SYLLABUS

Version 2022

AT*SQA
ASSOCIATION FOR TESTING & SOFTWARE QUALITY ASSURANCE
Global Certification Body for ISTQB and ASTQB

0. Introduction to this Syllabus

0.1. Purpose of this Document

This syllabus forms the basis of the AT*SQA certification for Testing Essentials. 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.

0.2 What is Essential?

The Information Technology (IT) world changes almost continuously as new technologies and techniques are adopted. Software testers (whether by title or in practice) must adapt quickly and be able to leverage their skills to meet new challenges. However, the essential skills and knowledge remain the same, serving as core understanding to which new information can be added. For the sake of readability, the term “software tester” will be used to refer to anyone who is testing software, regardless of their formal role.

This syllabus focuses on the essential areas of software testing that are required, regardless of the technology, lifecycle or tools in use. Some projects may use more or less of these skill areas, but all software testers need to understand and master this core skill set.

As the name indicates, this syllabus covers the “essentials”. This syllabus should be considered a springboard for additional certifications and knowledge areas. As a part of AT*SQA’s ISO compliant offerings, the certification must be kept current with additional learning completed within the defined timespan. For more details, see AT*SQA’s website. 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.

0.3 Syllabus Structure

This syllabus has been constructed to be tool and methodology agnostic. In places where different approaches are needed based on different lifecycles, those areas are highlighted with appropriate recommendations for tailoring the approach.

The intended target audience for this syllabus is anyone conducting software testing, whether or not they have the title of software tester. This includes Scrum team members, developers, Business Analysts (BAs), software specialists and anyone interested in learning the important aspects of software testing.

This syllabus is intended to be read in full, but if the reader is interested only in a specific area, each area can be read independently. It is recommended that the Test Approach and Testing Techniques sections (Sections 2 and 3, respectively) are considered compulsory reading, as these are generally applicable to any of the specialist areas of testing and provide a good background to general testing practices.

0.4 Examinable Learning Objectives

Each chapter notes the time that should be invested in learning and practicing the concepts discussed in that chapter. This information should be used as a guideline when creating training materials or for an individual conducting self-study.

All identified key terms are examinable, either individually or by use within an exam question. Full definitions for the key terms can be found in the AT*SQA glossary (see www.atsqa.org).

The Learning Objectives for each chapter are shown at the beginning of the chapter and are used to create the examination for achieving the Testing Essentials Certification. Learning objectives are allocated to a Cognitive level of knowledge (K-Level). A K-level, or Cognitive level, is used to classify learning objectives according to the revised taxonomy from Bloom [Anderson00]. AT*SQA uses this taxonomy to design all examinations.

This syllabus considers four different K-levels (K1 to K4) as noted for each Learning Objective (LO):

K-Level	Keyword	Description
1	Remember	The candidate should remember or recognize a term or a concept.
2	Understand	The candidate should select an explanation for a statement related to the question topic.
3	Apply	The candidate should select the correct application of a concept or technique and apply it to a given context.

4	Analyze	The candidate can separate information related to a procedure or technique into its constituent parts for better understanding and can distinguish between facts and inferences.

In general, all parts of this syllabus are examinable at a K1 level. That is, the candidate will recognize, remember and recall a term or concept. Other specific learning objectives are shown at the beginning of the pertinent chapter.

1. Introduction to Software Testing – 60 mins.

Keywords

requirements, test case, test condition, test plan, test strategy

Learning Objectives for Introduction to Software Testing

1.1 What is Software Testing

LO-1.1.a (K2) Summarize the various forms of requirements
LO-1.1.b (K1) Recall the meaning of “fit for purpose”

1.2 A Brief History

LO-1.2.a (K1) Recall the difference between a test engineer and a test analyst

1.3 Structured Testing

LO-1.3.a (K2) Explain the purpose of the documents used in a structured testing environment

1.4 The Role of a Tester

LO-1.4.a (K1) Recall who can be a software tester

1.1. What is Software Testing

Software testing has variable meanings. The term has evolved as new software development lifecycle (SDLC) models have been introduced. Regardless of the changes to the exact definition, software testing is an activity, or set of activities, that are conducted to evaluate software to determine the following:

- Have the requirements been met?
- Is the software “fit for purpose”?
- Has the risk been reduced enough?
- Have important defects been identified and addressed?

Each of these questions tends to elicit more questions.

1.1.1. Requirements

Software requirements come in many forms including:

- Formal requirements documents prepared by Business Analysts (BAs)
- Technical requirements documents, such as functional specifications, design documents, and interface design documents

- Higher level documents, such as use cases which describe how an expected user would accomplish tasks or goals by using the software
- SDLC unique documents, such as user stories in the Agile lifecycle model
- Very informal diagrams on white boards and results from workshops
- Word-of-mouth and drawings in a highly collaborative environment (where the team is all working together, all the time)

The ability to verify that the software meets the requirements is dependent on the clarity of the requirements. If a requirement is clear and defines exactly what the software is supposed to do, the verification is straightforward. Where the requirements are vague or missing, the tester must be able to apply their own knowledge of the users and domain in order to determine if the requirements have been met.

1.1.2. Fit for Purpose

All software is designed to fulfill a purpose, but just accomplishing a task is not enough. In order for it to be "fit for purpose", the software must work for the people who will be using it, in the environment in which they will be using it. For example, a mobile application that allows people to deposit checks by taking a picture of the check may work great in the lab with specific lighting and backgrounds, but may fail when used in a user's home. In this case, the requirement may be met (it works functionally), but it is not "fit for purpose" because it is not usable in the target environment.

1.1.3. Risk

Because there is rarely enough time to perform all the testing possible, risk prioritization is used to limit the testing to what is needed to mitigate risk to an acceptable level. Determining what is acceptable may be a matter of opinion, which is why risk analysis requires cross-functional input to ensure each risk is being considered and rated accurately. With the above example of the check deposit, if the decision is that a low-lighting environment is highly unlikely, that would reduce the rating of that risk. On the other hand, if it is determined that this is highly likely to occur and that the user will be unable to deposit their check, the risk would be considered as very high and additional work would be required to adequately mitigate that risk. Risk is discussed further in Section 2.6.

1.1.4. Finding Defects

One of the purposes of testing is to find and fix defects before the software is released to the users. Defects, also called bugs, are flaws in the software that cause it to function incorrectly or cause the user to use it incorrectly. Clear requirements help in determining what is a defect and what is not. The less clear the requirements, the more discussion will be needed to determine if an anomaly is actually a defect or if it is just an undocumented feature of the software. Keeping the user's view in mind when testing the software helps the tester to better determine what a user would consider to be a defect. For example, an incorrect text prompt "enter suer name" is clearly a defect. What if the user name always has to be between 5-15 characters but the user is not told that? Is that a defect? Defect identification and proper recording is

an important task for a tester. Defects that are not recorded accurately are difficult, if not impossible, to fix.

1.2. A Brief History

Software testing has existed for as long as there has been software. The formality, emphasis, funding and respect for software testing has varied over the years, but it will always be needed. Good practices that were popular in the 1970's still have merit today, just as new practices developed since that time also have merit. It is important to remember that there is a wealth of knowledge in software testing. Environments, languages, devices and approaches may vary, but understanding the essentials of software testing will allow the tester to work in, and adapt to, any environment.

In software testing, there tends to be a differentiation between technical testers (i.e., test engineers) and non-technical testers (i.e., test analysts). Technical testers are expected to have the skills such as those needed to write test automation, conduct performance tests or participate in code/design reviews. Test analysts are generally expected to conduct the functional testing (i.e., does the software meet the requirements), as well as to consider usability (i.e., will the target user be able to use the software effectively, efficiently, and enjoy using it) and domain/environment attributes of the software. In some cases, test analysts are also expected to work with end-users for user acceptance testing (UAT) and to help validate that the software will work in the target environment for target users who are accomplishing the target tasks.

Like software development, software testing will continue to evolve. Mastering the essentials of software testing will help make a tester resilient and able to adapt to changes.

1.3. Structured Testing

Highly-structured testing, such as that required by some sequential lifecycle models (discussed in Section 2.3), generally has a higher level of documentation. Formal test strategies, well-defined test plans, explicit test cases, controlled test data and test environments, and a well-managed defect lifecycle are all artifacts of a highly-structured approach to testing.

While the documents may vary depending on the environment, the following are normally found in a structured testing environment:

- Test strategy – a test strategy is an organization-wide document that defines how testing will be conducted across all comparable types of projects in the organization.
- Test plan – a test plan is the implementation of the test strategy for a particular project and includes the approach to be used for testing, a definition of the scope of testing for the project, the testing schedule, the resource requirements, a description of tools and their usage, a definition of

environments and any other information required to describe the testing process, and stakeholder agreement for a project.

- Test conditions – a test condition is a capability or characteristic of the software that needs to be tested. This could be something functional, such as the ability to enter a user name; or something non-functional, such as the expected response time of the application under a defined load.
- Test case – a test case is the information required for a tester to test a test condition. This can include the pre-conditions of the system (e.g., user does not exist), the post-conditions after the test (e.g., the user has been created) and the inputs and actions required to accomplish the goal of the test.
- Defect reports – each defect should be captured in a report that is then processed through a workflow to record all the actions taken to resolve the issue. A defect report normally records information, such as the environment used, steps to reproduce, priority/severity, expected/actual results and other descriptive information.

More information about the documentation used in testing can be found in Section 2.5. Depending on the environment, more or less of these documents will be prepared and maintained as part of the testing process.

1.4. The Role of a Tester

The role of a "tester" can vary with different organizations and different lifecycle models. While software testing is a profession, others may periodically carry the title of a software tester. For example, in an Agile lifecycle model, everyone on the team has testing responsibilities and may be considered to be a tester. Business users may become testers during UAT. Software developers are testers when they are testing their own or another developer's code.

Regardless of the name of the role, testers are responsible for gathering information that can be used to assess the quality of the software. This information includes tests that have been run and have met their goals (passed), tests that have not met their goals (failed), defects found, risks mitigated, test coverage (in terms of tests executed vs. not executed, code covered vs. not covered, risks mitigated vs. not mitigated, or requirements tested vs. not tested) and other information needed by the stakeholders.

All testers need to be familiar with the essential areas of software testing. Specialization in these areas may require further study, but a general familiarity is necessary to understand what can and should be tested for any software product.

6. Cybersecurity Testing – 200 mins.

Keywords

cyberattack, cybersecurity, cybersecurity testing, fuzzing, cybersecurity controls, social engineering, system hardening, threat actors, threat intelligence, threat modeling, vulnerabilities

Learning Objectives for Cybersecurity Testing

6.1 Introduction

LO-6.1.a (K1) Recall the definition of cybersecurity
LO-6.1.b (K1) Recall the cybersecurity attributes of assets
LO-6.1.c (K1) Recall the function of cybersecurity controls

6.2 The Purpose of Cybersecurity Testing

LO-6.2.a (K2) Explain the necessity to both verify requirements and validate the system during cybersecurity testing

6.3 Cybersecurity Differences

LO-6.3.a (K2) Explain the unique differences of cybersecurity testing

6.4 Cybersecurity Testing Approaches

LO-6.4.a (K1) Recall when cybersecurity test planning should take place within the SDLC
LO-6.4.b (K2) Explain how cybersecurity testing contributes to risk management
LO-6.4.c (K2) Explain how testing can enhance the stages of a cybersecurity effort

6.5 Conducting Cybersecurity Testing

LO-6.5.a (K1) Recall why cybersecurity testing must be pre-authorized
LO-6.5.b (K2) Explain how white-box testing is used in cybersecurity testing
LO-6.5.c (K2) Explain how black-box testing is used in cybersecurity testing
LO-6.5.d (K1) Recall the controls that can be targeted by cybersecurity tests

6.6 The Environment of Constant Change

LO-6.6.a (K1) Recall ways to keep up to date with cybersecurity and cybersecurity testing practices

6.1. Introduction

Cybersecurity was commonly called information security or information assurance - the latter emphasizing the need to be confident or assured that digital resources are in fact secured. Cybersecurity refers to protecting computer systems, including their electronic data, software and hardware, from theft or damage as well as from disruption or misdirection of the services they provide.

Valued data or computerized processes are referred to as assets. Their security attributes typically are described as confidentiality (and its corollary, privacy), integrity, and accessibility: the so-called "CIA" triad. Confidential material should only be entrusted to designated parties for specific purposes, respecting the owner's privacy. The integrity of an asset is dependent on the assurance that it has not been tampered with via unauthorized access. When data or a service is needed, it needs to be accessible only to authorized users.

Cybersecurity addresses the need to protect the integrity and confidentiality of digital assets, as well as the accessibility of online processes. Computerized systems in the past suffered almost all of their failures due to defects in design or implementation. Today, however, so-called threat actors (malicious individuals or groups) conduct cyberattacks to compromise systems by deliberately misusing them. Critical infrastructures – including energy, communication, transportation, and finance – are now heavily computerized; defenses against online assaults are the essential responsibility of cybersecurity.

Protection of system resources and processes is the function of cybersecurity controls. These controls range from straightforward precautions such as account passwords to sophisticated automated detection and response mechanisms. Different cybersecurity controls may be designed to reduce the likelihood of successful attacks or to minimize the damage done if any attacks succeed. None of these protections are flawless, hence, anyone wishing to compromise computerized systems will be looking for weaknesses, known as vulnerabilities, in the controls.

Consider the situation in terms of the classic elements of criminality: motive, means, and opportunity. Vulnerabilities in cybersecurity controls are the opportunities - success depends on bypassing these controls. Malicious individuals and groups might be motivated by greed, revenge or ideology. But they will need knowledge, skills, and resources – the means of attacking – to take advantage of an opportunity and successfully compromise security. The strength of their motivation determines their persistence or willingness to risk detection.

6.2. The Purpose of Cybersecurity Testing

Cybersecurity testing probes systems to reveal potential failures in furnishing the desired level of security. As with any other testing, cybersecurity testing reveals the

presence of defects but cannot confirm their absence. It provides value both as verification ("Was it built right?") and as validation ("Was the right thing built?").

Where security requirements have been specified, testing can verify how adequately these requirements are satisfied. Threat modeling, based on threat intelligence, anticipates the nature of assaults that might be encountered in the operating environment of the system. System designers provide corresponding controls, and testing can verify the extent to which those controls provide the specified protection.

Specification of what a system should do is often less difficult than elaborating what the system should not do. Beyond defining and confirming controls, one must probe for unacceptable behaviors and validate proper functioning in an operational environment. To be realistic and meaningful, these tests need to model the projected capabilities of realistic adversaries, as identified through threat modeling.

6.3. Cybersecurity Differences

- **It must be responsive to a rapidly changing environment** – Vulnerabilities in complex interrelated systems are usually subtle and may lie dormant for years before suddenly being exposed. Systems also evolve by the modification or addition of features and by new interactions with other hardware and software, any of which can introduce new vulnerabilities.
- **The adequacy of established controls is under constant active probing** – Threat actors of widely varying motivations, capabilities, and resources fill the arena. When they can expose the weakness of a control they will promptly seek to exploit it and, in some cases, expose it to other threat actors. The ability and resources possessed by threat actors will only increase, especially as state-sponsored attacks increase.

- **Cybersecurity testing must address detection, response, and recovery after controls fail** – It needs to establish the resilience of the system when vulnerabilities are successfully exploited. It must evaluate how well risk mitigation supplements risk avoidance.
- **Human shortcomings are at least as prominent as technological challenges** – Exploits through various low-tech means (known as social engineering) are a leading cause of security failures. Countermeasures must address policies, procedures, and awareness campaigns.

6.4. Cybersecurity Testing Approaches

Of all the quality attributes, security might be the most difficult to "bolt on" or "patch in" after substantial software development has already been done. Cybersecurity test planning must contribute to the design and implementation of a system as early as possible. It should reinforce secure design and coding best practices, and include a risk analysis that is focused on cybersecurity.

Risk exists when a threat might take advantage of a vulnerability. Lowering the probability of that occurring is known as risk avoidance, and decreasing the consequences when it does occur is known as risk mitigation. To go one step further, risk mitigation must also be tested by probing the responses that occur when a control fails. By revealing otherwise undetected weaknesses, cybersecurity testing can direct efforts that support risk avoidance. The goal is to reduce overall risk exposure (and its uncertainty) to an acceptable level.

Potential security weaknesses in isolated system components may be investigated by static techniques. Manual or automated inspections, such as port scans and code scans, can provide early identification of insecure design patterns or coding practices. After a completed system is installed in its operational environment, additional dynamic security issues will arise (i.e., issues that can only be seen during execution) and must also be addressed.

Testing can enhance each stage of a cybersecurity effort as follows:

- Identify – Confirm identification, categorization and prioritization of the data and processes to be safeguarded (note that an adversary might value these assets differently and have different priorities)
- Protect – Analyze the protection of each asset category with corresponding controls that will stop or slow attempts to compromise the system
- Detect – Demonstrate the ability to detect intrusions and provide timely actionable analysis
- Respond – Exercise the responses and determine how well affected parties are notified, damage is minimized, and further access is shut off
- Recover – Demonstrate the speed and extent of recovering operational status and restoring data and confidence

6.5. Conducting Cybersecurity Testing

A testing project should establish terms of engagement that define the nature and scope of activities. If a live system is to be tested, it should specify how much notice will be given and to whom. All cybersecurity testing must be pre-authorized in writing by appropriate management to avoid testers being mistaken for malicious insiders (which can carry criminal penalties).

Cybersecurity testing covers the full software development lifecycle, so it will parallel and reinforce secure software engineering practices. It may be performed or be a part of reviews and audits at key specification, design, implementation, and integration stages. It will evaluate the performance of threat modeling and static code analysis.

Cybersecurity testing, like other types of testing, can and should be conducted in two different modes: “black-box” - which attends to external system behaviors; and “white-box” (or clear-box) – which utilizes structural knowledge of the as-built system. In black-box testing, an “outsider” will gather information via system reconnaissance. They will use that information to craft attacks that can be mimicked in penetration

testing to see what they can attack. In white-box testing, the “insider” threat is modeled by treating the system’s internals as visible and thus uniquely exploitable.

A wide range of testing approaches are available. One automated black-box technique involves fuzzing - inserting random variations of expected input values to detect sensitivities that might be exploited. Another mode of testing is derived from the practice of military exercises. A designated “red team” of attackers (composed of internal or external experts) is pitted against a “blue team” of defenders (drawn from internal operations personnel). The exercise may be performed on a facsimile of the SUT, or may be a “tabletop” exercise without a real system.

Targeted tests should probe the adequacy of specific controls. These may include:

- Policy and procedure enforcement
- System hardening (reducing the attack surface by eliminating as many security risks as possible)
- Access control mechanisms for authentication (who you are), authorization (what you can do), and accountability (what did you do)
- Input validation
- Encryption
- Error and exception handling
- Intrusion detection
- Malware scanning
- Resistance to social engineering

6.6. The Environment of Constant Change

Best cybersecurity and cybersecurity testing practices may be found in various technical guidance documents and consensus standards, as well as through professional organizations and educational institutions. “Best practices” may be “required practices”, if mandated by government laws. Regulatory requirements may also apply to specific financial, medical, transportation, energy or other sectors. In such settings, the adequacy of testing would be judged by the extent to which it confirmed compliance with applicable mandatory requirements.

New technologies and new applications of existing technologies will continually introduce new security concerns. Seemingly stable, long-time tools and applications often reveal long-time vulnerabilities under the scrutiny of determined adversaries. External laws and regulations, as well as user expectations, change over time too. Keeping up-to-date with changes in these requirements and best practices, as well as technological advances, should be accomplished through ongoing professional development - reading, conversing and attending events with fellow practitioners.

10. References

10.1. ISO/IEC/IEEE Standards

- ISO/IEC/IEEE 12207:2017
- ISO/IEC/IEEE 15288

10.2. Trademarks

The following registered trademarks and service marks are used in this document:

- AT*SQA® is a registered trademark of the Association for Testing and Software Quality Assurance

10.3. Books

[Anderson00]: Anderson, L.W. and Krathwohl, D.R. (2000) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon: Boston MA, ISBN-10: 080131903X

[Firtman]: Maximiliano Firtman, "Programming the Mobile Web", O'Reilly Media; Second Edition (April 8, 2013), ISBN-10: 1449334970

[PMBOK] Project Management Institute, "A Guide to the Project Management Body of Knowledge (PMBOK Guide) – Sixth Edition, 2017, ISBN-10: 9781628251845

10.4. Other References

The following references point to information available on the Internet. Even though these references were checked at the time of publication of this syllabus, AT*SQA cannot be held responsible if the references are not available anymore. AT*SQA is not endorsing any of these sites or their products. The references are provided as a source of information only.

https://techcrunch.com/2013/03/25/ip-oh-my-gosh-all-that-money-just-disappeared/

https://www.reuters.com/article/us-facebook-settlement/facebook-settles-lawsuit-over-2012-ipo-for-35-million-idUSKCN1GA2JR

[NASDAQ] https://www.sec.gov/news/press-release/2013-2013-95htm

National Institute of Standards and Technology. Framework for Improving Critical Infrastructure Cybersecurity. Version 1.1. 2018. https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf

National Institute of Standards and Technology. Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy. Revision 2. 2018. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf

[WCAG] https://www.w3.org/WAI/policies/