



How to Test AI-Based Systems Micro-Credential Syllabus

Copyright Notice

Copyright AT*SQA, All Rights Reserved

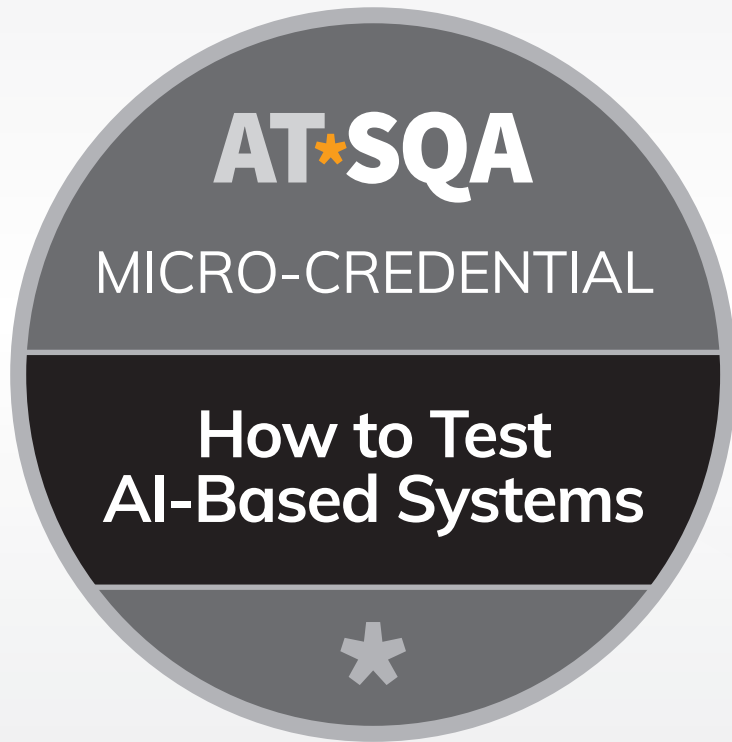


Table of Contents

How to Test AI-Based Systems

3	Why AI?
5	Keywords
6	Learning Objectives
7	Testing AI-based Systems
14	Testing Techniques for AI-based Systems
19	Testing Challenges
25	Test Oracles
26	Acceptance Criteria
28	Test Environments
31	Final Thoughts
32	Appendix A: References
32	ISTQB® Documents
32	Glossary References
32	Standards
33	Appendix B: Glossary
34	Purpose of this Document

Why AI?

Artificial Intelligence (AI) is here to stay. This is one of the major upheavals in technology, like the PC and the smartphone. It will change what we do, how we do it, and who will do it. AI is a large and emergent field, and any syllabus on the subject will be quickly outdated. This set of micro-credentials is designed to help the tester understand AI, Generative AI, and how this affects testing.

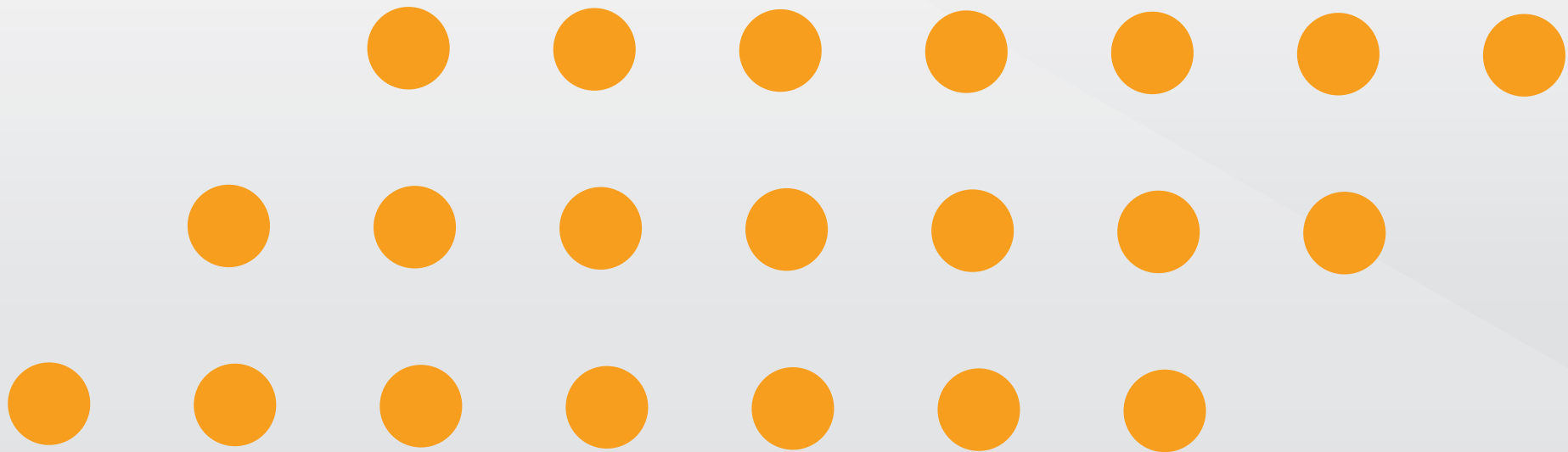
AI, in general, has a plethora of new terms associated with it. The first module of this stack of micro-credential syllabi acquaints the tester with the AI terminology

and the basics of AI. It will soon be an expectation that all testers are “familiar” with AI, and that will require having a basic understanding and the ability to learn as AI evolves. The “Introduction to AI” module covers the basics and touches on a bit of the technology behind AI. This is not intended to be a full and technical explanation of AI; there are many books already devoted to that.

The second module, “What to Test in an AI-based System” discusses how to identify the risk items and conditions that must be tested. It explains data privacy and security concerns with AI-based systems, and it highlights the quality characteristics that are unique to AI-based systems.

This module, “How to Test an AI-based System” explains how the traditional test levels can be applied to testing AI-based systems. It also discusses testing techniques that are particularly well-suited for AI system testing. The module includes a discussion of the challenges of testing AI systems.

The fourth module, “Using AI to Test”, explores how to use AI to help with testing. This looks at building test cases, building test automation, recording results, and automating many manual aspects of testing. Along the way, caveats are explored including what should be considered when AI is used. As the world has seen, AI is not always accurate.



Keywords

A/B testing, AlaaS, back-to-back testing, checklist-based testing, data poisoning, error guessing, exploratory testing, metamorphic testing, pairwise testing

LEARNING OBJECTIVES FOR HOW TO TEST AI-BASED SYSTEMS

Testing AI-based Systems

- (K2) Describe the general challenges with testing AI-based systems
- (K2) Summarize the considerations and goals in data testing
- (K2) Explain how the traditional testing levels apply to AI-based systems

Testing Techniques for AI-based Systems

- (K2) Explain how to apply test techniques to address risks in AI-based systems

Testing Challenges

- (K2) Explain the testing challenges that are common among AI-based systems
- (K2) Describe the specific challenges with testing autonomous, non-deterministic, and complex AI-based systems
- (K2) Explain testing for transparency, interpretability, and explainability

Test Oracles

- (K2) Describe how test oracles are used with AI-based systems

Acceptance Criteria

- (K2) Summarize the common acceptance criteria for AI-based systems

Test Environments

- (K2) Explain environment considerations for testing AI-based systems

Testing AI-based Systems

As with all testing, the clarity of the test basis will determine the accuracy of the testing. AI-based systems are particularly difficult to test because the exact outcome in a given situation may not be known. This is further complicated by issues with the requirements (test oracle) for a system.

There are a number of reasons for this, including the following:

- Requirements are often defined only at a high-level based on business goals. This requires predictions to be made regarding the outputs. AI-systems develop from the data sets used to train them. This means the predictions that are obtained from the training data sets are only as good and representative as the training data. With traditional systems, the logic is defined prior to testing. With an AI system, the logic evolves.
- Acceptance criteria usually are not specific. As the system evolves, so do the specifications and the criteria required for acceptance.

- AI-based systems are probabilistic in nature, which makes it necessary to understand the variances in “correct” outputs. For example, the metrics defined to determine the “accuracy” of a system can be applied to determine if the system is responding within an acceptable level of accuracy.
- When the system is expected to act like a human, the specifications are often vague. There is a wide variance in human decision making and the system rarely replicates this accurately across many variables.
- If the system has multiple user interfaces, such as natural language recognition, it must be flexible. While this flexibility is a requirement for a successful implementation, it adds complexity to testing.
- AI-systems have different quality characteristics from traditional systems. As noted in the previous micro-credential, some of these are quite difficult to test (i.e. evolution).

It is important to remember that while it is challenging to test AI-systems, this is still a relatively new area of software testing. It is important to stay current in the latest developments and to encourage the development team to spend time clearly defining their expectations to allow adequate testing.

Test Levels

While this syllabus has been focused on the AI-based components of a system, it is important to remember that there are usually non-AI components as well. Those traditional elements still require traditional testing. When developing test plans, it is important to specify the testing that will be conducted on all the components as well as the system testing of the entire system.

As explained in CT-AI, there are two new areas of testing that must be considered in the test levels for an AI-based system. These two areas cover the input data and the model itself.

DATA TESTING

The input data is used to train the system. The more representative the data is of the operational data, the more predictably the system will perform. Unfortunately, there is a long list of quality issues that can occur with data. As shown in CT-AI, the following quality aspects need to be considered:

Quality Aspect	Description
Wrong data	The captured data was incorrect (e.g., through a faulty sensor) or entered incorrectly (e.g., copy-paste errors).
Incomplete data	Data values may be missing (e.g., a field in a record may be empty, or the data for a particular time interval may have been omitted). There can be various reasons for incomplete data, including security issues, hardware issues, and human error.
Mislabeled data	There are several possible reasons for data to be mislabeled.
Insufficient data	Insufficient data is available for patterns to be recognized by the learning algorithms in use (note that the minimum required quantity of data will vary for different algorithms).
Data not pre- processed	Data should be pre-processed to ensure it is clean, in a consistent format and contains no unwanted outliers.
Obsolete data	Data used for both learning and prediction should be as current as possible (e.g., using financial data from several years ago may well lead to inaccurate results).
Unbalanced data	Unbalanced data may result from inappropriate bias (e.g., based on race, gender, or ethnicity), poor placement of sensors (e.g., facial recognition cameras placed at ceiling height), variability in the availability of datasets, and differing motivations of data suppliers.
Unfair data	Fairness is a subjective quality characteristic but can often be identified. For example, to support diversity or gender balancing, selected data may be positively biased towards minorities or disadvantaged groups (note that such data may be considered fair but may not be balanced).
Duplicate data	Repeated data records may unduly influence the resultant ML model.
Irrelevant data	Data that is not relevant to the problem being addressed may adversely influence the results and may lead to wasting resources.
Privacy issues	Any data use should respect the relevant data privacy laws (e.g., GDPR with relation to individuals' personal information in the European Union).
Security issues	Fraudulent or misleading data that has been deliberately inserted into the training data may lead to inaccuracy in the trained model.

If data quality issues exist in a system, it may exhibit any of the following negative characteristics:

- **Reduced accuracy** – accuracy is dependent on the data being correct, complete, labelled properly, valid, relevant, and, if needed, pre-processed
- **Biased** – data that is not complete, unbalanced, unfair, lacking the necessary range of diversity, or duplicated, will tend to create biases
- **Compromised** – data that violates data privacy rules or introduces security vulnerabilities can result in a compromised model

In order to ensure that the training data is fit for use, it is important to understand the data in the operational environment. A close alignment of the two will lead to a more accurate system and will help prevent or at least minimize these types of problems. Feature engineering is often used to better target the data. Feature selection is used to select the features that are most suited for use in training the model. Feature extraction is used to derive informative and non-redundant features from the existing data features. This helps reduce

the size of the data set and also improves model accuracy after training.

In addition to considering the training data, the data used for validation of the model (during development) and for testing the model (when development is thought to be complete) is also important. All of these data sets must meet the same quality characteristics in order to avoid false negatives or false positives during implementation and testing.

Testing the data itself requires reviewing the extraction algorithms (if data is extracted from other systems), transformation algorithms where data is extracted and then transformed, and any generation algorithms where data is created to ensure none of the above issues are being introduced. Statistical techniques can be applied for testing such items as bias. It is important to consider both component testing and integration testing when looking at data usage, since a particular set of data may work correctly for a component but may not be processed correctly by the integrated system.

It is also important to consider the different data pipelines. There is often a different data pipeline used for development and testing than the pipeline used in production. Ideally, both pipelines are tested, but where the production pipeline cannot be tested, it should be simulated and carefully reviewed to understand any differences from the test environment.

MODEL TESTING

As discussed above, the model itself should be subjected to white-box testing for the algorithm and use of data. It should also be subjected to testing to determine if the functional performance criteria are being met (i.e., accuracy, precision, recall, F1-score). The other quality criteria specific to the model, such as speed to train, speed to produce the prediction, use of computing resources and power, adaptability, and transparency, should also be tested as part of model testing.

As with the data testing, the model itself must be tested, but it must also be tested in the fully integrated environment to ensure it is still working

correctly. While full integration may be difficult to achieve in a test environment, at a minimum the data pipeline into and out of the model must be validated.

COMPONENT TESTING

Component testing of the AI-system applies to the model and data, as explained above, and the other components of the system that comprise the entire system.

COMPONENT INTEGRATION TESTING

As with conventional systems, the goal of component integration testing is to ensure the components work together. In AI systems it is particularly important to verify the data pipeline and that the output from the AI model is consumed correctly by the receiving components.

SYSTEM TESTING

As with conventional systems, the goal of system testing is to ensure the system as a whole provides the required functionality in a usable manner. This includes both the functional and non-functional quality characteristics. System testing is usually done to verify that the system meets the specified requirements, but AI systems tend to have requirements that are not definitive for testing. System testing may be conducted using controlled field trials or may even be conducted with simulators that can generate many inputs (such as for an autonomous car).

Performance testing is often conducted again at this level to ensure the fully integrated system meets the defined performance metrics. Other non-functional testing is usually conducted at the system level.

ACCEPTANCE TESTING

Acceptance testing for AI-based systems is the same as for conventional systems. The goal is to determine if the operational system is acceptable to the targeted users. Although the acceptance criteria may be loosely defined, the users will still be able to provide input regarding what they are expecting and what they have received. When the AI-based system is provided as a service (AlaaS), the suitability of the service will be evaluated.

AlaaS systems usually have service-level agreements (SLAs) that must be met for the system to be accepted. These SLAs often include such items as required uptime/availability and response time in the case of a defect. This is a growing area for AI systems and it is important for the tester to understand the SLAs that are being promised for a system that is being tested.

Testing for Automation Bias

Automation bias is the tendency for humans to be too trusting in the automation provided by the AI-based system. There are two forms of this type of bias:

- The human accepts what the system recommends without considering other options.

A good example of this is when AI is used to auto-complete sentences. It is not unusual for the human to just accept what has been proposed rather than using their own mind to realize that the words supplied are not correct, or at least not as accurate as they could be.

- The human misses a failure because they are not monitoring the system as well as they should. Autonomous cars are a good example of this because drivers tend to rely on the car and are slow to react when intervention is needed.

Testers need to think about how the human user can become complacent and dependent on the automation to handle decisions that should be validated by a human. Systems may need more warnings to help remind the human user that

intervention is expected at times.

Testing for Concept Drift

When the operational environment changes, but the trained model doesn't, concept drift can occur. This usually results in the trained model producing outputs that are less and less accurate and useful. COVID-19 is a good example of an environmental change that was not anticipated when models were trained. Imagine an algorithm used to order toilet paper that was not anticipating the panic buying brought on by a pandemic.

Less disruptive changes occur in system environments regularly. In order to test for these, deployed systems should be regularly tested to ensure they are still meeting their functional performance criteria. Because the AI system is always learning and adjusting, relatively subtle environmental changes can cause significant drift over time.

Testing Techniques for AI-based Systems

Because AI-based systems are usually embedded into traditional systems, the testing techniques discussed in the ISTQB Foundation Level syllabus (ISTQB, Core Foundation, 2025) still apply. In addition to those techniques, some new techniques and modifications to the old ones apply specifically to testing the AI part of systems. It is important to remember that an AI-based system is continually evolving. Testing it only prior to deployment will not ensure it is continuing to provide correct predictions.

Adversarial Attack Testing

Attackers can “perturb” valid inputs, making them invalid, but in ways that are difficult to detect. For example, image classifiers can be tricked by changing only a few pixels of an image (undetectable to the human eye) which will then cause the neural network to incorrectly classify other images.

There are two types of adversarial attacks:

- White-box attacks are conducted when the attacker has access to the internals of the model such as its settings and parameters. The attacker uses this knowledge to generate adversarial data and can observe how this data is consumed and processed. With this type of access, the attacker can effectively tune the attack data until the desired results are achieved.

- Black-box attacks are conducted when the attacker creates a duplicate model of the system that is to be attacked. This this duplicate model, the attacker can conduct a white-box attack on the duplicate as explained above. The attack data should be transferable between the models, so once the attack data is developed, it is then fed to the system under attack.

Testing for these types of attacks is two-fold. Ideally, the attacker should not be able to access the information needed to conduct a white-box attack, so protecting that information is the first way to close the vulnerability. Testing for the second type of vulnerability is done by acting like an attacker and feeding in attack data to see if the system can detect it and protect itself.

Data Poisoning Testing

Data poisoning attacks occur when an attacker manipulates the training data to either introduce corrupted or invalid data, or to create a future “back door” that can be accessed for malicious means. Bad data can be added during training, but also during operational use. This can be done by flooding the model with false data (such as an orchestrated spreading of misinformation via social media) or training it through providing incorrect feedback to the system, convincing it to change its predictions.

Testing to detect data poisoning is usually done by periodically testing the operational system with the test dataset to ensure the responses being given still align with the acceptance criteria. This is usually done using A/B testing (discussed further below). Periodic regression testing of the system using the trusted test data is also a good way to determine if a system has been poisoned.

Pairwise Testing

Because there are so many parameters at play with an AI-based system, pairwise testing often becomes a necessity to help reduce the parameters to be tested. Using any of the combinatorial techniques will help to reduce the amount of testing to a manageable set. Consider the huge sets of parameters that must be consumed by a self-driving car. Testing all the combinations of the parameters would take too long and is likely unnecessary. Practical and statistically accurate reduction of the parameters using the combinatorial techniques such as pairwise testing will allow the testing to be conducted in a reasonable timeframe without significantly increasing risk.

As with any system requiring the testing of large sets of parameters, test automation is helpful. The non-deterministic nature of AI-based systems makes definition of the expected result more challenging, but by using a larger range of “acceptable” behavior, the automation can be implemented.

Back-to-Back Testing

Back-to-back testing is conducted by providing the same inputs to the system under test (SUT) and another system that has been developed independently (i.e., using different frameworks, algorithms, and architecture) which serves as the pseudo-oracle. The outputs are then compared between the SUT and the pseudo-oracle with the expectation being that they would both fall within the acceptable parameters for output quality.

Pseudo-oracles can be very expensive and time-consuming to develop. A legacy system can be used as the pseudo-oracle, which saves time and money. Developing a full pseudo-oracle is usually done only in safety-critical applications or where a failure or deviation in the primary system could cause catastrophic consequences.

A/B Testing

A/B testing is done to compare the results from a new variant of the system (B) with the previous version (A). If the results are similar, or better, the new variant is accepted. If there is a degradation in the results, system B is usually rolled back and system A stays in use. This type of testing can be done with production systems where some users are routed to system A and others use system B. The results are then compared. This can also be used to determine if the system is learning correctly by comparing its outputs to a previous version that has processed the same data.

Metamorphic Testing

Metamorphic testing addresses the issue of not having a good test oracle that tells the tester that, given a specific input, a specific output should result. AI systems are non-deterministic, meaning that they will not necessarily provide a specific result and may indeed provide a different result

even when given the same inputs. Metamorphic testing uses the relationship (metamorphic relations) between the input and the expected output to help determine what is an acceptable output. For example, if the input is a query to find all the five-star restaurants in Seattle, it would be reasonable to assume that the query would provide a number smaller than a query for all the restaurants in Seattle. Additional queries could be put forth asking for the number of five-star restaurants with ocean views, which would return an even smaller number. While this only provides a broad stroke regarding the accuracy of the output, it is a quick way to test that the processing is generally correct when the input is varied.

Experience-Based Testing

As with traditional testing, experience-based testing for AI systems includes error guessing, exploratory testing and checklist-based testing. All of these testing techniques are applicable to testing AI-based systems and work in much the same way as they do with traditional systems.

Error Guessing is based on the tester's knowledge of the likely errors made by the developers, problems seen with similar software, and the tester's experience with testing in general. Using this information, the tester targets likely errors that have been made during system development and seeks to induce those errors during testing.

Exploratory testing is conducted by iteratively designing, creating, and executing tests. Based on the results of the tests, additional tests may be generated. Because exploratory testing works well when system specifications are poor or non-existent, it is an effective technique with AI-based systems. Determining if the output is correct

requires a combination of logic, knowledge of the system, knowledge of the expected usage of the system, and understanding the expectations of the users.

Checklist-based testing is conducted using a checklist to guide the testing. There are publicly available checklists for testing AI-based systems, but homegrown checklists are often used based on the experience of an organization. The checklist is designed to be a guideline to help the tester remember to test specific characteristics of the system. Checklists are normally living documents that are updated when testing gaps are discovered.



Testing Challenges

Testing AI systems is challenging, to say the least! Per the CT-AI syllabus, there are challenges that are unique to AI-based systems.

Common Testing Challenges

AI-based systems provide challenges that don't usually exist in traditional systems which produce predictable outputs and don't change over time. The following is a list of the most common challenges:

- **Unexpected change:** While there may be requirements and acceptance criteria for the initial system, the system will modify itself as it learns. While tests can be designed to ensure the system meets the original specification, designing tests for what it might become is a lot more difficult. Even re-running tests that passed with the original version of the system may result in unexpected results. These results have to be evaluated to determine if they are still within the parameters of acceptable behavior for the system. If the system has learned to use a different approach to solving problems, the original tests may no longer be valid and new ones will be required.
- **Complex acceptance criteria:** Defining flexible acceptance criteria that can be used to test an evolving system is a difficult task. Because the system is learning and adapting, the original assumptions regarding how the system will work may become invalid. It's important to frequently measure the functional performance (e.g., accuracy, precision) to ensure the system is still performing within acceptable parameters. Consulting with the users in the operational environment will help to determine if the system is changing in an unacceptable manner, but this is a time-consuming effort and requires that the users are able to see enough of the data to determine if there is an issue. If they only ever see the output, without the input, they may not be able to validate if it is correct.

- **Insufficient testing time:** While this is almost always an issue with traditional systems, it is worse with AI-based systems. Lack of well-defined specifications and acceptance criteria is often an issue with AI-based systems, which makes test definition more difficult and time-consuming. Add to that the need for continuous testing after deployment to ensure that the system is still functioning within acceptable boundaries. Traditional systems require on-going testing when their operational environment changes or when known changes are made to the system. AI-systems are in a continuous learning and evolving mode, which requires continuous testing. Automation is the only way this can be efficiently addressed, but automation works off comparing the actual to expected results, which can be difficult to predict with an evolving system. Automation must be developed to understand and evaluate a potential set of responses to determine if they are correct.
- **Resource requirements:** In addition to its operational requirements, the system may also have a need for resources that are consumed as it learns. These resources may be physically limited, or may be defined as limited. Testing may be required to ensure the system is not using more resources than were allocated.
- **Insufficient specifications of the operational environment:** Operational environments can evolve just as the system can evolve. When this occurs, the system may be receiving inputs that are outside the parameters of what it was trained to handle. For example, if a right-side drive system is used in a left-side drive country, significant errors would result. Ideally, testing should cover any expected or anticipated changes to the environment in which the system is intended to operate.

- **Complex and expensive test environments unavailable:** As with any testing project, there may not be sufficient money or time to create suitable test environments that accurately mirror production. When this occurs, simulation of conditions may be required in order to gauge the system's responses. Consider the complexity required for the test environments of autonomous cars.
- **Undesired behavior modifications:** During testing, bad or invalid data is provided as inputs as well as good data and usually in proportions that would not be encountered in the operational environment. This can result in the system being trained on bad data because of the volume it experiences during testing. It is important to ensure that the system does not modify its behavior in a negative way just because the test data has reinforced the negative behavior.

Challenges with Autonomous AI-based Systems

One of the characteristics of autonomous systems is that they do require human intervention. Testing requires that the tester induces the need for human intervention and tests that the system requests the appropriate intervention, that it accepts the intervention, and that it doesn't request intervention when it is not needed.

While boundary value analysis can be used to determine which conditions should cause a request for intervention and which should not, creating the actual conditions can be time-consuming. Intervention conditions often require feeding something into the system that it was not expecting, but the system will learn how to handle these inputs, so new ones will need to be created that haven't been experienced yet.

Intervention requests are easier to test when they are in response to anticipated circumstances, such as an autonomous car needing fuel or recharging. The ease of defining these test conditions depends on the clarity of the requirements, the defined acceptance criteria and, most importantly, common sense.

Testing Non-Deterministic Systems

Non-deterministic, including probabilistic, systems often have multiple acceptable outcomes for a given set of inputs. Unlike traditional systems, using the same preconditions and input values may result, correctly, in differing outputs. This makes testing difficult since the expected outputs may not be definable.

Building repeatability into tests is something testers strive to do, but the lack of a predictable outcome requires a wider definition of an acceptable output to achieve repeatability. This often comes down to a “reasonableness” check. Is the answer correct and within the defined constraints?

Although a test may not be truly repeatable to generate the same response, tests should be repeated to help define the set of acceptable answers. This will assist with developing test automation and setting up a form of repeatability

in the tests. In text prediction software, there can be a set of words that would logically be predicted. Defining and adding to that set as testing progresses allows future tests to be automated not to seek a specific answer, but rather an acceptable answer.

Testing Complex AI-Based Systems

It is difficult, if not impossible, to test what we cannot understand. AI-based systems can provide very complex processing and can perform functions that are beyond the capability of a human (such as understanding all the nuances in facial recognition). Testing these areas means that the tester needs to have a way to predict the outcome. This is sometimes done by providing the same input to two similar systems to check the outputs against each other. This, of course, requires having access to two similar systems that are close enough in functionality to serve the purpose.

In some cases, the code that is running the system has been generated by AI, which makes it even more difficult to perform any white-box testing. When black-box testing is the only option, the reasonableness test has to be applied to the outputs. Does the output lie in the set of “acceptable” outputs? For example, the tester can test a facial recognition system without understanding the internals of the decision making. They might miss a defect that is only seen when two people have identical eyebrows, but that is a risk that has to be accepted with AI systems. This is why monitoring in production is so important and why continuous testing must occur after deployment.

AI-systems are only becoming more complex, with systems interacting with each other. If each interacting system is returning non-deterministic results, the variability in the outputs can quickly become unmanageable. If there is sufficient understanding of how the inputs relate to the outputs, combinatorial testing techniques can be applied to help reduce the expected output set. Key to testing these complex systems is understanding the intentions and defining the set of “acceptable” outputs.

Testing for Transparency, Interpretability and Explainability of AI-Based Systems

As has been discussed before in these micro-credentials, the accuracy and effectiveness of testing are highly dependent on understanding the requirements for a system. Each of these characteristics of the system must be built-in when the system is designed. For systems that are acquired, such as AlaaS, these characteristics may not be present or visible for the tester.

The primary benefit of transparency is to increase the trust that the users and other stakeholders of the system can have in the system. It builds understanding and confidence. Transparency is tested by comparing the specifications for the data and the algorithm with the implemented system. If there are no specifications, or if the specifications are unclear, this testing will likely not be productive.

Similar to transparency, interpretable systems help to build trust with the users, but can also help to ensure fairness, promote ethical decision-making and may be required for regulatory requirements. The more interpretable the system is, the more maintainable it is. Interpretable systems allow the developers to understand the likely effects of changes. For testing, understanding how the system is reasoning will help to target tests to the critical functionality. To determine what to test, the tester must understand who needs to interpret what the system is doing, and why. In this way, the tester can target the testing to verify that interpretation is possible. For example, why did the chatbot quote an out-of-date regulation that it says is required?

Explainability is closely aligned to transparency and serves the same purposes. The more explainable the input/output relationship is, the more confident users will be. Testing is focused the same as with transparency, focusing on making sure the actions of the system can be explained in a form required by the consumers of the information. For example, a system that is calculating complex financial numbers may need to be able to show the steps of its calculations in order to meet the requirements of regulators.



Test Oracles

A test oracle is the source or a mechanism that is used to determine if the output from the system is correct. This is usually done by comparing the output to the defined expected result. In AI-based systems, there is sometimes no test oracle to use because non-deterministic answers are not exact. If the requirements specify a range of acceptable responses, such as how close to an object an autonomous car must stop, then that can be used as the oracle. For items such as text predictors, the range of what is “correct” can be huge.

This syllabus has referred to the “reasonableness” test – does the answer fit within what the user would expect given the inputs? This assumes that the tester has enough knowledge to determine what is reasonable – and that their definition of reasonable matches the user’s. For example, if the user asks an AI-based personal assistant to “map out my day”, what does that mean? Generate a driving map of where to go? Create a schedule? The acceptable answer depends on the user’s expectations.

In AI systems, experts are often used to determine if the output is correct. There are some issues with experts though:

- Experts may not be available or affordable
- The technology may be so new that there are no experts
- Experts vary in their levels of competence and breadth
- Experts can disagree with each other
- Humans tend to caveat their responses (e.g., “I think that’s right”, “That sounds right, but...”)

As discussed in Testing Techniques, techniques such as A/B testing, back-to-back testing and even metamorphic testing can help with determining the correct outputs and the changes to those outputs as the system learns. The better informed a tester is regarding the user’s expectations and uses of the system, the better that tester will be able to target the testing.

Acceptance Criteria

As has been discussed, defining testable acceptance criteria is very difficult. These need to be based on the various aspects of the system and the unique nature of AI-based systems. The following table from CT-AI provides a checklist for how these criteria can be approached and helps with working with the team to define the criteria that are acceptable for a particular AI implementation.

Aspect	Acceptance Criteria
Adaptability	<ul style="list-style-type: none"> • Check the system still functions correctly and meets non-functional requirements when it adapts to a change in its environment. This may be implemented as a form of automated regression testing. • Check the time the system takes to adapt to a change in its environment. • Check the resources used when the system adapts to a change in its environment.
Flexibility	<ul style="list-style-type: none"> • Consider how the system copes in contexts outside the initial specification. This may be implemented as a form of automated regression testing executed in the changed operational environment. • Check the time the system takes and/or the resources used to change itself to manage a new context.
Evolution	<ul style="list-style-type: none"> • Check how well the system learns from its own experience. • Check how well the system copes when the profile of data changes (i.e., concept drift).
Autonomy	<ul style="list-style-type: none"> • Check how the system responds when it is forced outside of the operational envelope in which it is expected to be fully autonomous. • Check whether the system can be “persuaded” to request human intervention when it should be fully autonomous.
Transparency, interpretability and explainability	<ul style="list-style-type: none"> • Check transparency by reviewing the ease of accessing the algorithm and dataset. • Check interpretability and explainability by questioning system users, or, if the actual system users are not available, people with a similar background.
Freedom from inappropriate bias	<ul style="list-style-type: none"> • Where systems are likely to be affected by bias, then this can be tested by using an independent bias-free test suite, or by using expert reviewers. • Compare the test results using external data such as census data in order to check for unwanted bias on inferred variables (external validity testing).
Ethics	<ul style="list-style-type: none"> • Check the system against a suitable internationally accepted checklist. These checklists are updated frequently, so always verify that testing is being conducted against the latest version that is applicable to the region of usage.

While this table will likely expand as AI evolves, it is a great starting point for the team. Each point should be discussed openly and the team should agree on how these items will be documented and tested. This also provides an opportunity for the tester to ask “what else?”. Testing an AI system requires input from the entire team and it’s important to ask the questions and then review the answers to be sure everything has been captured. This will help prioritize and target the testing and will also bring everyone to the table to discuss the difficulties with testing.



Test Environments

AI-based systems run in many different environments. Because AI systems are input data dependent, creating a test environment that has a representative set of data can be very complex. Some of the special considerations for AI system environments are dependent on the type of system being created.

The considerations include the following:

- Self-learning systems are expected to adapt to an environment that is changing. The operational environment may not be completely definable at the time of deployment, which means any changes to the operational environment should be mimicked in the test environment. Unfortunately, this is rarely known at the time when testing is commencing, so the testers need to work with the developers and operational users to understand what is likely to change.
- Autonomous systems are expected to respond to changes in their environment with little or no human intervention. They also need to demonstrate an understanding of when human intervention is needed. Since autonomous environments are dependent on many incoming stimuli, replicating or simulating this can be difficult and expensive. Again, working with the

team to understand the necessary environment and to build the necessary simulators is required so adequate time and money are allocated.

- Multi-agent systems work with other AI-based systems. This requires a test environment that can host the integrated systems, can feed the necessary data into all systems for coordinated testing, and can handle the evolution of those systems. These environments often span the control of multiple teams and system administrators, and coordination is critical. Fully integrated test environments are rare, so integrations may need to be tested individually which is less desirable than all the systems working together.
- Hardware is required for all AI-based systems, and this may be located in the cloud or on the premises. Specifically tuned AI processors may be required, particularly if learning is expected. Test systems often use scaled down data sets which can help control the capacity needed for

the AI system. It is important to understand exactly what data sets will be used, how large those sets are, and what processing is expected (e.g., exercising a model that is already trained, or training a model) to size the systems correctly.

In addition to physical systems, virtual systems may be used for the test environment. This may provide more control over the environment, particularly when simulators are used. The virtual system must be representative of the production system or there is a risk that the test results won't represent production responses. Similarly, if simulators are developed, they must act like the real environment. There are both commercial and open-source virtual test environments available. This can be a cost-effective approach particularly for systems where using the real environment is not practical, too expensive, or dangerous.

As with other testing decisions, it is helpful to involve a wide set of stakeholders when determining the best test environments. This will help to procure the time and money needed to

create the systems and will help to ensure that the results will be considered representative. Test environments can be very expensive, so ensuring that the proper budget is allocated at the beginning of the project will help reduce the pain later. It is also important to remember that test environments for AI systems must be kept in working order as long as the system is deployed. Continuous testing is required and the test environments will be in continuous use throughout the life of the system.



Final Thoughts

Particularly with AI systems when so much variability in the outputs is expected, the tester must clearly define with the team the boundaries of the testing. Otherwise the expectations that “it will just work” will exist, even if unstated. In most organizations, the default expectation is that testing will cover 100% of the functionality in all its forms. As testers, we know this is unreasonable, but it still persists. It is important that the entire team understands the difficulty of testing, the lack of a reliable oracle, and the need for continuous testing as the system evolves.

Testing AI systems is not easy and requires significant understanding of the system, skills in testing, and flexibility to adapt as the larger AI environment changes. It’s an exciting time for testers, but new challenges will continually arise. Staying current with the latest testing techniques will be helpful as these will continue to evolve as the systems under test evolve.

Appendix A: References

ISTQB® Documents

[CT-AI] ISTQB Certified Tester – Artificial Intelligence Syllabus, v1.0, 2021

[CT-GenAI] ISTQB Certified Tester – Generative AI Syllabus, v1.0, 2025

Glossary References

ISTQB® Glossary <https://glossary.istqb.org/>

Standards

ISO. (n.d.). Responsible AI. Retrieved from <https://www.iso.org/artificial-intelligence/responsible-ai-ethics>

The previous references point to information available on the Internet and elsewhere. Even though those references were checked at the time of publication of this syllabus, AT*SQA cannot be held responsible if the references are not available anymore.

Appendix B: Glossary

This glossary is composed of excerpts from the [CT-AI] and [CT-GenAI] syllabi. Refer to those syllabi for additional terms and references.

A/B testing: A test approach in which two variants of a test object are statistically evaluated to determine which performs better for specified characteristics.

AlaaS: AI as a system

back-to-back testing: A test approach in which a pseudo-oracle is used.

checklist-based testing: An experience-based test technique in which test cases are designed to exercise the items of a checklist.

data poisoning: An adversarial attack where corrupted or misleading data is introduced into a machine learning model's training set to manipulate its behavior.

error guessing: A test technique in which the tests conditions are based on the tester's knowledge of past failures or failure modes.

exploratory testing: A test approach in which tests are dynamically designed and executed based on tester's knowledge, exploration of a test item, and previous test results.

metamorphic testing: A test technique in which test conditions are metamorphic relations.

pairwise testing: A black-box test technique in which test cases are designed to exercise pairs of parameter-value pairs.

Purpose of this Document

This syllabus forms the basis of the AT*SQA certification for AI for Testers. AT*SQA is an International Standards Organization (ISO) compliant certification body for software testers. AT*SQA provides this syllabus as follows:

1. To training providers - to produce courseware and determine appropriate teaching methods.
2. To certification candidates - to prepare for the exam (as part of a training course or independently).
3. To the international software and systems engineering community - to advance the profession of software and systems testing, and as a basis for books and articles.

AT*SQA may allow other entities to use this syllabus for other purposes, provided they seek and obtain prior written permission.

This syllabus has been constructed to be tool-agnostic, but tools will be discussed where they are commonly used. When tools are referenced, this is not a recommendation for the use of a particular tool, but examples of commonly used tools to help clarify points.

There are no prerequisites required for this certification. The certification can be achieved by passing the assessments for all four AI micro-credentials. As a part of AT*SQA's ISO compliant offerings, the certification must be kept current with additional learning completed within the defined timespan. This helps software testers to continue to expand their knowledge and marketability and acknowledges the very real need for continuing education in the software testing industry. For more details, see AT*SQA's website.

This syllabus has been built from information in the ISTQB Certified Tester – Artificial Intelligence [CT-AI] and Certified Tester – Generative AI [CT-GenAI] syllabi, as well as information gathered from multiple other sources. This is an introduction to both of these syllabi, but it will not provide a full preparation for the ISTQB exams. For those certifications, it is best to study the full syllabi and consider attending some focused training.

AT*SQA

MICRO-CREDENTIAL

**How to Test
AI-Based Systems**



www.atsqa.org

