TMMi® FOUNDATION

Testing AI-systems and TMMi

#SuccessWithTMMi

EDITOR: ERIK VAN VEENENDAAL

www.tmmi.org

TMMi is intended to be independent of the lifecycle being applied or the type of system being tested. This document explains how the TMMi test improvement model can be used and applied beneficially when testing AI systems. The level 2 TMMi process areas and their goals are discussed one by one. It is stated how these relate to testing AI systems and what the practices will typically look like. If a practice is not expected to be performed when testing AI systems, because it has no added value, this is also explicitly stated. Future versions of this document will also address the process areas at higher TMMi levels. With TMMi level 3, testing of quality characteristics specific for AI-based systems such as flexibility, adaptability, autonomy and bias will be discussed. Note that this document is not intended to be a full testing AI systems syllabus, but rather a document that "only" shows how the TMMi goals should be interpreted when testing AI systems and provide ideas and directions for further study. The document Testing AI systems and TMMi also not intended to be a description, replacing the full TMMi model. Testing AI systems and TMMi should always be used in conjunction with the TMMi reference model document. This document is a supplement to, not a replacement for, the TMMi reference model. Readers should consult the TMMi reference model for complete goal and practice descriptions.

Contributors

Amanda Dean (Australia)
Mattijs Kemmink (The Netherlands)
Vipul Kocher (India)
Poonam Jain (India)
Qin Liu (China)
Srinivas Padmanabhuni (India)
Çiçek Tuna (Germany)
Erik van Veenendaal (The Netherlands)
Kelly Ye (Australia)

Revisions

This section summarizes the key revisions between releases this document.

This section is provided for information only.

Release	Revision Notes
V1.0	Initial version of the “Testing AI-systems and TMMi” document.

Table of Contents

1 Testing AI-based systems using TMMi

1.1 Artificial Intelligence

Artificial Intelligence (AI) stands at the forefront of one of the most transformative technological revolutions. It is a field of computer science that seeks to ingrain machines with the ability to think, learn, and adapt in ways that were once reserved for the human mind. With its potential to reshape industries, society, and our daily lives, AI has become a driving force behind many cutting-edge innovations. At its core, AI encompasses the development of computer systems capable of performing tasks that typically require human intelligence. These tasks range from simple calculations to complex problem-solving, pattern recognition, language understanding, and decision-making. AI systems can process vast amounts of data, recognize patterns, and continuously improve their performance based on experience, making them powerful tools in a world that is rapidly becoming more and more driven by data-based insights.

Machine learning (ML), a subset of AI, is an essential part of AI systems. It involves training algorithms on vast datasets, allowing them to recognize patterns and make predictions without explicit programming. This capability is changing various fields, such as healthcare, finance, marketing, and manufacturing, by enabling data-driven decision-making and automation of repetitive tasks. However, the potential of AI comes with unique challenges. Ethical concerns, algorithmic bias, data quality, privacy, interpretability, and transparency are major issues that must be addressed in AI systems. AI also raises important questions about its impact on jobs and society.

1.2 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 and DevOps. 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 many languages including Spanish, French and Chinese. TMMi is also available in published book format.

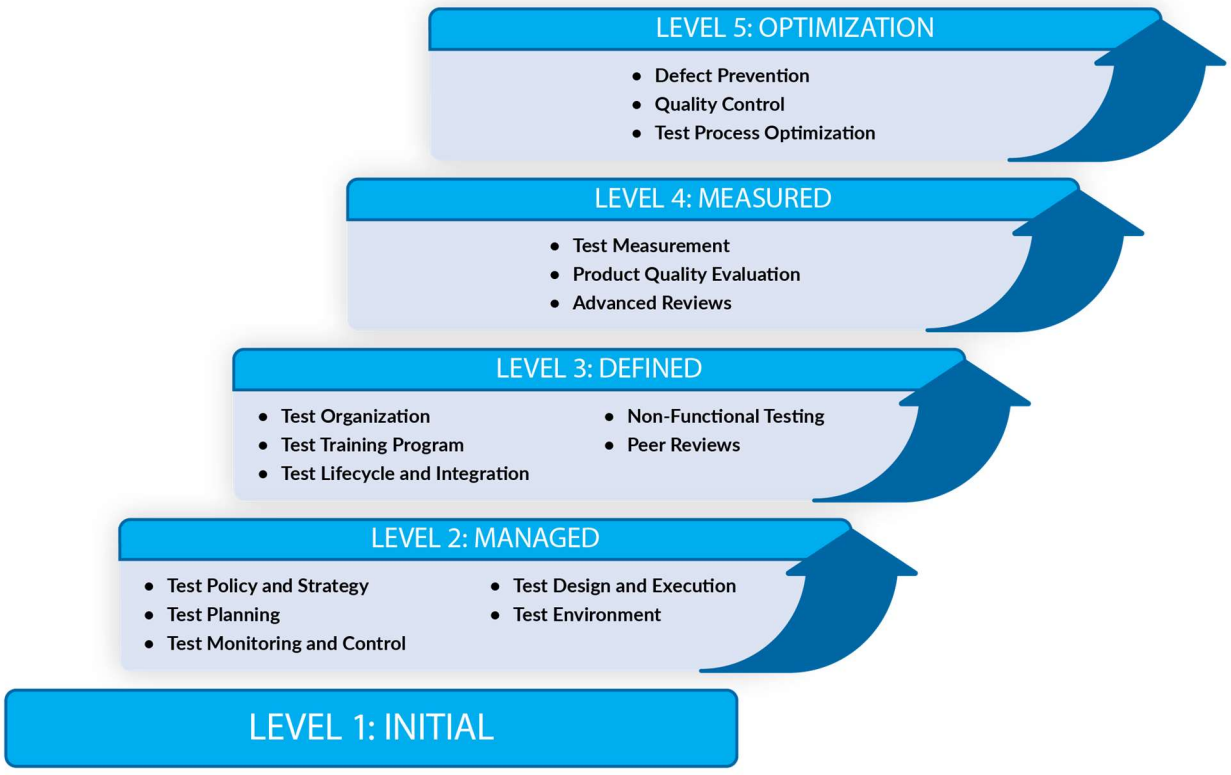

Figure 1: TMMi maturity levels and process areas

1.3 Testing AI-based systems and TMMi

1.3.1 Testing AI-based systems vs. using AI for testing

Before discussing the relationship with testing AI-systems and TMMi, and how to use TMMi in the context of testing AI systems, we first need to point out an important difference between “Testing AI systems" and "Using AI for testing". These are two different concepts related to the application of artificial intelligence (AI) in the software testing domain.

Testing AI-systems refers to the process of evaluating and validating the performance, functionality, and quality of software applications or systems that incorporate AI components. The primary goal is to ensure that

AI algorithms, models, and functionalities within a software system work as expected, produce adequate results, and meet the desired quality standards. Testing AI systems involves a variety of testing activities such as input data testing, model testing, integration testing, functional testing, performance testing, security testing, compliance testing, and testing for ethical considerations like bias and fairness. Note, that is also possible to use AI when testing AI-based systems thereby combining the two concepts.

Using AI in testing involves the application of artificial intelligence techniques and tools to enhance and optimize (activities within) the software testing process itself. The primary goal is to improve testing efficiency, accuracy, and coverage by leveraging AI technologies to support and (partly) automate various testing activities and make data-driven decisions. However, AI can also be used in testing for estimations, defect prediction, test analyses, test case generation, test data generation, test execution, test result analysis among others. AI can also help prioritize test cases based on risk and historical data.

In summary, "testing AI systems" is about ensuring the quality of AI-driven software, while "using AI in testing" involves leveraging AI to improve the efficiency and effectiveness of the testing process itself. In the context of this document the scope is on testing AI-systems only and how this relates to TMMi, and where TMMi can help. Using AI in testing may well be something to be considered as testing practices for the next version of TMMi (TMMi 2.0).

1.3.2 Challenges testing AI-systems

How do we test AI systems, especially since the most common form, machine learning, will change its behaviors in response to our tests? Also, in testing we want to be able to say that a test passed or failed. But what does “passing a test” even mean for an AI system? While manually it might be possible to say whether the test passed or failed with less complex reasoning, for large volume of data it may not be possible. There may not be a clear and full specification, the correct behavior may change over time, or we may not even know what the correct behavior is beyond what the system tells us.

Some of the challenges when dealing with testing AI systems:

- AI systems are being used to deal with complex problems where the number of possibilities is huge (even compared to existing software)
- Data has biases, and these can very easily become embedded in AI systems
- AI-based systems may experience model drift, data drift and/or concept drift
- Lack of test oracles that are able to determine expected results
- Performance metrics for the AI model are typically very different from traditional system performance metrics and new to many testers
- Traditional test techniques will often not be applicable anymore. We need to re-think our test techniques. We also need to re-think how we measure test coverage.
- The non-deterministic nature of AI systems; this characteristic of many AI-based systems makes it inherently challenging to guarantee the precise behavior of these systems.

The self-adaptive characteristic of some AI systems, as they may change their behavior during testing or as a result if the testing.Testing AI systems today is challenging. There are no easy solutions to all the challenges. Perhaps an interesting way to start to learning about testing AI-based systems is by studying the certification

syllabi as developed by AiU United [AiU] and ISTQB [ISTQB]. Generally speaking, an organization needs to have a level of test maturity before starting to face the challenges of testing an AI-based system.

1.3.3 Objectives and approach

This document explains how combining testing AI-systems with the TMMi test process improvement model is a possible route to achieve higher level of test maturity and ultimately your business objectives. It explains how TMMi can be used and applied beneficially in the context of testing AI-systems. Proven alternatives are provided to traditional testing approaches that implement TMMi practices to be applied when testing AI-systems. This document covers how TMMi can help organizations involved in testing AI-systems, irrespective of how mature the organization is on their journey to become more mature and increase their quality levels, e.g., 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 testing AI-systems and what the testing practices will typically look like. If a TMMi practice is not expected to be performed when testing an AI-system, since it has no added value, this is also explicitly stated. Note that the testing practices for testing AI-based systems are identified and mapped to the TMMi practices and goals, but not described in detail. There are good resources in the market that provide additional information on the testing practices for testing AI-systems, e.g., [Smith].

Since testing an AI-system is not “just” testing another type of system, the changes to the practices are to be found in the process areas that related to test engineering activities, e.g., test planning, test design, non-functional testing etc. In the TMMi model the test engineering process areas are especially located at TMMi levels 2 and 3, this document will therefore only discuss TMMi level 2 (Managed) and TMMi level 3 (Defined) in the context of testing AI-based systems.

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 the context of testing of AI systems 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.

2 TMMi Level Managed 2

2.1 Process Area 2.1 Test Policy and Strategy (TPS)

The purpose of the Test Policy and Strategy process area is to develop and establish a test policy, and an organization-wide or program-wide test strategy in which the test activities, e.g., test types and test levels, are 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 of course true in any organization regardless of the type of system being tested. First of all, it is important to understand AI, the risk it brings and that different test approaches will be needed. Some topics, such as ethics, bias, data and regulations, may need to be introduced or get more explicit focus. In addition, non-functional characteristics specific for AI-systems such as transparency, interpretability and explainability need to be introduced. As a result, test objectives and goals probably need to be re-considered and also the basic views regarding testing and test process definition. While most of these will be largely the same as for non-AI systems, there will be variances. For example, the objectives may include a statement about reducing risks introduced by AI systems, the process definition may need to make reference to the fact that AI systems can return different results for the same inputs, a model testing phase prior to integration. Such a phase is generally not there with traditional systems.

Ethics may need to consider how testers use data with AI-based systems but is also related to ethical behaviour of the system itself. Where an AI-system can render serious decision consequences to life and liberty in the real-world ethical decisions are at stake and important. Ethical concerns in AI will be different depending on whether the system is learning from its results or only from specified training datasets.

Data will bring issues like bias, privacy and security as being important items to consider in a test policy. Good data requirements, and an understanding of the population are becoming of utmost importance. For example, facial recognition systems need to be trained and tested with data that is representative of the affected population and not just a convenient set of data. A multinational product will mean testing with a far bigger range of data than one could get from a local population. Note, that checking for bias shouldn't stop after training the model. Bias should also be monitored when the model is actually being used to make predictions (inference time).

Quality is not normally a **regulated** aspect of IT systems, except in areas such as safety critical systems. However, due to the increasing use of AI-based systems that make decisions about people and the inability of people to fully understand these decisions, various regulations are progressing or have come into force, that legally constrain organizations about quality.

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 and test levels that are to be performed. It is not good enough to just state for example that within each project integration testing and acceptance testing will be carried out. We need to define what is meant by integration 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 are 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.

A test strategy is a vital document also when testing AI systems. Testing AI systems is relatively new to many organizations, as such a common organization-wide test strategy will guide the projects in their AI testing and prevent projects from re-inventing the wheel. A test strategy also ensures that all those involved in testing understand the bigger picture. With traditional systems the test strategy is built around the test levels and test types to performed. They are identified and defined in relation to each other. Many of this is different with testing of AI systems.

The AI testing test strategy first of all distinct two major phases: Off-line testing and On-line testing.

During the **off-line phase** the AI model is tuned, validated and tested using metrics, e.g., accuracy or precision, to evaluate the achievements of its objectives. Typically, the machine learning model is tested for functional behavior but also some non-functional tests that are appropriate for the model in isolation, such as speed of training, speed of prediction, computing resources used, adaptability, and transparency are conducted. In addition to model testing, the offline phase also includes another new level of testing: input data testing. Input data testing is to ensure that the data used by the system for training and prediction is of the highest quality.

During the **on-line phase** the integration of the trained model with the rest of the system, including other AI and non-AI components and its operational environment are tested. Both functional and non-functional tests, e.g., performance, are performed during this phase. During the online phase the test levels and test types applicable with traditional testing can be used to structure the test strategy.

- Component testing is applicable to any non-AI model components, such as user interfaces and communication components.
- Component integration testing is conducted to ensure that the system components (both AI and non-AI) interact correctly.
- System testing is conducted to ensure that the complete system of integrated components (both AI and non-AI) performs as expected, from both functional and non-functional viewpoints, in a test environment that is closely representative of the operational environment. Depending on the system, this testing may take the form of field trials in the expected operational environment.

- Acceptance testing is used to determine whether the complete system is acceptable to the customer. However, there are aspects that may need additional focus. It could be that the system is designed to replace human activity, in which case it might be necessary to compare the behavior of the system to that of humans and against defined performance criteria.
- A test level also applicable with testing of AI systems, is monitoring the system's performance on an ongoing basis. With AI systems there is often a need to continually review and revisit the quality of the system in live usage. An important reason for monitoring the quality in production is bias detection. It can be difficult to determine bias sometimes in a test environment due to a potential difference between the training data and the data a model is exposed to in the live environment, and continuous monitoring can help with this. Another important reason for continuous monitoring is drift. With AI-based systems model drift, data drift and concept drift are all factors to consider and monitor over time. Continuously monitoring model performance metrics is required to detect signs of drift early. Drift refers to the fact that working models may degrade over time because of the change in the relationship between input features and output. While the model is being used, its situation may evolve and the model may drift away from its intended performance.

Note, that it may also be the case that the AI-systems are not developed in the organization itself but they implement AI-systems (as part of a larger configuration) developed externally, e.g., by Microsoft, Apple or Google. This will of course also influence their AI test strategy, which will in turn resemble a COTS test strategy where the focus is on integration with other systems, tailoring and configuring the systems to the organizational needs.

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.

If testing of AI systems is part of the test process improvement program and takes place on a regular basis, specific test performance indicators will be needed for testing of AI systems to show the results of the test improvement program. Because of the differences of nature of an AI system and accompanying test objectives, traditional performance indicators, e.g., Defect Detection Percentage, typically do not apply. The test performance indicators for AI systems will measure how accurately and reliably the trained or tested model delivers its results. Note that the performance indicators to be used with AI

		Predicted	
		Negative **(N)** -	Positive **(P)** +
Actual	Negative -	True Negative **(TN)**	**False Positive (FP) Type I Error**
	Positive	**False Negative (FN)**	

Figure 1: Confusion Matrix

systems will depend on the learning type (unsupervised or supervised) and model type (clustering, association, classification or regression). For example, for supervised classification AI systems a confusion matrix (see figure 1) could be used and based on this accuracy and/or precision can be calculated and used as a performance indicator. With supervised regression AI systems, the Mean Square Error (MSE) could be used as performance indicator [ISTQB]. MSE is the average of the squared differences between the actual value and the predicted value. The value of MSE is always positive, and a value closer to zero suggests a better regression model. More examples of metrics possible to be used as (test) performance indicators can be found in the AiU Certified Tester in AI syllabus [AiU] and the ISTQB Certified Tester AI Testing syllabus [ISTQB].

2.2 Process Area 2.2 Test Planning (TPL)

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").

2.2.1 SG1 Perform Risk Assessment

Exhaustive testing is impossible. This is also true with testing AI-based systems, and choices need always to be made and priorities always need to be set. This TMMi goal is therefore also applicable when testing AI-based systems. The test approach will be based on a risk analysis for the AI-based system and will include both traditional testing as well as more specialized testing to address those features specific to AI components and AI-based systems.

Typical techniques for assessing product risks in AI-based systems include brainstorming, expert interviews, and checklists.

- Brainstorming is driven via one or more meetings where communication is the key. Often no one in a project will have the complete picture of how the AI-based system works. Multi-disciplinary teams with a wide range of specialist skills especially related to AI, are essential to the quality of the process, where not one aspect of the project is in the majority. The objective of the team is to provide a wide range of perspectives and perceptions on all aspects of the AI-based system and its operation.
- Expert interviews provide valuable insights and expertise, enhancing the risk assessment and identification process. By utilizing interview techniques, organizations can gather information directly from AI experts, enabling a more comprehensive understanding of potential risks and their impact.
- Checklists for risk identification pool the experiences of earlier AI projects. Ideally the list should focus on generic risk areas of AI-based systems. The objective is prompt thought in the areas of sometimes forgotten, frequently occurring product risks that could be considered and indications of the type of problems the AI-based systems could be subjected to.

Examples of product risks for AI systems:

- Input data quality too low
- Insufficient data volumes for training
- Problems with operational data pipeline
- The ML workflow used to develop the model may not be completely correct

- An incorrect choice of ML framework, algorithm, model, model settings and/or hyperparameters
- The desired ML functional performance not achieved in production environment
- Users are dissatisfied with delivered ML functional performance
- The self-learning system is not providing the service expected by users
- Users are frustrated by not understanding how the system determines its decisions
- Working models degrade over time because of the change in the relationship between input features and output (often referred to as concept drift)
- Pre-trained models have their own unknown and/or undocumented biases and shortcomings
- Hallucinations, which can come from poor training data, the model's tendency to fill in gaps with plausible-sounding text, or misinterpreting prompts, and could be a significant challenge because the AI often presents its fabricated output with high confidence.

2.2.2 SG2 Establish a Test Approach

A test approach is defined to mitigate the identified and prioritized product risks. The test approach is also guided by the defined organizational test strategy (see paragraph 2.1.2). It should be made clear to those involved what to do and how the testing will be performed. As with the test strategy, the detailed test approach distinguished two major phases: offline and online testing of the application. A test approach will be defined for each test level and test type to be performed.

The offline phase starts with data preparation, preprocessing and data input testing (the initial test level). Activities to be performed are data manipulation, data filtering and data preprocessing. Data preprocessing includes data imputation, i.e. dealing with missing values, data visualization to get a big picture and treating anomalies and outliers, correlation analysis and reducing dimensions. Thereafter the AI model is evaluated and analyzed. This also involves splitting the available input data set into a training, validation and testing dataset. The training set contains the data that the model is trained on. The validation dataset is used by the ML algorithm to evaluate if the training was effective. In the post-training phase, the ML model is evaluated with the testing dataset.

During the online phase the non-AI components are tested and the interaction between the AI and non-AI parts are tested. This typically requires a detailed study and understanding of the architecture of the AI application. Finally, the fully integrated AI system is tested. In addition to exploratory testing, a popular technique to identify and define test scenarios for AI-based systems is metamorphic testing.

Entry criteria, also called definition-of-ready (specific practice 2.3) and exit criteria, also called definition-of-done (specific practice 2.4) are also part of the defined test approach. The entry criteria for a testing phase are typically the fulfillment of the exit criteria of the previous testing phase, but also relates to test preparation being completed and the availability of the test environment. Depending on the testing phase, some of the exit criteria may resemble those with traditional systems, e.g., test coverage and outstanding defects. There are also metrics available that can be used as a basis for defining exit criteria that are very different from the exit criteria used with traditional systems.

The table below provides example of exit criteria (metrics) that can be used per type of AI model [AiU]:

Learning type	Model type	Exit criteria used
Unsupervised	Clustering	Inertia Adjusted Rand Score
	Association	Support Confidence Lift
Supervised	Classification	Accuracy Precision Recall/Sensitivity Specificity F1-score
	Regression	Root-mean-square-error R-square error

Table 1: Exit criteria and metrics per type of AI model

Whether specific practice 2.5 Define suspension and resumption criteria is relevant with testing AI-based systems depends on the lifecycle being applied and approach being taken. When a more traditional approach is taken it will be applicable. With Agile it is likely to be less relevant. With Agile it is typically used implicitly at status and progress meeting, but defining them up front explicitly per testing phases is overdone. When there are blocking issues that could be considered as potential or actual threats to the progress, these are discussed in meetings. In the meeting the team will decide what actions, if any, need to be taken to solve the issues. The Agile project management routine thus serves as an alternative practice for this specific practice.

2.2.3 SG3 Establish Test Estimates

Test effort estimation involves predicting the amount of test-related work needed to meet the test objectives of a project or team. It is important to make it clear to the stakeholders that the estimate is based on a number of assumptions and is always subject to estimation error. This is especially true when an organization has less experience with testing a specific type of system, e.g., an AI-based system.

Generally speaking, there are two estimation approaches available:

Estimation based on ratios. In this metrics-based technique, figures are collected from previous projects within the organization, which makes it possible to derive “standard” ratios for similar projects. The ratios of an organization’s own projects (e.g., taken from historical data) are generally the best source to use in the estimation process. These standard ratios can then be used to estimate the test effort for the new project. Since testing AI-testing systems is relatively new to the industry and to most organizations, few (if any) metrics are available making this estimation approach one that today is not (yet) preferred with testing AI-based systems.

Expert-based estimation, e.g., Wideband Delphi. In this technique, “experts” make experience-based estimations. Based on a work breakdown each expert, in isolation, estimates the effort for individual tasks. The results are collected and if there are deviations that are out of range of the agreed upon boundaries, the experts discuss their current estimates. Planning Poker is a variant of Wideband Delphi, commonly used in Agile software development. With metrics not being available, this technique seems to be the one preferred for estimating the effort and cost associated with testing AI-based systems. Those involved in the project or team, and perhaps those who have been involved in other AI projects from within the organization can be part

of the estimation team. Following the process, they will make their best guess, also documenting any assumptions being made, for estimating test effort and cost. Wide-based Delphi estimation session are typically provided with input such as a list of the test work products to be developed based on the defined test approach, and a list of test tasks to be performed related to the test work products.

Estimating efforts for testing AI systems is typically more complex than traditional software testing. Hereafter are some examples of AI-specific factors that can be considered:

- Data Complexity, testing for AI systems/applications requires extra effort for data validation, bias detection, and drift monitoring to ensure accurate and fair predictions.
- AI Model Behavior, unlike traditional software, AI learns and evolves, so it needs continuous testing, explainability checks, and tuning.
- Continuous Monitoring, AI needs real-time tracking, performance checks, and model retraining to maintain its performance.
- Regulatory Compliance, AI testing must follow GDPR, AI Act, and industry-specific rules to ensure ethical and legal use.

2.2.4 SG4 Develop a Test Plan

A test plan should always be developed with two main parties in mind: (1) the testers (they need to know what to do) and (2) the stakeholders (they need to understand the testing that will take place, their involvement, agree to resources, associated risks, etc.). The level of detail of the test plan with testing an AI-based system also depends on the lifecycle being applied, e.g., a more Agile way of working will imply a more lightweight test plan.

A test plan for AI systems must extend beyond traditional software testing due to AI's data-driven, adaptive, and probabilistic nature. Unlike conventional applications, AI models learn, evolve, and make probabilistic decisions, demanding specialized testing strategies and focused test plans. Given AI's complexity and evolving nature, testing must be continuous and automated, with pipelines ensuring on-going validation.

Since testing an AI-based system is relatively new to many organizations, a documented test plan will be needed to ensure that testers understand what they need to do and stakeholders are well-informed. The test plan may take the format of a document, a presentation or a mind-map, as long as it serves its purpose. The test plan will document the results of the previous test planning activities, a prioritized list of product and project risks, detailing the approach and activities for both the offline phase (data input testing, model testing) and the test levels and test types of the online phase.

A (test) schedule will be developed. As testing is performed as an integral part of development, there will typically not be separate test schedule but rather an overall schedule that also includes the testing activities. The schedule may take the format of detailed documented schedule, e.g., like with sequential lifecycles, but it can also take the format of a task board with priorities, e.g., like with Agile lifecycles.

As part of test planning, a plan is created for the availability of the necessary test staff resources who have the required knowledge and skills to perform the offline and/or online testing. This of course maybe challenging since testers with the required knowledge and skills to perform this kind of testing is typically a scarce resource. It may there be necessary to define training needed as part of the test plan to ensure the test staff is capable of getting the testing job done.

2.2.5 SG5 Obtain Commitment to the Test Plan

Finally, the test plan and defined test approach needs to be agreed upon by the stakeholders. Stakeholders must understand and agree upon the prioritized list of product risks and the test mitigation actions to be performed. Their understanding and commitment can for example be achieved by means of a short presentation followed by a discussion explaining the product risks, test approach and its rationale, and budget required. The discussion on the test plan and test approach should take place in the context of the purpose of the ML model / AI-system to be developed and ensure alignment with business priorities. Acceptance criteria are of course another aspect that is important to discuss and agree upon with the stakeholders.

If a more Agile approach is used to develop and establish the test plan and test approach then this has already been a team-based exercise, possible led by an AI 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 as it is a team-effort. Of course, other stakeholders, those outside the team, may also need to provide commitment to the test plan, for them the presentation and discussion session, as addressed earlier, will continue to be important and need to be conducted.

2.3 Process Area 2.3 Test Monitoring and Control (TMC)

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 or product quality deviates significantly from expectations.

2.3.1 SG1 Monitor Test Progress against Plan

Monitoring test progress, conducting progress reviews and reporting on test progress is not different for testing an AI-based system compared to testing a traditional (non-AI) system. The way monitoring test progress is implemented and performed depends largely on the lifecycle being applied. With sequential V-model type of lifecycles test progress monitoring is typically more formal and documentation intense. With Agile type of lifecycles test progress monitoring is typically much less formal and supported by task board and low on documentation, see also [TMMiAgile].

2.3.2 SG2 Monitor Product Quality against Plan and Expectations

When monitoring product quality, it's important to understand that testing an AI-based system is different from testing a traditional system. This difference also affects which aspects of product quality need to be monitored. With traditional systems "test" refers to various processes such as searching for defects, executing tests and measuring performance. Its meaning in AI refers specifically to statistical evaluation of the system performance against a dedicated dataset. Basically, the type of exit criteria that are being monitored and reported upon are different by nature (refer to paragraph 2.2.2 for some examples). In summary monitoring product quality is especially performed for identified product risks and defined exit criteria. Monitoring defects may well be performed for the non-AI parts of the systems. Refer to the previous paragraph for the ways monitoring product quality can be implemented.

Another difference with monitoring product quality of AI systems, is that monitoring the system's performance on an ongoing basis. With AI systems there is often a need to continually review and revisit the quality of the system in live usage. As stated before in the document with AI-based systems model drift, data drift and

concept drift are factors to consider and monitor over time. Continuously monitoring model performance metrics, e.g., accuracy, is required to detect signs of drift early. Other factors factor that are important reasons to keep monitoring product quality after release include preventing adversarial attacks, data poisoning and self-learning evolution behavior.

2.3.3 SG3 Manage Corrective Actions to Closure

As with monitoring test progress (see above) also managing corrective actions to closure is not different for testing an AI-based system compared to testing a traditional (non-AI) system. However, the type of issues may well be different. Typical issues to expect when testing AI systems are in the area of input data quality. Data quality issues may have a significant impact on the project and/or take long to fix.

The way managing corrective actions to closure is implemented and performed depends largely on the lifecycle being applied. With sequential V-model type of lifecycles the process for issues related to progress are typically managed by a test manager or project manager. With Agile type of lifecycles, a Scrum master is often responsible for handling progress impediments. The status on this type of corrective actions will be shown on a task board and discussed at the daily standup meeting.

2.4 Process Area 2.4 Test Design and Execution (TDE)

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 purpose statement is at a high level still applicable, the process of defining test conditions, test cases and executing them is very different as explained hereafter. Once a model has been generated, (i.e., it has been trained, evaluated and tuned), it should be tested against an independent test dataset set to ensure that the agreed functional performance criteria are met (see section 2.2.2).

2.4.1 SG1 Perform Test Analysis and Design using Test Design Techniques

When testing an AI system, the concept of “test conditions” and “test design” differs fundamentally from traditional software testing. Instead of deriving explicit input–output pairs, the focus is on constructing and validating datasets that can meaningfully evaluate the behavior, accuracy, and robustness of the trained model.

The primary goal becomes the definition and design of datasets that represent the range of conditions under which the AI model will operate. This includes identifying the types, sources, and quality of data to be used for training, validation, and testing. Typically, three distinct yet statistically representative datasets are established from a common data source:

- Training dataset – used to train the AI/ML model.
- Validation dataset – used to evaluate and tune model parameters.
- Test dataset (holdout dataset) – used to assess the tuned model’s generalization performance.

In the TMMi context, the test analysis and design phase involves identifying and prioritizing the test data conditions and evaluation metrics derived from the AI system’s purpose, requirements, and risk assessment.

Here, the test basis includes not only specifications and design documents, but also data collection strategies, labeling guidelines, and model performance criteria.

Because suitable, unbiased data is often limited, the training and validation datasets are frequently derived from a single combined dataset, while the test dataset must remain isolated. This separation ensures objective model evaluation and prevents data leakage. There is no universal ratio for splitting datasets, but a typical guideline is 60:20:20 for training, validation, and testing. Data splitting should be random, unless specific stratification or representativeness constraints exist (e.g., small datasets or critical subpopulations).

Test design in AI focuses on the selection and prioritization of test cases and conditions that reflect realistic and edge-case scenarios. This may involve:

- Designing adversarial or stress test data to probe model weaknesses.
- Using equivalence partitioning and boundary testing in the data space (e.g., edge values of feature distributions).
- Using combinatorial testing techniques, such as pairwise testing, since the number of parameters of interest for an AI-based system can be extremely high, especially when the system uses big data or interacts with the outside world.
- Metamorphic testing, a technique aimed at generating test cases which are based on a source test case that has passed, is another techniques that can be used with testing AI systems.
- Applying bias, fairness, and explainability tests as part of the test condition set.
- Establishing traceability from data sources and model requirements through to test datasets and performance outcomes.

The outputs of this goal include a dataset specification, documentation of test data coverage, and defined

acceptance criteria (e.g., accuracy thresholds, confidence intervals, bias limits). These artifacts collectively ensure that testing of the AI model follows a structured approach.

2.4.2 SG2 Perform Test Implementation

Test implementation for AI systems involves acquiring, preparing, and finalizing the test data and environments as defined during test design. AI testing focuses on obtaining and curating datasets that will be used to verify the trained model's performance, robustness, and compliance with requirements. The acquired data may take diverse forms (e.g., numerical, categorical, image, tabular, text, time series, sensor, geospatial, video, or audio) and typically requires extensive pre-processing before it can be used for actual testing. The main pre-processing activities include cleaning, transformation, augmentation, and sampling.

- Cleaning: Incorrect, duplicate, or inconsistent data are identified and either corrected or removed. Missing data may be addressed using imputation techniques, while anonymization or pseudonymization is applied to protect personal or sensitive information. Outliers may be removed or retained depending on their relevance to the AI system's operational context.
- Transformation: Data formats or structures are converted to a consistent form suitable for the model under test (e.g., converting categorical data to numerical form, normalizing numerical features, or re-encoding image formats). These transformations ensure comparability and stability across test executions.
- Augmentation: To improve coverage and robustness, the dataset can be artificially expanded by generating new or modified samples (e.g., rotated images, paraphrased text, or noise-injected data).

Augmentation may also include adversarial examples designed to test the system's resistance to perturbations or attacks.

- Sampling: Representative subsets are selected from large datasets to ensure feasible testing while maintaining statistical validity. Sampling methods (random, stratified, or risk-based) should preserve diversity and the critical characteristics identified in test design.

All pre-processing implementation steps must be documented and be traceable to the dataset specifications. The implemented test data, test scripts, and environment configurations collectively form the test work products required for test execution and model evaluation.

2.4.3 SG3 Perform Test Execution

Test execution for AI systems involves running the prepared test datasets and scripts to collect evidence on how the trained model performs against its defined acceptance criteria. The goal is to ensure that the model behaves as intended, achieves expected levels of performance, and operates reliably across the range of data conditions identified in earlier phases. Testing is only started once the training, tuning and validation of the AI model are done and adhere to the defined criteria. This can be considered an entry criteria for starting test execution.

In testing AI systems, executing a test generally means applying the test dataset to the trained and validated model and recording its outputs (predictions, classifications, scores, or actions). These outputs are compared against the expected or reference results (ground truth) and evaluated using confusion matrices and quantitative performance metrics such as accuracy, precision, recall orF1 score (see paragraph 2.2.2).

Due to the inherent stochastic nature of AI systems, individual test executions may produce slightly different outcomes. Therefore, tests should be repeated under controlled conditions, and results should be analyzed statistically (e.g., mean and variance of performance metrics). This ensures that conclusions are based on reproducible and stable evidence.

All test outcomes, anomalies, and deviations from expected performance must be documented and traceable to the corresponding test data and conditions defined. Test incidents are analyzed to determine whether they originate from data quality issues, model limitations, or implementation defects. The outputs of this specific goal include executed test logs, captured results, performance reports and incident reports.

2.4.4 SG4 Manage Test Incidents to Closure

The process and practice for managing test incidents to closure is no different for incidents and defects related to AI-systems. Of course on a more detailed technical level there are differences in the type of defects being found and the way tasks such as defect analysis are being performed, etc. For example a detailed data quality analysis may reveal defects related to reduced accuracy, biased model and compromised model.

The incident management process also largely depends on the development lifecycle being applied. There are substantial differences in how incidents are documented and managed to closure in a sequential traditional project compared to a project in an Agile context [TMMiAgile].

2.5 Process Area 2.5 Test Environment (TEV)

The purpose of Test Environment is to establish and maintain an adequate environment 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. AI-based systems can be used in a wide variety of operational environments, which means that the test environments are similarly diverse. Examples of characteristics of AI-based systems that typically cause the test environments to differ from those for conventional systems include self-learning, autonomy, multi-agency and big data. Also, virtual test environments are commonly used when testing an AI-based system. This typically makes it easier to test dangerous, unusual, extreme and time-intensive scenarios. Virtual test environments also provide far greater controllability of the test environment. The simulation of hardware by virtual test environments allows systems to be tested with (simulated) hardware components that may otherwise not be available.

Note that data preparation which is typically a big part of AI/ML-system workflow is part of the test design and execution process area (see paragraph 2.4). The data in principle encompasses the (test) cases that will used to train, validate and test the tuned model. As such (test) data plays a very role for testing AI-based systems compared to traditional systems.

2.5.1 SG1 Develop Test Environment Requirements

Specification of test environment requirements for an AI-based system should be performed early in the project. It requires a detailed understanding, by those defining the test environment requirements, of the AI system to be developed, since its characteristics (see above for some examples) will determine what is required. 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. Especially with AI-based systems this is very important since the test environment is often not very straight forward and can be quite complex.

2.5.2 SG2 Perform Test Environment Implementation

With this TMMi specific goal the test environment requirements are implemented and the test environment is made available to be used for testing the AI model. Unit tests are performed on test environment components as appropriate and a confidence test is performed at the end of the implementation phase to determine whether the test environment is ready to be used for testing the AI model. Generic test data will be used during testing the environment.

2.5.3 SG3 Manage and Control Test Environments

Management and control of the test environment will take place throughout an AI project. Often the project team needs to set up and maintain the test environment themselves, but this may also be done by a separate unit outside the team. When the activities are done by the 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 project, e.g., they need to be addressed in a planning session, put onto the task board and to 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 applicable and do not change in essence. The things may be different is the type of test environment. test data and its components that are needed.

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 SG1 Establish a Test Organization

Test Organization is an often-misunderstood process area. Many read this as TMMi requires an independent test group or even department that does independent testing. However, the most common testing model for AI-systems is Agile, continuous testing embedded in the AI development team, supported by automation and monitoring. With Agile and thus also with testing AI-systems, the test organization takes the format of a test competence centre that focuses on improving the organizational test processes and profession, including the skills and career paths. An organization that already has a test competence center in place, thereby, in principle, the center should also cover testing of AI-systems. The main change will be in the required competences (see SG2 hereafter) and responsibilities.

Examples of new, AI-model and AI-based system testing, specific responsibilities for the test organization include:

- define an AI test strategy
- define AI testing processes and standards
- provide AI testing best practices
- support teams in evaluating AI-model quality and data bias
- define monitoring and post-deployment validation practices.

3.1.2 SG2 Establish Test Functions for Test Specialists

Testing is regarded as a valued profession and discipline. Testing an AI system requires specific knowledge and skills, which means the tester's job description must be updated. This update is not only about adding responsibility for testing AI systems, but more importantly about expanding the required knowledge and competencies. While testers will still need traditional testing knowledge and skills, they must also develop additional capabilities specific to AI systems. These include a basic understanding of AI and machine learning concepts such as supervised, unsupervised, and reinforcement learning; familiarity with training, validation, and test datasets; and an understanding of issues like overfitting, underfitting, and model drift. Testers also need strong awareness of data quality and data testing practices, including data profiling and exploration, checking for completeness, bias, noise, outliers, and duplicates, and understanding data pipelines. Because AI results are rarely binary, testers must apply statistical and probabilistic thinking, with knowledge of concepts such as precision, recall, F1-score, confidence intervals, false positives versus false negatives, and sampling techniques. In addition, testing AI-systems requires knowledge and skills related to new quality aspects such

as bias, fairness, ethics, explainability, and interpretability. Finally, testers must be prepared for ongoing monitoring and post-release testing, as AI systems can change over time even without new software releases.

Testing an AI-system is a team effort. The tester will work closely with non-testers, e.g., data engineers and scientists, ML engineers and domain experts amongst others. It is important to elaborate other non-test specific job descriptions with expectations regarding their test activities, roles, responsibilities and required testing knowledge and skills.

3.1.3 SG3 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. The career path as already in place within the organization will largely remain unchanged. However, in AI heavy organizations, where testing of AI-systems becomes a vital part of part of testing activities, often the AI testing specialist becomes a career path opportunity for those who stay on the testing path and focus on AI system. The AI testing specialist is expected to have in-depth knowledge and skills on AI (testing) aspects and typically has ownership of the AI test strategy.

3.1.4 SG4 Determine, Plan and Implement Test Process Improvements

The *goal* of test process improvement is the same (reduce risk, increase confidence) as with traditional systems, but AI systems change what “quality,” “coverage,” and even “bugs” mean, so the way improvements are identified has to evolve. With traditional software, test improvements usually come from escaped defects, test coverage gaps, test execution metrics and root-cause analysis. Although these still apply with AI systems, this list is incomplete. AI systems introduce new failure modes that do not map to traditional testing. With AI-testing test process improvements also come from model performance degradation, data-related failures, bias, fairness and ethical issues, and non-repeatable test results.

Since the trigger for a test process improvement is often different, this is also true for the improvement action to be taken. Some examples of typical test process improvements with testing of AI-systems are provided in table 2.

Trigger	Typical test improvement action
Model performance degradation	Add data drift detection Improve monitoring-based testing Introduce retraining validation gates
Data-related failures	Expand data coverage analysis Add synthetic data generation Improve data validation tests
Bias, fairness, and ethical issues	Add fairness testing Segment evaluation by protected attributes Formalize ethical risk reviews

Non-repeatable test results	Control randomness (seeds) Shift focus to statistical confidence Use distribution-based assertions

Table 2: Examples test improvements with AI-systems testing

Once the test process improvements that are needed are determined, they can be planned and subsequently improved. For these parts of the specific goal, the standard TMMi model specific practices (SP4.3 Plan test process improvements and SP4.4 Implement test process improvements) are applicable.

3.1.5 SG5 Deploy the Organizational Test Process and Incorporate Lessons Learned

Also, for projects or teams involved in testing AI-systems, it is important to have access and re-use the organizational standard test process for testing AI-systems and the related test process assets as much as needed and have added value. The test organization will ensure the test process and test process assets are deployed across all teams involved in testing AI-systems. Retrospective meetings can be used to identify valuable lessons learned and to communicate test improvements identified by the team to the test organization. The lessons learned and test improvement can subsequently be submitted as test process improvement proposals and managed accordingly.

Generally speaking this specific goal and it specific practices are all, without changes also applicable when testing AI-systems. The only difference being that the processes should not just be deployed to the testers but also to data scientists and ML engineers. They will also be involved in testing, since data effectively drives the behavior of AI systems. As a result, they play a key role in validating training data quality, bias, leakage, and representativeness, as well as in testing early models using metrics such as accuracy, precision and recall, robustness, and drift.

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. A quality training program ensures that those involved in testing continue to improve their testing knowledge and skills, acquire the necessary domain knowledge and other knowledge and skills related to testing. Since quality and testing are typically a team responsibility, it is expected that AI test related training is not provided to test professionals only, but rather to the whole team.

3.2.1 SG1 Establish an Organizational 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 testing AI-systems (e.g., through the ISTQB certification scheme [ISTQB]). Examples include knowledge of AI and machine learning concepts, familiarity with training, validation, and test datasets and an understanding of issues like overfitting, underfitting, and

model drift. Testers also need knowledge and skills of data quality and data testing practices, and understanding data pipelines. As stated, the identification of test training needs is not limited to just the testers, it is of utmost importance to also specify the AI test related training needs for other team members.

The test training needs will be translated into a (lightweight) training plan. The training plan needs to be reviewed and commitments need to be established. Note that 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 the AI test training needs should be determined per knowledge and skills area.

3.2.2 SG2 Perform Necessary Test Training

Based on the organizational training plan, test professionals and other team members are identified that need to receive AI test training necessary to be able to perform their test role effectively. The training is subsequently scheduled, including required resources, and conducted. As stated above informal training vehicles can also be used.

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

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 SG1 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. With testing of AI-systems being a very different process to testing of traditional systems, a dedicated standard test process for testing AI-systems with the organization needs to be developed. Having a standard test process for testing AI-systems within the organization also prevents for each project to "re-invent the wheel".

Since testing AI-systems may vary across projects based on the type of AI-systems being tested, it is highly recommended to split the "what needs to be done" from the "how to do it" information. Test processes should not contain "how to" information or tool information unless one has decided to mandate this across all projects regardless of the type of AI-system being tested. Separate guidelines, e.g., work instructions, will contain the "how-to" information. Note that with "how-to" information, it can also be decided to just reference an existing book or paper. Especially in the context of the rapidly evolving AI domain this may be very beneficial preventing re-developing new guidelines every so often.

While the choice of process assets is up to each organization, in line with the previous paragraph, it is recommended to have just two levels of process assets. 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 project or team to decide how exactly to work within the defined framework and usually includes supporting templates. In most organizations testing AI-systems, step-by-step procedures are replaced by tool guides and training/mentoring.

The organization’s set of standard AI systems 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 results in a disproportionate amount of effort for relatively simple AI projects, making the approach less practical. When you tailor down, how far can one go? Is there a minimum defined? However, if you start with a minimum set of activities that every AI project must perform, you eliminate the risk of tailoring out a “must do”. A tailoring up approach fit much better with the flexibility that is needed in AI projects, and completely TMMi compliant. 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 AI test process a dedicated test process asset library will be created and maintained, often on a wiki, in which testers can find templates, best practices and checklists, that will assist them in their AI testing activities. Also, a dedicated test process database in which AI 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.

3.3.2 SG2 Integrate the Test Lifecycle Models with the Development Models

The AI test lifecycle model defines the main activities and deliverables for the various test levels and test types. The AI test lifecycle model is aligned with the AI development lifecycle model 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 AI projects is ensured. In AI, development and testing should be integrated within the lifecycle, and testing should already be taking place as early as possible. As testers are part of the team, they participate in up-front activities such as planning, risk analysis, defining acceptance criteria and data preparation and testing of the data.

It is important that a common understanding is created by defining how testing can be involvement and performed early. If done appropriately, AI development and testing should be fully integrated and there is a common understanding on how this is done. This specific goal is a very important and should ensure aligned of AI development and testing activities.

3.3.3 SG3 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 and test types. 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.

When testing a more complex, regulated and risk-related AI system, establishing a master test plan will be beneficial. The master test plan will address an overall test strategy across multiple models, data pipelines, environments and compliance and risk areas with possibly subordinate test plans underneath. It is advantageous for testing AI systems because it provides centralized governance for evolving models, data-driven behavior changes, cross-cutting risks (bias, drift, compliance), and multiple test efforts which is something a traditional level test plan cannot manage effectively.

From a governance perspective, a master test plan for testing an AI system typically aims at coherently addressing data (pipeline) testing, model validation, regression testing, testing of bias, integration testing, system testing, monitoring and model drift. Consider a master test plan as a test governance and coordination tool, not just to plan test execution. Hence enough reasons why establishing a master test plan (and thus this TMMi specific goal) is also applicable when testing AI-systems.

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.

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

3.4.1 Quality Characteristics for AI-based system

Product quality is centered on satisfying stakeholders' needs. These needs have to be translated into functional ("what" the product does) and non-functional ("how" the product does it) requirements. The International Organization for Standardization (ISO) has defined a set of non-functional characteristics that are used to define the quality of a software product or system [ISO 25010]. However, extensions to the traditional software quality models and quality characteristics are needed to address unique aspects of AI-based systems.

ISO has also published a specific standard addressing quality characteristics for AI-based [ISO 25059]. In this standard, which is based on ISO/IEC 25010, a number of modified and new quality characteristics specific for AI systems have been identified and defined. Some of the sub-characteristics have different meaning or contexts as compared to the ISO/IEC 25010 model. The following modified and new quality sub characteristics are part of ISO/IEC25059:

- **Functional correctness** (modified): Perfect correctness is usually unattainable for probabilistic AI systems. Instead, quality is demonstrated by quantifying the system's accuracy, error characteristics, and precision using appropriate evaluation metrics, and by ensuring these meet the requirements of the intended use. (Functional correctness is part of Functional suitability)
- **Functional adaptability** (new): Functional adaptability of an AI system is its ability to adapt to changing dynamic environments in which it is deployed. This characteristic requires testing approaches that can evaluate system behavior across multiple environmental conditions and over extended time periods. AI

systems can learn from new training data, production data and the results of previous actions taken by the system. (Functional adaptability is part of Functional suitability)

- **Environmental sustainability** (new): This sub characteristic requires an organization to consider three main areas with AI system, each with their own sustainability aspects: economic sustainability, social sustainability (e.g., trustworthiness) and environmental sustainability. (Environmental sustainability is part of Performance efficiency)
- **User controllability** (new): User controllability is a property of an AI system such that a controller can intervene in its functioning in a timely manner. Engagement of control is important if unexpected behavior cannot be completely avoided and intervention is required to handle negative consequences. (User controllability is part of Interaction capability)
- **Transparency** (new): Transparency relates to the degree to which appropriate information about the AI system is communicated to stakeholders. Transparency of AI systems can help potential user of AI systems to choose a system to fit their requirements, improving stakeholders' knowledge about the applicability and the limitations of an AI system, and assisting with the explainability of AI systems. (Transparency is part of Interaction capability)
- **Robustness** (new): This is used to describe the ability of a system to maintain its level of functional correctness under any circumstances including the presence of unseen, biased or invalid data input, external interference, environmental conditions encompassing generalization, resilience or reliability and attributes related to the proper operation of the system as intended by the developers. (Robustness is part of Reliability)
- **Intervenability** (new): The extent of intervenability can be determined depending on the scenarios where the AI system can used. The key to intervenability is to enable state observation and transition from an unsafe state to a safe state. Compared to operability, intervenability is more fundamental from a quality perspective and is intended to prevent an AI system from doing harm or hazard. (Intervenability is part of Security).

In addition to the above, which are all product quality characteristics, ISO 25059 has also identified a new quality in use characteristic: **Societal and ethical risk mitigation** (a new sub-characteristic of 'Freedom from risk'). It considers many areas to mitigate societal and ethical risk, including accountability, fairness and non-discrimination, professional responsibility, promotion of human values, privacy, safety and security, human control of technology, community involvement and development, human centered design, respect for the rule of law, respect for international norms of behavior, environmental sustainability, and labor practices.

3.4.2 SG1 Perform a Non-Functional Product Risk Assessment

The product risk techniques, e.g., brainstorming, expert interviews and checklist, 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 the start of the project, the product risk assessment, now also for non-functional testing, can be performed based on the defined product vision, model objectives, system requirements and acceptance criteria. A product risk assessment for AI is best treated as a living artifact that evolves throughout development and operation, informing both pre-release non-functional testing and ongoing production monitoring.

A product risk assessment for non-functional testing of AI systems extends traditional software risk assessment by accounting for the unique properties of AI models: probabilistic behavior, model drift, data dependence, and uncertainty. The goal is to identify which non-functional characteristics (e.g., per ISO 25059) present the greatest business or operational risk so testing effort can be prioritized. For some non-functional areas, specialists may be needed to assist.

Examples of non-functional product risks for AI systems to be considered during a product risk assessment:

- Performance efficiency: Can the AI system consistently meet response time and throughput requirements under expected and peak loads?
- Robustness: Does the AI behave predictably when given unexpected, incomplete, malicious, or out-of-distribution inputs?
- Security: Is the AI protected against prompt injection, adversarial attacks, unauthorized access, model abuse, and data exfiltration?
- Compatibility: Can the AI system interoperate correctly with other systems, APIs, models, and services without introducing failures?
- Interaction Capability: Can users understand, control, and appropriately interpret AI outputs, confidence levels, and explanations?
- Safety: Can the AI system prevent or mitigate unacceptable harm to people, organizations, or the environment when failures or unexpected outputs occur?
- Portability: Can the AI solution be deployed, migrated, or executed across different environments, cloud platforms, or hardware accelerators without unacceptable degradation?
- Transparency: Can decisions be explained, monitored, audited, and evaluated for fairness, robustness, privacy, and accountability throughout the AI lifecycle?
- Fairness & Responsible AI: Has the model been evaluated for bias, consistency, and equitable performance across relevant user groups?
- Observability & Monitoring: Can failures, performance degradation, hallucinations, drift, and abnormal behavior be detected, logged, and investigated?

3.4.3 SG2 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. New non-functional product risks may become apparent during the AI system development 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 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. Note that non-functional quality characteristics, especially those related to model quality and drift, are also tested and measured after deployment in production using telemetry relying on observability. This should be documented as part of the test approach.

The non-functional AI system test approach is part of the overall test approach and will be maintained and 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., security testing demonstrates no successful prompt injection leading to disclosure of confidential instructions, the system safely handles malformed and adversarial inputs without crashes (for reliability) or classification accuracy

differs by no more than 5% across evaluated demographic groups (for fairness). The development of the AI-system should result in the implementation of the agreed set of non-functional requirements and meet the non-functional (test) exit criteria as defined in the DoD. However, testing doesn't stop when the AI system is released. In AI system context there is a strong focus on shift-right approaches, measuring product quality characteristics in production through continuous monitoring against defined criteria. Shift right testing involves moving testing activities closer to production environments, enabling real-world performance evaluation and continuous quality monitoring.

3.4.4 SG3 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. For AI-based systems, non-functional test analysis and design focus on the quality characteristics that have been selected as part of the non-functional product risk assessment and the established non-functional test approach. Depending on the type of AI-based system, these characteristics may include performance efficiency, robustness, transparency, explainability, fairness, adaptability, autonomy, privacy, security, resilience and other AI-specific quality characteristics.

Test analysis and design activities should identify the relevant test conditions, determine the required test data and define the appropriate acceptance criteria and metrics for evaluating the selected non-functional quality characteristics. Since AI-based systems are data-driven and may exhibit probabilistic or non-deterministic behavior, the design of representative test datasets and realistic operational scenarios is often an important aspect of this specific goal. In addition, edge cases and situations that may adversely influence the behavior of the AI model should be considered during test design.

The techniques used for non-functional test analysis and design will depend on the selected non-functional quality characteristics and the associated product risks. Examples include techniques for robustness testing, adversarial testing, fairness and bias testing, explainability testing and resilience testing. Guidance on applicable techniques, metrics and test design approaches for AI-based systems can be found in the ISTQB Certified Tester AI Testing syllabus [ISTQB]. Identifying observability and telemetry requirements and stories could also be an outcome of non-functional analysis and design, to help measure the non-functionals in production.

Specific test data necessary to support the execution of non-functional tests is identified. For AI-based systems, the test data required depends largely on the selected non-functional quality characteristics and may include representative, edge-case, synthetic or adversarial data. Whereas some test data may be prepared during test execution, automated and repeatable non-functional tests typically require the necessary datasets and acceptance criteria to be specified upfront. As some non-functional characteristics are evaluated using production-like or operational data, production data may also be used during testing. In such cases, privacy, security and applicable regulations, such as GDPR, must be considered.

The outputs of this goal typically include a specification of the non-functional test conditions, the required test data and environments, and the associated acceptance criteria and quality metrics. These artefacts provide the basis for the subsequent non-functional test implementation and execution activities.

3.4.5 SG4 Perform Non-Functional Test Implementation

Test implementation is about getting everything in place that is needed to start the execution of tests, to implement the non-functional tests that have been identified and designed during the previous testing phase. For AI-based systems, this also includes preparing the necessary test data, test environments, tools and test procedures needed to evaluate the selected AI-specific quality characteristics.

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 test 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.

The implementation of non-functional tests for AI-based systems is often more data-intensive and technically demanding than for traditional systems. Depending on the non-functional characteristics being tested, representative and sometimes specialized datasets need to be acquired, generated or prepared. For example, testing characteristics such as robustness, fairness, transparency and explainability may require dedicated test datasets that cover realistic, boundary and exceptional situations. In addition, test environments may need to simulate operational conditions or support large volumes of test data and repeated test executions.

Typical implementation activities include preparing and preprocessing test datasets, configuring test environments and simulation platforms, implementing automated test scripts and setting up mechanisms to collect and analyze test results. In some situations, synthetic or augmented test data may be used to improve coverage of rare or difficult-to-obtain scenarios. The implementation activities should also ensure that the selected quality characteristics can be measured in a repeatable and manageable manner.

As with traditional systems, all implemented test artefacts should be traceable to the identified non-functional product risks, test conditions and acceptance criteria. Although the implementation activities may differ significantly from those used with traditional systems, the intent of the TMMi goal remains unchanged: to ensure that the non-functional tests are properly prepared and ready for execution.

3.4.6 SG5 Perform Non-Functional Test Execution

As with the previous goal in this process area, we will largely refer 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 purpose of this specific goal is to execute the implemented non-functional tests and evaluate whether the AI-based system satisfies the defined acceptance criteria for the selected non-functional quality characteristics. The practices for execution of non-functional tests, reporting test incidents and writing test logs when testing AI-systems are basically the same as for functional testing. Note that non-functionals are also tested / measured after deployment.

Non-functional test execution is done in line with the priorities defined during planning. Some tests may be executed using a documented test procedure, but typically many non-functional tests will be executed using exploratory and session-based testing as their framework. With AI-systems and their continuous monitoring during operation, there is a high need to organize and structure regression testing also for non-functional testing using automated regression tests and supporting tools.

Non-functional test execution for AI-based systems often differs from traditional systems due to their probabilistic and sometimes non-deterministic behavior. Depending on the type of AI system and the non-functional quality characteristics being evaluated, test execution may involve running the model against representative and previously unseen datasets, performing repeated test runs and statistically analyzing the results obtained. The use of quantitative metrics is typically required to determine whether the non-functional requirements have been achieved. Typical examples of non-functional characteristics that may be evaluated during test execution include robustness, transparency, explainability, fairness, adaptability, resilience and operational performance. The selected metrics and acceptance criteria will depend on both the nature of the AI-based system and the associated product risks identified earlier in the process.

Unlike traditional systems, some non-functional characteristics of AI-based systems cannot always be fully assessed during pre-release testing. Characteristics such as bias, drift and degradation of model performance may only become visible when the system is exposed to real operational data over time. Consequently, non-functional test execution will typically also include activities that support continuous evaluation and monitoring of the operational behavior of the AI-based system after deployment.

Although the execution practices differ from those used with traditional systems, the intent of the TMMi goal remains unchanged. The goal is to provide objective evidence that the AI-based system satisfies its defined non-functional quality requirements and performs adequately in its intended operational context.

Process Area 3.5 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 SG1 Establish a Peer Review Approach

Peer review is an important defect detection technique during any development, regardless of the lifecycle model being used or type of product being developed. As such it is also a beneficial technique for the development and testing of. The peer review approach for AI-based systems should be lightweight, risk-based, and integrated AI systems into an iterative development cycles, while explicitly addressing the unique characteristics of AI such as non-determinism, data dependency, and probabilistic outcomes. Peer reviews should be planned as continuous activities embedded in the ML workflow, with a focus on early defect detection and quality control of both traditional test artifacts and AI-specific assets.

Defining a peer review approach starts by identifying and selecting the most critical work products to be reviewed. During the development and testing of AI systems, work products to be reviewed typically include AI/ML requirements and acceptance criteria (especially those defined in statistical or threshold terms), data-related artifacts (training, validation, and test datasets, data preparation procedures, labeling guidelines), model documentation (including assumptions, limitations, and performance metrics), test strategy, test approach and test cases (including those for bias, robustness, and adversarial conditions).

Given the exploratory and evolving nature of AI-systems, the review approach should preferably include:

- lightweight peer reviews and desk checks for iterative changes
- walkthroughs for complex artifacts such as data pipelines or model behavior

- inspections for high-risk elements (e.g., safety-critical models, ethical risk mitigation, or regulatory compliance).

Review participants should represent a multidisciplinary team to reflect the cross-functional nature of AI systems, including testers, data scientists, ML engineers, domain experts, and where relevant, ethics or compliance specialists. The reviewer roles should ensure both technical correctness (e.g., model performance, data quality) and alignment with AI-specific quality characteristics such as transparency, robustness, and fairness.

The peer review approach for an AI system could be documented in a simple table stating the artefacts to be reviewed, type of review and participants involved. This will allow for communication where all parties have an overview of the review approach and their involvement.

3.5.2 SG2 Perform Peer Reviews

As with any approach, it should not only exist as a defined and agreed-upon approach, but it should also be followed in practice. Peer reviews are especially important for identifying AI-specific risks that are difficult to detect later through dynamic testing alone. Examples include biased or incomplete datasets. As stated in the previous section, reviews often involve a broader range of stakeholders.

As with Agile software development, peer reviews within AI projects are typically lightweight, collaborative, and iterative. Reviews are frequently integrated into day-to-day development activities and performed continuously throughout the lifecycle due to the evolving nature of AI models and datasets. The objective remains unchanged: improving quality through early defect detection and shared understanding. However, for AI systems the emphasis shifts from reviewing only deterministic behavior to evaluating whether the AI system behaves responsibly, reliably, and within acceptable risk boundaries under varying operational conditions.

References

[AiU] Artificial Intelligence United (2019), AiU Certified Tester in AI (CTAI) Syllabus V1.0, Artificial Intelligence United

[ISTQB] International Software Testing Qualifications Board (2026), Certified Tester AI Testing (CT-AI) Syllabus V2.0, ISTQB

[ISO 25010] ISO/IEC 25010 (2023), *Software and systems engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model*, International Organization for Standardization

[ISO 25059] ISO/IEC 25059 (2023), *Software and systems engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality models for AI systems*, International Organization for Standardization

[Smith] A.L. Smith, R. Black, J. Davenport, J. Olszeweska, J. Röβler and J. Wright (2022), Artificial Intelligence and Software Testing, BCS Learning and Development,

[TMMiAgile] E. van Veenendaal (ed.) (2020), *TMMi in the Agile world V1.4,* TMMi Foundation