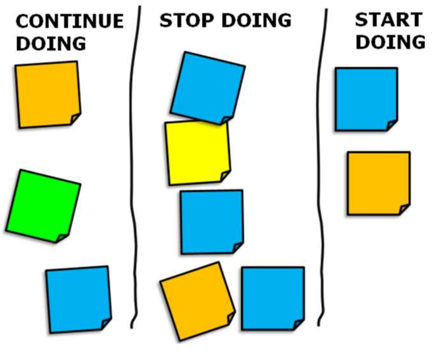
Test career paths are established to allow test professionals to improve their knowledge, skills, status and rewards, and thus allow them to advance their careers. Based on the defined test career paths, personal test career development plans are developed and maintained for every member of the test organization. This specific goal and its accompanying specific practices is also fully applicable in an Agile context. With the Agile emphasis on the people aspect, this is clearly an important specific goal. The Agile team needs testers that understand their challenging job, can coach other team members and are motivated. Many of these are addressed with test career paths.

However, the actual definition of test career paths will typically be somewhat different in an Agile organization compared to those in an organization that follow a sequential lifecycle. Whereas in a traditional organization the test career paths often have a focus around vertical growth (from tester to test manager), test career paths in an Agile organization tend to focus on a horizontal growth (from junior tester to senior tester and perhaps test coach). There is typically less need for various hierarchical functions and roles, such as test management and a higher need to grow along the roles of a Product Owner or Scrum master. Note that many of the traditional test management tasks are still relevant, but these are taken over by others (e.g, Scrum master, Agile team).

3.1.4 SG 4 Determine, Plan and Implement Test Process Improvements

The improvement process as described by this specific TMMi goal and its specific practices are largely covered by the retrospective meeting performed by an Agile team. The iteration retrospective is an opportunity for the Agile team to inspect itself and define actions for improvements to be enacted during the next iteration. A retrospective typically occurs after the iteration review and prior to the next iteration planning.

During the retrospective, the team discusses what went well in the iteration, what could be improved, and what will be committed to improve in the next iteration. During each retrospective, the Agile team identifies and plans ways to increase product quality, e.g., by improving work processes or adapting the Definition of Done if appropriate. Retrospectives can result in test-related improvement decisions focused on test effectiveness, test productivity, and quality of tests. They may also address the testability of the applications, user stories, features, or system interfaces. Root cause analysis of defects can drive testing and development improvements. Testers play an important role in retrospectives. They are part of the team and bring their unique perspective. All team members, testers and non-testers, can provide input on both testing and non-testing activities.



By the end of the retrospective, the Agile team should have identified (test) improvements that it will implement in the next iteration. Implementing these improvements in the next iteration is the adaptation to the inspection of the Agile team itself. The team votes on the most important improvements for the next iteration, so the (test) improvements may stay on the backlog for longer than one iteration, but they will eventually be implemented. It is important that the team only take on improvements that they can fit within the next iteration, so probably a maximum of two or three per iteration. Testers should take ownership of test improvements to ensure that they are implemented within the team. The retrospective is an important mechanism that allows a team to continuously evolve and improve throughout the life of a project.

Because the scope is often limited to the previous cycle or iteration, small but frequent improvements are made which focus mainly on solving specific project problems. The focus of these improvements is often not on cross-project learning and institutionalization of improvements. Looking at how (test) process improvement is organized and managed, there is likely to be less focus on a (test) process group at an organizational level and more emphasis on the self-management of teams within the project. While this is not a bad thing, it is important to also create a mechanism whereby local test improvements that are successful are shared with the rest of the organization and those improvement areas that are beyond the power of the Agile team can also be addressed. A possible solution could be to have a representative of the test organization attend local retrospective meetings to ensure that improvements are shared and institutionalized across the organization when appropriate. The test process improver can also take away broader or fundamental issues affecting testing across the organization.

3.1.5 SG 5 Deploy the Organizational Test Process and Incorporate Lessons Learned

Also for an Agile project it’s important to have access and re-use the organizational standard test process and test process assets as much as needed and has added value. The test organization will ensure these are deployed across all projects. (Refer to the Test Life Cycle and Integration process area for more information on the organizational standard test process and test process assets and how this relates to Agile projects.)

The implementation of the organization’s standard test process and the use of test process assets on Agile projects can be monitored by the self-organizing teams themselves, e.g., by addressing this in retrospectives. With a representative of the test organization, e.g., a test process improver, being present at these meetings, this person can keep track on the implementation status and address cross-project issues. This person can also take valuable lessons learned and test improvements from the Agile team back to the test organization. The lessons learned and test improvement can subsequently be submitted as test process improvement proposals and managed accordingly. The lessons learned from testing with the Agile team are thus if appropriate incorporated into the organizational standard (Agile) (test) process and test process assets.

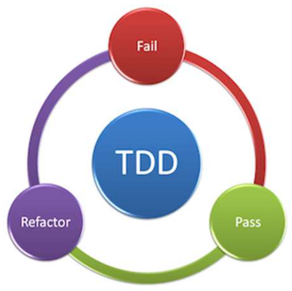
3.2 Process Area 3.2 Test Training Program

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

The main objective of the Test Training Program is to provide necessary (test) training to the testers and other team members. As a result of Agile manifesto statement “individuals and interactions *over* processes and tools”, processes in an Agile environment will typically be documented lightweight and not address all possible scenarios. Training of people however, then becomes a critical success factor that needs focus and is almost like a countermeasure for the lightweight processes. A quality training program ensures that those involved in testing continue to improve their testing knowledge and skills, and acquire the necessary domain knowledge and other knowledge and skills related to testing. Since quality and testing now are team responsibilities, it is expected that test related training is not provided to test professionals only, but rather to the whole Agile team.

3.2.1 SG1 Establish an Organizational Test Training Capability

The process starts with identifying the strategic test training needs of the organization. In addition to the traditional training needs for testers, e.g., on test design techniques and coverage measures, there is now also a need to acquire knowledge and skills on the specifics of Agile testing. Examples include the Agile manifesto and its principles, frameworks such as Scrum, Kanban and XP, Agile testing concepts (e.g., testing pyramid and testing quadrants), Test Driven Development (TDD), Behavior Driven Development (BDD), Acceptance Test-Driven Development (ATDD), user-story development including supporting the definition of acceptance criteria and reviewing them for testability, test automation and exploratory testing. Also some typical test management knowledge and skills areas such as planning, estimation (e.g., planning poker), monitoring and risk analysis (e.g., risk poker) will now become areas that need to be part of the knowledge and skill set for most testers working in an Agile team and, not just for the test manager. The identification of strategic test training needs is not limited to just the testers, it is of utmost importance to also specify the test related training needs for other members of Agile teams.



Just like in any other organization the test training needs will be translated into a (lightweight) training plan, possibly also addressing some specific project test training needs. The training plan needs to be reviewed and commitments need to be established. Note that especially in Agile some skills are effectively and efficiently imparted through informal vehicles (e.g., training-on-the-job, lunch training sessions, coaching and mentoring) whereas other skills require formal training. The appropriate approach to satisfy specific test training needs should be determined per knowledge and skills area, e.g., with exploratory testing training-on-the-job typically being an important aspect.

3.2.2 SG2 Provide Necessary Test Training

Largely speaking the specific practices for SG2 are the same in a traditional and Agile organization. Based on the organizational training plan, team members are identified that need to receive training necessary to be able to perform their test role effectively. The training is subsequently scheduled, including required resources, and conducted. This seems like an obvious statement, but some Agile organizations struggle to find time for training. Agile teams are always fully engaged and trying to meet the end-of-iteration deadline. Of course time for training needs to be addressed in iteration planning whereby the attendee will be less available for the next iteration.

As stated above informal training vehicles are often used in an Agile organization. For example, in addition to formal training, on-the job mentoring and coaching is typically continuously encouraged and employed. In fact mentoring is everyone’s responsibility and expected at all levels of an organization. Working in pairs is another commonly used vehicle by Agile teams for up-skilling and knowledge transfer. When using Scrum, it’s the Scrum masters responsibility to coach the team in Scrum practices.

Records of the test training conducted, either informal or formal, will be created and the effectiveness of the test training is assessed. A (test) training attended can also be evaluated as part of a retrospective meeting whereby a discussion takes place whether the knowledge and skills of the attendees are now more adequate for performing the test tasks.

3.3 Process Area 3.3 Test Life Cycle and Integration

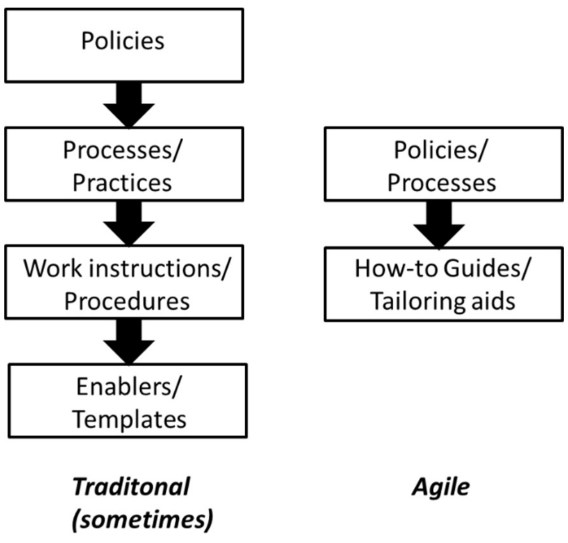
The purpose of Test Lifecycle and Integration is to establish and maintain a usable set of organizational test process assets (e.g., a standard test lifecycle) and work environment standards and to integrate and synchronize the test lifecycle with the development lifecycle. The integrated lifecycle ensures early involvement of testing in a project. The purpose of Test Lifecycle and Integration is also to define a coherent test approach across multiple test levels, based on the identified risks and the defined test strategy, and to provide an overall test plan, based on the defined test lifecycle.

3.3.1 SG 1 Establish Organizational Test Process Assets

An important responsibility of the test organization is to define, document and maintain a standard test process, in line with the organization’s test policy and goals. It amongst others, contains a description of the various test types and levels to be executed. While TMMi does not mandate full blown detailed, defined and documented test processes, often focused templates, elaborated with examples, are a great approach of establishing a common way of working. Refer also to Bob Galen's three pillars of Agile quality and testing, where he identifies standards being checklists, templates and repositories as one of the pillars of Agile software testing [Galen]. A template identifies required activities to gather information and document results implied within its structure. A process does not need to be a strict sequence of activities. When dependencies do exist, they can be captured by notes in the template, or fields in a form that are not accessible until other pre-requisite fields are completed. Templates have practical value conveying the real intent of a process. Templates also avoid the ambiguities commonly found in "wordy" process documents. As an Agile organization if you have effective processes you may just need to document them in a light way manner, and perhaps add a few things. Some key guidelines often used in Agile organization when establishing processes:

- Process "must do's" are packaged separately from guidelines
- No process is more than two pages (goal, soft rule)
- Processes do not contain "how to" information or tool information unless one has decided to mandate this across all project regardless of their size and scale.
- Separate guidelines contain tailoring/planning options, and "how-to" information. Note that with "how-to" information one can also decide to just reference to an existing book or paper.

It is not uncommon in many large organization to see four levels of process assets such as policies, processes/practices, work instructions/procedures and enablers/templates. This is not because of anything the TMMi requires, but because of the way many large organizations have chosen to implement their process assets. While the choice of process assets is up to each organization, most Agile organizations found that two levels of process assets is sufficient. This is accomplished by consolidating a policy statement with the associated process description that encapsulates "what must be done" carrying out the process. The second level contains "how to" guidelines carrying out the process and tailoring it. This level can be viewed as aids for tailoring the process, giving autonomy to the team to decide how exactly to work within the defined framework and usually includes supporting templates. In most Agile organizations step-by-step procedures are replaced by tool guides and training/mentoring. As stated above a template, such as a test charter template, test session sheet template or test plan template, can serve as a process with the required process activities implied with the template. This is a common technique being observed for developing effective Agile process descriptions. [McMahon]



Process Asset Structure

The organization's set of standard test processes can be tailored by projects to create their specific defined processes. In the context of SP 1.3 Establish tailoring criteria and guidelines most organizations today use a so-called tailor down approach. This approach requires people to spend effort explaining why something is not needed. This adds the greatest amount of work for the simplest projects, which is not an Agile-friendly approach. When you tailor down, how far can one go? Is there a minimum defined? However, if you start at a minimum set where the minimum set is what everyone must do, you eliminate the risk of tailoring out a "must do". A tailoring up approach is much more consistent with Agile approaches, and completely TMMi compliant. Tailoring is an extremely powerful mechanism for Agile organizations that wish to maintain their agility or for high mature (test) organizations seeking to increase their agility. A set of simple criteria could help to ask the right questions when tailoring up. Without criteria to help, some tend to tailor-up too much.

To support the deployment of the standard test process a test process asset library will be created and maintained, often on a wiki, in which testers can find for example templates, best practices and checklists, that will assist them in their daily work. Creating a test process asset library also makes sense in an Agile context as long as the teams get an added value when using the provided assets during their test activities. More or less the same reasoning goes for a test process database in which test data, e.g., estimates, coverage, quality measures, is collected and made available to the teams. Ensure that whatever is collected and made available to allow cross-team learning and sharing of experiences has added value for the teams.

Of course, also in an Agile context work environment standards are used to guide the creation of project work environments.

3.3.2 SG 2 Integrate the Test Lifecycle Models with the Development Models

The standard test lifecycle model defines the main activities and deliverables for the various test levels. The standard test lifecycle model is aligned with the development lifecycle models to integrate the testing activities in terms of phasing, milestones, deliverables, and activities. Lifecycle integration is done in such a way that early involvement of testing in projects is ensured. This all sounds like there is nothing to do here in the case an Agile lifecycle has been adopted; it's all already been taken care of. In Agile, development and testing should be totally integrated within the lifecycle, and testing should already be taking place at early as possible.

As much as this is true in many organization that are using Agile, there are also organizations that struggle with this aspect. Their iteration resembles a mini V-model life cycle for development where testing only starts towards the end of the iteration, is often performed under time pressure and sometimes even cut short. Of course this is not an implementation of an integrated Agile development and testing life cycle as it should be. It does however show that for some organizations that are "Agile-like" this specific goal is an area for improvement. It is important that a common understanding is created by defining how testing can be involvement and performed early in the iteration.

This specific goal is thus also applicable to an organization doing Agile software development. If done appropriately, development and testing should be fully integrated and there is a common understanding on how this is done. In such cases, nothing additional needs to happen. In other circumstances, as described above, this specific goal is a very important improvement area and should lead to much better aligned development and testing activities performed in an iteration.

3.3.3 SG 3 Establish a Master Test Plan

At TMMi level 3, testing is concerned with master test planning which addresses the coordination of testing tasks, responsibilities and test approach across all test levels. This prevents unnecessary redundancy or omissions of tests between the various test levels and can significantly increase the efficiency and quality of the overall test process. The master test plan describes the application of the test strategy for a particular project, including the particular levels to be carried out and the relationship between those levels. A TMMi level 2, (test) planning is already addressed both at release and at iteration level as part of the Test Planning process area. An Agile team typically performs multiple test levels which should all be addressed appropriately as part of release and iteration planning.

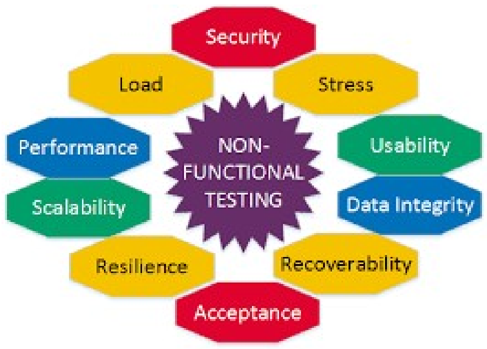
However, some projects that are working according to an Agile life cycle model have adopted a somewhat hybrid approach, in that in addition to testing within an Agile team, some testing is organized outside the team. In such circumstances a master test plan will define and manage the relationship from Agile teams to those test levels that are performed outside their scope, e.g., specific non-functional testing (security, performance), hardware/software integration testing, system integration testing or beta testing. All to this specific goal related specific practices will than typically also be applicable. Especially in larger projects and as organizations grow, documenting an overall master test plan and related decisions in a consistent way begins to have greater value. This is particularly true where an organization is using scaled Agile frameworks.

In case all testing takes place with the Agile teams, a specific master test plan has no added value, and thus this TMMi goal will likely in such cases to be non-relevant. Here planning at release and iteration level is expected to cover all necessary testing aspects.

3.4 Process Area 3.4 Non-Functional Testing

The purpose of the Non-Functional Testing process area is to improve test process capabilities to include non-functional testing during test planning, test design and execution.

Whether non-functional testing is important, is independent of the lifecycle being applied. The product being developed will dictate which non-functional aspects are relevant for testing. This of course is true for both sequential lifecycle models and Agile development models. Non-functional testing is therefore also applicable to Agile software development.



However, depending on the nature of the non-functional aspects and the test approach being used, some non-functional tests, e.g., performance and reliability, when tested in a traditional way cannot as such be tested in an short iteration. As these non-functional tests may, depending on the methods and techniques being used, take weeks to set up, organize and execute, their throughput-time is then less applicable to the heart-beat of iterations. A different approach to testing non-functional attributes is therefore often needed. This is one of the fundamental changes for the better with agile, in that quality attributes are already tested early throughout iterations, and not left until the end like with a traditional life cycle model. Not all of the components/functions may be available from iteration one to do a full non-functional test however the results of this testing can be used to give early indications of quality problems.

The user stories should address both functional and non-functional elements to insure that the right product is developed for users and customers. ISO 25010 quality characteristics can help structure the requirements and testers should consider these non-functional elements [ISO25010].

3.4.1 SG1 Perform a Non-Functional Product Risk Assessment

The product risk session(s) for Agile projects, as identified and described at the specific goal SG1 Perform Risk Assessment as part of the Test Planning process area, will now explicitly be extended to also include non-functional aspects and risks. At release level the risk assessment, now also for non-functional testing, can be performed based on the product vision. At iteration level this is performed using the user stories that define non-functional requirements as a major input.

Preferably all team members, including the product owner, and possibly some other stakeholders should participate in the product risk sessions. For some non-functional areas, specialists may be needed to assist. The product risk sessions will typically result in a documented list of prioritized (non-) functional product risks. As stated at the Test Planning process area, in agile projects both the process being used and the resulting documentation will be much more light-weight compared to a traditional project following a sequential lifecycle model.

3.4.2 SG 2 Establish a Non-Functional Test Approach

A test approach for the relevant non-functional quality characteristics is defined to mitigate the identified and prioritized non-functional product risks. For a specific iteration, the non-functional features to be tested are identified during iteration planning. The prioritized list of non-functional features to be tested typically relates to the user stories to be tested in this iteration. New non-functional product risks may become apparent during the iteration requiring additional testing. Issues like new non-functional product risks requiring additional testing are typically discussed at daily standup meetings.

The non-functional test approach that is defined at iteration level to mitigate the non-functional risks will typically cover the identification of appropriate non-functional test methods and test technique(s) based on the level and type of non-functional risks. Usually it will also address the usage of supporting tools and the approach to test automation for non-functional testing. The test approach for non-functional testing at release level will be at a much higher level and should be based on the defined test strategy at programme or organizational level, and also the defined master test plan, if that exists. (Refer to SG3 Establish a Master Test Plan within the process area Test Life Cycle and Integration for rationale to establish or not to establish a master test plan). Both the non-functional test approach at release and iteration level are part of the overall test approach and will held or displayed on the team/project wiki.

Non-functional exit criteria are part of the so-called Definition of Done (DoD). It is important that the DoD has specific criteria related to non-functional testing e.g., Mean-Time-Between-Failures (MTBF) or “front-end web pages have been tested for the OWASP top 10 risk list”. The iteration should result in the implementation of the agreed set of non-functional user stories (including their acceptance criteria) and meet the non-functional (test) exit criteria as defined in the DoD. Note that non-functional attributes can also be part of the acceptance criteria for functional user stories and do necessarily need to be specified as separate non-functional user stories. Not only will a Definition of Done exist at iteration level, often there is also a DoD at release level spanning multiple iterations. Also the DoD at release level may have non-functional related criteria.

3.4.3 SG 3 Perform Non-functional Test Analysis and Design

This specific goal largely follows the same practices, but now from a non-functional perspective, as with the specific goal SG1 Perform Test Analysis and Design using Test Design Techniques from the Test Design and Execution process area. During test analysis and design the test approach for non-functional testing is translated into tangible test conditions and tests. In Agile, test analysis and design, and test execution are mutually supporting activities that typically run in parallel throughout an iteration. This is also true for most non-functional testing. Non-functional test analysis is thereby not an explicit separate activity, but rather an implicit activity that testers perform as part of their role within collaborative user story development.

Based on the analysis of the non-functional user stories, non-functional test conditions are identified. Test conditions are basically an identification of “things” that need to tested/covered. Provided the defined acceptance criteria are detailed and clear enough, they will typically take over the role of traditional test conditions. The acceptance criteria are subsequently translated into non-functional tests. With non-functional testing it is often beneficial to perform test analysis at a higher level rather than just user stories. For example analyzing a feature or epic or a collection of stories to identify non-functional test conditions that are more abstracted than those at user story level and also span multiple user stories. With the test-first principle being applied with Agile, non-functional tests that cover the set of non-functional test conditions will be identified (and possibly automated) prior to, or at the least in parallel with, the development of the code.

For most manual non-functional testing, tests will be identified/refined as the team progresses with non-functional test execution. Tests are most often not documented in as much detail as in traditional projects, but rather in a format of test ideas when using exploratory testing. For complex and critical non-functional areas a more traditional approach to test design and test case development using formal non-functional test design techniques may be the best way to cover the risks. However, also with this approach the amount of documentation will be limited compared to non-functional testing in a traditional sequential lifecycle environment. The prioritization of non-functional tests typically follows the prioritization of the user story they are covering. However, the prioritization may also be driven by the time needed to prepare and perform a certain non-functional test.

Specific test data necessary to support the execution of non-functional tests is identified. In Agile the test data needed is typically not first fully specified in a test specification document. When specified, it is often recorded as part of the user story. However, provided the necessary tools and/or functionalities are available, the test data needed to support the non-functional tests is most often created instantly to allow a prompt start of the execution of non-functional manual tests. In contrast, for automated non-functional tests, the data typically needs to be specified upfront.

Traceability between the non-functional requirements, test conditions and tests need to be established and maintained. Teams need to make clear that they have covered the various non-functional user stories and acceptance criteria as part of their testing. In many Agile organizations, user stories are the basis for developing a set of related acceptance criteria and subsequently tests. Using this approach horizontal traceability from requirements to test can be achieved.

3.4.4 SG 4 Perform Non-functional Test Implementation

Test implementation is about getting everything in place that is needed to start the execution of tests. Typically the development of test documentation, e.g., test procedures, to support test execution is minimized. Rather automated (regression) test scripts are developed and prioritized.

Non-functional test implementation will follow many of the practices already described at the specific goal SG2 Perform Test Implementation as part of the Test Design and Execution process area.

What specifically needs to be done, and how non-functional test implementation is done largely depends on the approach defined, techniques being used and which non-functional characteristics need to be tested and to what level. For example, exploratory testing, requiring less preparation, can be suitable for usability testing but is often less suitable for extensive reliability and performance testing.

For some non-functional quality characteristics the availability of test data is essential and needs to be created during test implementation. This is largely the same as with traditional projects, apart from them maybe being documented in a brief manner thereby still allowing for re-use for regression testing.

Of course, test implementation and preparation will start as soon as possible in parallel to other (testing) activities. It's not a separate phase, but rather a set of activities (listed on the task board) that need to be performed to allow for an efficient and effective test execution.

3.4.5 SG 5 Perform Non-functional Test Execution

As with the previous goal in this process area, we will largely refer back to the related specific goal SG 3 Perform Test Execution part of the Test Design and Execution process area discussed earlier in this document. The practices for execution of non-functional tests, reporting test incidents and writing test log's in an Agile environment are basically the same as for functional testing in an Agile environment. It will typically be done with much less documentation intensiveness than in a traditional project. Often no detailed test procedures and test logs are produced.

Non-functional test execution is done in line with the priorities defined during iteration planning. Some tests may be executed using a documented test procedure, but typically also many non-functional tests will be executed using exploratory and session-based testing as their framework. With iterative development there is an increased need to organize and structure regression testing. This is sometimes done manually but will preferably be done using automated regression tests and supporting tools. Regression testing, of course, also applies to the non-functional aspects of the system that have been identified as important to test.

The non-functional incidents that are found during testing may be logged and reported by the team. There is typically a discussion within Agile projects whether all incidents found should indeed be logged. Some teams only log incidents that escape iterations, some log it if it can't be fixed today, some only log high priority incidents. If not all incidents found are logged, criteria must be available to determine which incidents should be logged and which ones shouldn't. Sometimes incidents are logged on the Agile task board, either as a sticker on a task or as a separate task, visualizing that it's blocking a story and its tasks from getting completed. Some choose to log and document incidents found using a tool, e.g., defect management or defect tracking tool. Such a tool should than be used as lean as possible and not force any elements of non-functional test incident reports to be submitted that have no or too little added value.

It is considered a good practice also in Agile teams to log data during non-functional test execution to determine whether the item(s) tested meet their defined acceptance criteria and can indeed be labelled as "done". The information logged should be captured and/or summarized into some form of status management tool (e.g., test management tools, task management tools, task board), in a way that makes it easy for the team and stakeholders to understand the current status for all testing that was performed.

3.5 Process Area 3.2 Peer Reviews

The purpose of the Peer Review process area is to verify that work products meet their specified requirements and to remove defects from selected work products early and efficiently. An important corollary effect is to develop a better understanding of the work products and of defects that might be prevented.

3.5.1 SG 1 Establish a Peer Review Approach

Agile teams typically do not do formal peer reviews in the sense that they usually don't have single defined time when people meet up to provide feedback on a product. However, they do achieve the intent of Peer Reviews by doing continual less formal peer reviews throughout the development. This is a common practice in many Agile organizations. However there still needs to be a discipline when conducting these activities. This TMMi goal covers the practices for establishing a peer review approach within a project. A review approach defines how, where, and when review activities should take place and whether those activities are formal or informal. Establishing a peer review approach is also applicable to Agile projects, however typically the review techniques applied and the way reviews are organized is very different.

Examples of peer reviews typically performed within Agile projects:

- having refinement / grooming sessions on the specifications (e.g., user stories) with the team and business stakeholders on a regular basis throughout an iteration;
- daily meetings with other team members to discuss openly the work products, e.g., code or tests, being developed and providing feedback.
- the demonstration of products early and often to customers, at least at the end of an iteration during the iteration review;

Poor specifications are often a major reason for project failure. Specification problems can result from the users' lack of insight into their true needs, absence of a global vision for the system, redundant or contradictory features, and other miscommunications. In Agile development, user stories are written to capture requirements from the perspectives of business representatives, developers, and testers. In sequential development, this shared vision of a feature is accomplished through formal reviews after requirements are written; in Agile development, this shared vision is accomplished through frequent informal reviews while the requirements are being established. These informal review sessions are often referred to as backlog refinement or backlog grooming sessions. During refinement meetings the business representative and the development team (and stakeholder if available) use review techniques to find the needed level of detail for implementation and to clarify open issues.

Almost continuous reviewing by the team on work products being developed is part of the whole team approach. They are performed with the intent of identifying defects early and also to identity opportunities for improvement. Agile methods and techniques like XP or pairing also include peer reviews as a core practice to create feedback loops for the teams. Of course in case of high complexity or risk the team can opt to apply a semi-formal or formal review technique, e.g., inspection. In these cases there is a clear rationale for spending effort on a more formal review and applying a more disciplined way of working. A part of the review approach criteria may be defined when to use a more formal review technique.

While requirements engineering emphasizes requirements validation via methods like informal reviews, walkthroughs, inspections or perspective-based reading, Agile methods rather strive to validate requirements through often and early feedback to quickly implement valuable product increments. The need for early formal validation is reduced by showing quick results in the form of integrated product increments. If the increment does not completely meet the requirements of the stakeholders the delta is put back into the product backlog in the form of new requirements and prioritized with all other backlog items. A demonstration of what actually got built during an iteration is simply a very efficient way to energize a validation-based conversation around something concrete. Nothing provides focus to the conversation like being able to actually see how something works. The demonstration is an activity that is performed during the iteration review. Features are demonstrated to and discussed with stakeholders, necessary adaptations are made to the product backlog or release plan to reflect new learnings from the discussion.

3.5.2 SG 2 Perform Peer Reviews

Of course like any approach, it should not only exist as a defined and agreed upon approach, it should be adhered to. The team needs to spent a substantial amount of time on backlog refinement sessions during an iteration, applying informal (and possibly) formal reviews as part of their daily routine, and have stakeholder demos on a regular basis. Stakeholder / customer demonstrations are expected at least at the end of each iteration.

Entry criteria are typically only applicable with formal reviews, and therefore likely to be less or non-relevant within Agile projects. However, the usage of exit criteria is applicable and has added value. INVEST [Wake] is an example of a set of exit criteria that are commonly referred to in Agile project when reviewing and updating user stories. INVEST is an acronym which encompasses the following concepts which make up a good user story:

I independent
N negotiable
V valuable
E estimable
S small
T testable

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable.

Of course other quality related exit criteria exist and may be used beneficially as well.

It's important that the tester, being part of the Agile team, participates in review sessions. This is especially true for sessions in which work products are discussed and reviewed that are used as a basis for testing throughout the iteration, e.g., at backlog refinement sessions. It is recommended to have at least one developer and one tester present when refining the backlog, to ensure alternate viewpoints of the system are present. Typically, the tester's unique perspective will improve the user story by identifying missing details or non-functional requirements. A tester can especially support the identification and definition of acceptance criteria for a certain user story. A tester contributes by asking business representatives open-ended "what if?" questions about the user story, proposing ways to test the user story, and confirming the acceptance criteria.

Specific practice 2.3 Analyze peer review data is especially relevant with formal reviews. With formal reviews much effort is spent, to ensure the effort is spent both efficiently and effectively peer review data is gathered and communicated to the team in order to learn and tune the review process. As stated, formal reviews, e.g., inspection, are less common in Agile projects. Whereas gathering data is an essential part of formal reviews, it's much less common with informal review. As a result, one may argue that detailed peer review data gathering, analyses and communication is less or even not relevant in an Agile context. In the context of deciding to collect data when doing peer reviews, ask the following questions:

- Who is going to use this data if it is being collected?
- How does this data relate to our business objectives?

If no one is going to use the data meaningfully, then don't waste valuable resources collecting it. As a conclusion, for most Agile project Specific practice 2.3 Analyze peer review data will be considered as likely being non-relevant. Note that some basic valuable review data will still be collected and used in the context of the generic practices, e.g., GP 2.8 Monitor and Control the Process.

4 TMMi Level 4 Measured

As stated earlier in this document, it is important to remember there is a natural break between TMMi level 3 and TMMi levels 4 and 5. There continues to be significant controversy over the value of moving an organization to TMMi level 4 and 5. Especially Agile organizations are recommended to critically choose and only do the practices at TMMi levels 4 and 5 that matter and have added value. Whilst TMMi is comprehensive, to be successful, organizations must identify the key testing practices and improvements requiring focus. Interesting, Jeff Sutherland, co-founder of Scrum, has published a paper on the value of using Scrum together with practices of higher maturity levels [Sutherland *et al*]. Hereafter information will be provided regarding TMMi levels 4 and 5 on how one can gain value of these higher TMMi levels and its practices by using them less formally along with an Agile approach.

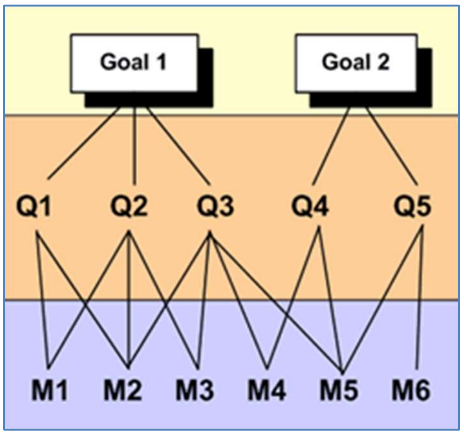
Firstly somewhat contrasting with what has been stated, one may also consider already selectively using TMMi level 4 and 5 practices with Agile techniques to address key business objectives, while trying to achieve TMMi level 2 or 3. You do not need to wait and you shouldn't. Be flexible and do not get stuck on TMMi level 2 and 3 testing practices. Use the model based on your business drivers and use process areas, goals and practices that have the most value. As such, use TMMi in a more continuous mode to some extent, and don't use the maturity levels too strictly.

4.1 Process Area 4.1 Test Measurement

The purpose of the Test Measurement is to identify, collect, analyze and apply measurements to support an organization in objectively evaluating the effectiveness and efficiency of the test process, the productivity of its testing staff, the resulting product quality and the results of test improvement. As such, the test organization will develop and sustain a test measurement capability that is used to support management information needs.

Agile approaches support in gaining commitment of those who must perform and accurately measure work in projects, However, Agile approaches don't tend to provide much support when it comes to stepping back to ensure the right measures are used across the organization, and ensuring those measures are aligned with measurement objectives derived from organization's business needs. These are strengths the TMMi test measurement process area can bring to Agile organizations.

4.1.1 SG 1 Align Test Measurement and Analysis Activities



Too often measurement programs seem to be about collecting as much data as possible. A measurement program should be highly focused, only things that matter should be collected. Start by defining clear objectives for the measurement program and revisit those on a regular basis, e.g., "It should allow us to be more successful at testing in our projects". Next discuss with stakeholders which test metrics will truly support these objectives. Based on this brainstorm a limited set of core metrics will be defined.

Goal-Question-Metric (GQM) [Van Solingen and Berghout] is a highly practical method that will support this approach. Specific context relevant measures are needed based on business needs. Company standard metrics are most often insufficient for real (test) process improvement. The purpose of measurements is to guide decisions. Use small empowered teams to derive meaningful measures and then review and refine them in short cycles.

With a focused test measurement program starting from added (business) value, the TMMi Test Measurement process area is totally in line with the Agile principle of simplicity. Ensure there is a lean test measurement and data collection strategy within the organization. It's worth pointing out that one can evolve a test measurement program by starting with just a few focused measurement objectives and a few key measures, and expand or modify them later as business needs evolve and change.

Note that within Agile the focus will be more team-based and systems thinking. This may result in a corresponding broadening of some of the metrics to the team and overall system rather than being confined solely to the specifics of testing itself. This may well lead to more challenges at the measurement analysis phase.

Of course the metrics that will be defined and for whom data will be collected will depend on the measurement objectives, but some examples of typical testing metrics for agile teams are:

- Defect cycle time; Agile teams should strive to fix defects as quickly as possible. In fact, one of the main aims of the collaborative Agile approach is to fix defects quicker so that software is released sooner.
- Defect spill-over (Number of defects deferred to a future release); Agile teams aim to produce working software with each iteration. Defect spillover measures the defects that don't get fixed during a given iteration or sprint by simply counting the defects remaining at the end of each sprint or iteration. This is, in some Agile organizations, referred to as technical debt.
- Number of defects found during an iteration
- Number of defects found after release to production/customers
- Number of defects found by people outside of the team
- Number of customer support requests
- Percentage of automated test coverage
- Percentage of code coverage.

4.1.2 SG 2 Provide Test Measurement Result

Once the relevant measures are defined based on business needs, the specific practices within this specific goal are typically all applicable in largely the same way as within a traditional organization. The only difference being that this will be done with a different mindset keeping the practices as lean as possible.

The necessary test measurement data is collected and checked for completeness and integrity, subsequently the data is analyzed as planned and the results are communicated to all relevant stakeholders.

4.2 Process Area 4.2 Product Quality Evaluation

The purpose of the Product Quality Evaluation is to develop a quantitative understanding of the quality of products and thereby support the achievements of specific projects' product quality goals.

4.2.1 SG 1 Measurable Project Goals for Product Quality and their Priorities are Established

In essence there is no real difference to identifying and setting measurable prioritized goals for product quality in an Agile context compared to doing this in a traditional environment. The overall objective is to contribute to satisfying the needs and desires of customers and end users for quality products. There are multiple options for doing this in an Agile context. Some use Definition-of-Done to define measurable goals for product quality, others use features, epics or user stories to define them whereby they will also use acceptance criteria to make them measurable. The choice (way of working) depends amongst other things on the nature of product quality goal. Refer to [ISO25010] for an overview of quality attributes.

Definition of Done (DoD) is a comprehensive checklist of necessary team activities and/or work products that ensure that only truly done user stories are delivered, not only in terms of functionality but in terms of product quality as well. The advantage of a stating product quality goals within the DoD is that it keeps them visible to and therefore actionable by the entire team as and when the increments get developed. It is obvious that the quality goals will need to be globally applicable to the application for it to be mentioned within the DoD. If the quality goals are not globally applicable, probably the best alternative way of working is to define them as a feature or epic.

Sometimes initially “only” a high level list of relevant product quality attributes is specified at release level. These will be gradually elaborated upon during refinement session as a feature, epic or user story. Subsequently at the start of an iteration the team will consider whether one or more of the specified product quality goals will be part of the objective for that specific iteration.

4.2.2 SG 2 Actual Progress towards Achieving the Project's Product Quality Goals is Quantified and Managed

Once the quantitative product quality goals have been set, they will be monitored, controlled and possibly adjusted when necessary. Since the product quality will be defined as part of the DoD and/or as specific features, epics or user stories with measurable acceptance criteria, the same methods and techniques can be used for tracking and managing progress towards achieving the product quality goals as already explained and described in the process area 2.3 Test Monitoring and Control at TMMi Level 2. Progress will be measured daily and discussed through the daily standup meeting reporting on progress, quality and risk. In addition Product Quality Evaluation is typically also supported by the Test Measurement process area which provides a measurement infrastructure.

4.3 Process Area 4.3 Advanced Reviews

The purpose of the Advanced Reviews, building on the practices of the TMMi level 3 process area Peer Reviews, is to measure product quality early in the lifecycle and to enhance the test strategy and test approach by aligning peer reviews (static testing) with dynamic testing.

4.3.1 SG 1 Coordinate the Peer Review Approach with Dynamic Test Approach

Whereas in traditional projects static testing, e.g., reviews, is most often not linked to the dynamic testing being done by an independent test team, this is expected to be totally different in an Agile context. In an Agile environment the approach to ensuring product quality should be one in which all testing related activities, both static and dynamic, are organized as one coherent and coordinated approach. It's of course the responsibility of the Agile team to integrate static and dynamic testing and find the optimal mix. The Agile team is expected to take the lead in establishing coherent and effective integrated approach. The approach for static testing (peer review) will thus be aligned and coordinated with the approach for dynamic testing.

4.3.2 SG 2 Measure Product Quality Early in the Lifecycle by Means of Peer Reviews

At TMMi level 4, the organization sets quantitative goals for software products and related work products. Peer reviews play an essential role in achieving these goals. Whereas at TMMi level 3 peer reviews are mainly performed to find defects, the emphasis is now on measuring product/document quality to control product quality early in the lifecycle. The review practices are enhanced to include practices like sampling, applying exit criteria, and prescribing rules. Indeed this is what Tom Gilb calls Agile inspection, it is all about shifting emphasis from cleanup to sampling, measurement, motivation and prevention of defects [Gilb]. This is clearly a practice that will fit with a high maturity Agile organizations. Another example would be to use the INVEST rules [Wake] for user stories as a Definition of Ready (DoR) and to measure the quality of the user stories by reviewing them against these rules. Having a Definition of Ready means that stories must be immediately actionable.

4.3.3 SG 3 Adjust the Test Approach Based on Review Results Early in the Lifecycle

When peer review results and dynamic testing are coordinated, early review results and data can influence product risks and the test approach. As already stated at specific goal 1 of this process area, in Agile, quality is a team effort and the results of verification and validation based activities will be discussed at team meetings. The principle of using early testing results to influence successive testing activities, therefore has an almost perfect match with the way-of-working within Agile teams and projects.

Product Risk Matrix
Likelihood
Impact

5 TMMi Level 5 Optimization

The goals and practices at TMMi level 5 are derived from high maturity quality management and process improvement approaches. At this high level of maturity the goals and practices are almost independent of the software lifecycle that is being applied. Of course a lean Agile based organization will approach things differently than traditional hierarchical based organizations, but the essential practices around quality control, process optimization and defect prevention will largely remain unchanged. The rationale for this is that these practices are considered to be quality management practices, typically to be largely performed at a higher level than the operational product development level. These practices are thus not practices directly related to the primary activities of software development and/or testing within a project or product delivery stream.

5.1 Process Area 5.1 Defect Prevention

The purpose of the Defect Prevention is to identify and analyze common causes of defects across the development lifecycle and define actions to prevent similar defects from occurring in the future.

Agile organizations typically already have a focus on defect prevention, which is an important foundation upon which the specific goals and specific practices of this process area can be build. Efficient agile organizations focus on defect prevention rather than only on defect discovery. A culture of defect prevention includes separating "work in process" defects, from defects escaping beyond the iteration that user stories are developed in.

The agile retrospective is a practice for teams to reflect, learn, and to continuously become better in what they do. Although retrospectives are most often used to explore the current way of working, they can also be used to investigate quality issues or to agree upon actions that can improve the quality of the software that is delivered.

The process area Defect Prevention addresses the practices for identifying and analyzing common causes of defects across teams, products and value streams, and defining specific actions to remove the common causes of those types of defects in the future. All defects, whether found during development, testing or in the field, are within the scope of the process area. Since defect prevention at this level needs measurement data, defect prevention builds upon the TMMi level 4 measurement practices and available measurement data regarding development, testing and product quality.

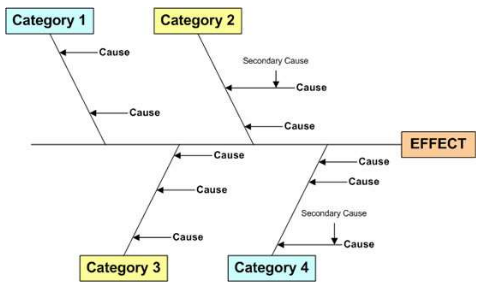
There are basically two distinct types of defect prevention activities in Agile organizations: team level and cross-team level. The team related defect prevention activities are the responsibility of the team and those have already been addressed at TMMi level 3, the cross-team defect prevention activities are performed by a Test Process Group or test guild and these are the ones that are specifically addressed by this process area..

5.1.1 SG 1 Determine Common Causes of Defects

Within Agile improvements, including test process improvements, generally take place based on frequent feedback loops. Because the scope is often limited to the previous iteration, small but frequent improvements are made that focus mainly on solving specific local problems. The focus of these improvements is often not on cross-team learning and institutionalization of improvements. Solving problems only on a team level could also easily lead to suboptimization and losing touch with the bigger picture. This is where this TMMi process area adds value, building on the existing Agile practices, defect prevention is now also applied at organizational level (across teams, products and value streams) and typically managed by a (Test) Process Group or test guild (see paragraph 3.1.1).

The selection of defects to be analyzed should be based on various factors including risk and business value. Focus needs to be given to those areas where prevention of defects has the most added value (usually in terms of reduced cost or risk) and/or where the defects are most critical. Although the defect prevention process area is commonly used for defects, it can also be used for problems such as issues around velocity.

Although a Test Process Group may co-ordinate the defect prevention activities, this should be done by a multidisciplinary team, e.g., with representatives from requirements engineering, system engineering and/or software development, as improvement actions will often affect other disciplines. Of course, this is already comon in Agile delivery methods and teams. Ensure however, that all required disciplines are involved, as per improvement actions.



Analytical methods for evaluating the root causes of defects, such as cause-effect diagrams, fishbone diagrams or the 5 why's, are typically used for defect prevention and already common Agile practices.

All three specific practices (1.1 Define defect selection parameters, 1.2 Select defects for analysis and, 1.3 Analyze root and common causes of selected defects) as defined by the TMMi model for this specific goal are also applicable to an Agile organization.

5.1.2 SG 2 Prioritize and Define Actions to Systematically Eliminate Common Causes of Defects

There are no real differences in performing the specific practices under this specific goal, whether being practiced in an Agile or sequential lifecycle. Of course the actual solution proposed and subsequent improvement proposal will be different since it needs to fit with the Agile way of working, e.g. by creating additional backlog items instead of formalized action proposals with approvals. Appropriate actions are taken to minimize the likelihood of the problem recurring. Often this means combining process improvement with additional mentoring and/or training for team members and other staffing. One needs to address both people and process issues in an integrated way, rather than by trying to artificially separate them.

Also Agile organizations are build upon self-managing teams. This implies that teams can choose which improvements to implement and/or to reject. As such during prioritzation and defining actions, one has to involve an adequate representation of the teams to discuss whether these improvements indeed provide added value to them.

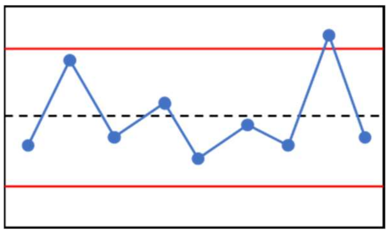
Also the way the improvements are deployed will be different, it will be less top-down and more like an offering on a voluntary basis to the self-managing teams. More on deployment of test improvements can be found at the process area Test Process Optimization hereafter.

5.2 Process Area 5.2 Quality Control

The purpose of the Quality Control is to statistically manage and control the test process. At this level, test process performance is fully predictable and stabilized within acceptable limits. Testing at project level is performed using statistical methods based on representative samples in order to predict product quality and make testing more efficient.

5.2.1 SG 1 Establish a Statistically Controlled Test Process

The principles and essentials for quality control are the same regardless of the lifecycle method being applied. Process quality control involves establishing objectives for the performance of the standard test process. Based on the measurements taken on test process performance from the teams, analysis takes place and adjustments are made to maintain test process performance within acceptable limits. When the test process performance is stabilized within acceptable limits, the defined test process, the associated measurements and the acceptable limits for measurements are established as a baseline and used to control test process performance statistically. The test process capability of the organization's standard test process, i.e., the test process performance a team can expect to attain, is now fully understood and known. As a result, the deviations from these expectations can be acted upon in a team early and consistently to ensure that they perform within the acceptable limits.



It is important to select subprocesses for statistical control that are critical to the business and have experienced problems in the past. Experience indicates that most often subprocesses that best meet these criteria are not isolated to individual teams, but rather cross multiple teams, projects and product delivery streams. When using Agile methods and techniques the work is partitioned into short iterations, where the work completed in each iteration involves performing multiple development and testing activities. Therefore as a result, Agile development supports, more effectively than traditional development, the monitoring of the most valuable subprocesses for statistical control since cycles are much shorter leading to almost instant feedback on test process performance. Fast feedback is typically highly valuable to teams and an important Agile principle.

5.2.2 SG 2 Testing is Performed using Statistical Methods

Product quality control builds on operational profiles [Musa] and usage models of the product in its intended environment to make statistically valid inferences resulting in a representative sample of tests. This approach, especially useful at the system level, uses statistical testing methods to predict product quality based on this representative sample. In other words, when testing a subset of all possible usages as represented by the usage or operational profile, the test results can serve as the basis for conclusions about the product's overall performance.

This is basically an approach that fits well with Agile, because it is lean by nature and is driven by feedback from operational use and review sessions regarding the product. However, it does require an Agile organization to have the right level of maturity to be able to apply this approach. It also strongly builds and depends on measurements and data gathered and being available.

An organization that uses statistical methods in testing is able to quantify confidence levels and trustworthiness (refer to the TMMi framework for more information on these terms). Both the level of confidence and trustworthiness are typically used as Definition-of-Done when applying statistical testing.

5.3 Process Area 5.3 Test Process Optimization

The purpose of Test Process Optimization is to continuously improve the existing testing processes used in the organization and to identify new testing technologies (e.g., test tools or test methods) that may be appropriate and to transition them into the organization in an orderly manner. Test Process Optimization also supports the re-use of test assets across the organization. The improvements support the organization's product quality and test process performance objectives as derived from the organization's business objectives.

5.3.1 SG 1 Select Test Process Improvements

Test improvement proposals are collected and analyzed. The test improvement proposals are from internal sources and typically cross-team, cross-project and cross-product stream. Internal sources include issues and ideas coming from defect prevention activities, team reviews, retrospectives, chapters and guilds. After an analysis for costs and benefits, the interesting test improvement proposals are piloted to assess new and unproven major changes before they are organization-wide deployed across the teams, as appropriate. Finally a selection is made of test improvements for deployment across the organization.

5.3.2 SG 2 New Testing Technologies are Evaluated to Determine their Impact on the Testing Process

As part of test process optimization, the centrally organized test process group (or test guild) pro-actively searches the market for new technologies, such as tools, methods, techniques or technical innovations, that can improve the testing capability of the Agile teams. By maintaining an awareness of test-related technology innovations and systematically evaluating and experimenting with them, the organization, in consultation with team representatives, selects appropriate testing technologies to improve the quality of its products and the productivity of its testing activities. Pilots are performed within the teams to assess new and unproven testing technologies before they are incorporated into standard practice.

5.3.3 SG 3 Deploy Test Improvements

Test improvements and appropriate new testing technologies are deployed across the Agile teams to improve testing. Their benefits are measured and information about new innovations is disseminated across the organizations. There is of course a difference between the deployment in a traditional versus an Agile organization. In a traditional organization the deployment most often result in new practices that shall be applied by all those involved. In an Agile organization the team has more autonomy and is self-managed. This also implies that the deployment process is different. Test improvements and appropriate new testing technologies are incorporated into the organizational test process assets as appropriate, training and support are organized and offered. However, in the end the Agile team themselves should decide whether or not the test improvement will become part of their way-of-working.

5.3.4 SG 4 Establish Re-Use of High Quality Test Assets

At TMMi level 3 some re-use of testware across teams, projects and product streams may already take place; however, re-use of test assets becomes a major goal at TMMi level 5. Team should no reinvent the wheel, but rather draw on the expertise and assets already available and as such save effort and time. Test assets of high quality that can probably be re-used throughout the organization are identified, e.g., at retrospective meetings, lessons learned sessions, discussion within the test guild, or test assessments. Subsequently re-usable test assets are selected and added to the central test repository in a format that is modifiable for re-use in future product delivery and projects.

6 Overview Applicability TMMi Specific Goals and Practices

6.1 TMMi Assessments

To assist lead-assessors and assessor in their assessments an overview is provided in this chapter of the applicability of process areas, specific goals and specific practices in an Agile context. In preceding the chapters a limited of specific practices have been identified that are expected to be less or perhaps even non-relevant. These specific practices are listed hereafter. During a TMMi assessment, a lead-assessor or assessor will need to explicitly evaluate these items for their relevance in the specific Agile context. In case it is decided that an item is indeed less or even non-relevant this is expected to be clearly documented with rationale both in the processes of the organization and in the assessment report.

6.2 TMMi level 2 Managed

Process Area 2.2 Test Planning

SG2 Develop Test Approach	Applicable
- *SP2.3 Define entry criteria*	*Likely less/non relevant*
- *SP2.5 Define suspension and resumption criteria*	*Likely less/non relevant*
SG3 Establish Test Estimates	Applicable
- *S3.2 Define test lifecycle*	*Likely less/non relevant*
SG5 Obtain commitment to the Test Plan	Applicable
- *SP5.2 Reconcile work and resource levels*	*Likely less/non relevant*

Process Area 2.3 Test Monitoring and Control

SG2 Monitor Product Quality Against Plan and Expectations	Applicable
- *SP2.3 Monitor entry criteria*	*Likely less/non relevant*
- *SP2.5 Monitor suspension and resumption criteria*	*Likely less/non relevant*

Process Area 2.4 Test Design and Execution

SG2 Perform Test Implementation	Applicable
- *SP2.3 Specify intake test*	*Likely less/non relevant*
- *SP2.4 Develop test execution schedule*	*Likely less/non relevant*
SG3 Perform Test Execution	Applicable
- *SP3.1 Perform Intake Test*	*Likely less/non relevant*

6.3 TMMi level 3 Defined

Process Area 3.5 Peer Reviews

SG2 Perform Peer Reviews	Applicable
- *SP2.3 Analyze peer review data*	*Likely less/non relevant*

6.4 TMMi level 4 Measured

All TMMi level 4 process area, specific goals and specific practices are applicable. No specific practices are indicated as likely less/non relevant

6.5 TMMi level 5 Optimization

All TMMi level 5 process area, specific goals and specific practices are applicable. No specific practices are indicated as likely less/non relevant

References

[Black, Van Veenendaal] Rex Black, Erik van Veenendaal and Dorothy Graham (2012), Foundations of Software Testing ISTQB Certification (3rd edition), Cengage

[Cohn] M. Cohn (2009), Succeeding with Agile*: Software Development using Scrum*, Addison-Wesley

[Galen] R. Galen (2015), *Three Pillars of Agile Quality and Testing: Achieving Balanced Results in your Journey Towards Agile Quality*, RGCG, LLC

[Gilb] T. Gilb (2004), *Agile Specification Quality Control,* International Council on Systems Engineering (INCOSE)

[ISO25010] ISO/IEC 25010 (2011), *Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) System and Software Quality Models*, International Organization for Standardization

[Sutherland et al] C.R. Jakobsen, K. Johnson and J. Sutherland (2007), *Scrum and CMMI level 5: The Magic Portion for Code Warriors* in: IEEE Agile 2007 Conference, Washington, D.C.

[McMahon] Paul. E. McMahon (2011), *Integrating CMMI with Agile Development – Case Studies and Proven Techniques for Faster Performance Improvement,* Addison Wesley

[Musa] J. Musa (1998), Software Reliability Engineering Testing, McGraw-Hill Education

[Van der Aalst and Davis] L. van der Aalst and Cecile Davis (2013), TMap NEXT in scrum – Effective testing in agile projects, Sogeti Nederland B.V.

[Van Solingen and Berghout] R. van Solingen and E. Berghout (1999), *The Goal/Question/Metric method,* McGrawHill

[van Veenendaal] E. van Veenendaal (2014), PRISMA: Product Risk Assessment for Agile Projects, in: *Testing Experience*, Issue 04/12, December 2012

[Wake] B. Wake (2003), *INVEST in Good Stories, and SMART Tasks*, xp123.com/articles/invest-in-good-stories-and-smart-tasks