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.

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.

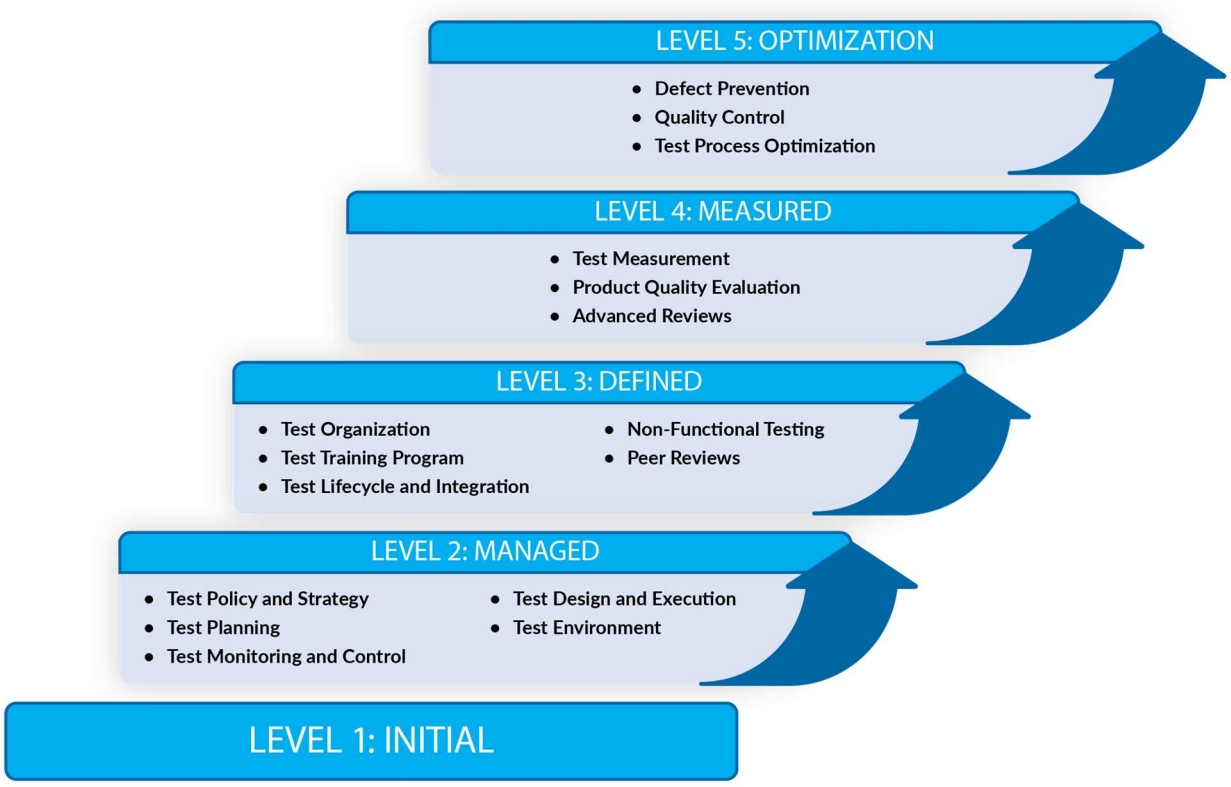

Figure 1: TMMi maturity levels and process areas

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.

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.

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.

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 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

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

Figure 1: Confusion Matrix

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

Examples of product risks for AI systems (continued):

- 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 will be 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.

References

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

[ISTQB] International Software Testing Qualifications Board (2021), Certified Tester AI Testing (CT-AI) Syllabus V1.0, ISTQB

[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