

ASTQB Certified Tester Syllabus

Soft Skills for Testers



American Software Testing Qualifications Board

Copyright Notice

Copyright Notice © American Software Testing Qualifications Board (hereinafter called ASTQB®)
ASTQB® is a registered trademark of the American Software Testing Qualifications Board.
All rights reserved. The authors hereby transfer the copyright to the ASTQB®. The authors (as current copyright holders) and ASTQB® (as the future copyright holder) have agreed to the following conditions of use:

Extracts, for non-commercial use, from this document may be copied if the source is acknowledged. Any Accredited Training Provider may use this syllabus as the basis for a training course if the authors and the ASTQB® are acknowledged as the source and copyright owners of the syllabus and provided that any advertisement of such a training course may mention the syllabus only after official Accreditation of the training materials has been received from an ASTQB®-recognized Member Board.

Any individual or group of individuals may use this syllabus as the basis for articles and books, if the authors and the ASTQB® are acknowledged as the source and copyright owners of the syllabus.

Any other use of this syllabus is prohibited without first obtaining the approval in writing of the ASTQB®.

Any ASTQB®-recognized Member Board may translate this syllabus provided they reproduce the Copyright Notice mentioned above in the translated version of the syllabus.

Revision History

Version	Date	Remarks
V0.1	2025/09/30	Draft for review
V1.0	2025/10/1	Reviewed comments incorporated. Release version.
V1.0	2025/11/1	Review comments incorporated from Beta review.
V1.0	2026/1/23	Final review for minor edits

Acknowledgements

This document was formally released by the ASTQB® on October 1, 2025.

It was produced by a team from the American Software Testing Qualifications Board: Judy McKay

The following persons participated in the reviewing, commenting and balloting of this syllabus: Earl Burba, Marie Potthoff, Randy Rice

Table of Contents

Copyright Notice	2
Revision History	2
Acknowledgements	2
Table of Contents	3
0 Introduction	5
0.1 Purpose of this Syllabus	5
0.2 The Soft Skills for Testers	6
0.3 Learning Objectives and Cognitive Level of Knowledge	6
0.4 The CT-SS Certificate Exam	7
0.5 Accreditation	7
0.6 Handling of Standards	8
0.7 Level of Detail	8
0.8 How this Syllabus is Organized	9
1 Professional Communication – 5 hrs 30 minutes	10
1.1 Staying Professional	12
1.1.1 Safest Approach	13
1.1.2 Keep it Professional	14
1.1.3 Check Before You Send	15
1.2 Showing Empathy	15
1.2.1 Read the Room	16
1.2.2 Working with Business Analysts (BAs)	17
1.2.3 Project Managers	19
1.2.4 Developers	21
1.2.5 Other Testers	22
1.2.6 Your Manager and Lead	24
1.2.7 Cross-Culture Awareness	25
1.3 Communication Modes	26
1.3.1 Video Conferencing	26
1.3.2 Chat	27
1.3.3 Email	28
1.3.4 In-Person	30
1.4 Deflecting Anger and Frustration	32
1.4.1 Anger Targeting	32
1.4.2 Know Yourself	34
2 Business Writing and Communication – 3 hrs 45 mins	35
2.1 General Rules for Business Writing	36
2.1.1 Be Professional	37
2.1.2 Test Strategies and Test Plans	39
2.2 Business Writing in Testing	42
2.2.1 Test Reports	42

2.2.2	Test Cases	44
2.2.3	Defect Reports	47
2.3	Miscellaneous Meetings	49
2.3.1	Defect Triage Sessions	49
2.3.2	Daily Standups	50
2.3.3	Project Status Meetings	51
2.3.4	Retrospectives	52
3	Promoting the Value of Testing – 4 hrs, 45 mins	54
3.1	Level of Understanding	55
3.1.1	Risk	56
3.1.2	Testing is Hard	57
3.1.3	Testing the Quality Characteristics	59
3.1.4	Cost of Quality	60
3.1.5	Schedule Over Quality	62
3.2	Opportunities for Testing	63
3.2.1	Testing Gaps	63
3.2.2	Right Testing, Wrong People	65
3.2.3	Busy SMEs	66
3.2.4	Production Escapes	67
3.2.5	Inefficient Product Lifecycle	68
3.2.6	Test Automation Use	69
3.3	Empathy with the User	71
3.3.1	Usability	71
3.3.2	Accessibility	72
4	Professional Development– 2 hrs 45 minutes	74
4.1	Rising in the Organization	75
4.1.1	Preparing for Promotion to a New Role	76
4.1.2	Receiving a Promotion to a New Role	77
4.2	Interviewing for a New Position	78
4.2.1	Preparation	78
4.2.2	Promoting your Skills and Capabilities	79
4.2.3	Asking Questions	78
4.2.4	Ensuring a Good Fit	81
4.2.6	A Career in Testing	83
5	References	84
5.1	Standards	87
6	Trademarks	88
7	Appendix A – Learning Objectives/Cognitive Level of Knowledge	89
	Level 1: Remember (K1)	89
	Level 2: Understand (K2)	90
	Level 3: Apply (K3)	91

0. Introduction to this Syllabus

0.1 Purpose of this Syllabus

This syllabus forms the basis for the American Software Testing Qualification for the Soft Skills for Testers. The ASTQB® provides this syllabus as follows:

1. To member boards, to translate into their local language and to accredit training providers. Member boards may adapt the syllabus to their particular language needs and modify the references to adapt to their local publications.
2. To certification bodies, to derive examination questions in their local language adapted to the learning objectives for this syllabus.
3. To training providers, to produce training materials and determine appropriate teaching methods.
4. To certification candidates, to prepare for the certification exam (either as part of a training course or independently).
5. To the international software and systems engineering community, to advance the software and systems testing profession, and as a basis for books and articles.

0.2 The Soft Skills for Testers

The Soft Skills for Testers qualification is aimed at anyone involved in software testing. This includes people in roles such as testers, test analysts, test engineers, test consultants, test managers, user acceptance testers, and software developers.

Please note that any references to tools is only intended to provide examples of tools that were popular when this syllabus was written. This is not intended to be an endorsement of these particular tools.

0.3 Learning Objectives and Cognitive Level of Knowledge

Learning Objectives support the business outcomes and are used to create the CT-SS exams.

In general, all contents of this syllabus are examinable, except for the Introduction and Appendices. The exam questions will confirm knowledge of keywords at K2 level (see below) or learning objectives at the respective level of knowledge.

The specific learning objectives and their levels of knowledge are shown at the beginning of each chapter, and classified as follows:

- K1: Remember
- K2: Understand
- K3: Apply

Further details and examples of learning objectives are given in Appendix A.

For all terms listed as keywords just below chapter headings, the correct name shall be remembered (K1), and the usage of the term in the syllabus shall be understood at the (K2) level.

0.4 The CT-SS Certificate Exam

The CT-SS Certificate exam will be based on this syllabus. Answers to exam questions may require the use of material based on more than one section of this syllabus. All sections of the syllabus are examinable, except for the Introduction and Appendices. Standards and books are included as references, but their content is not examinable, beyond what is summarized in the syllabus itself from such standards and books.

The entry criterion for taking the CT-SS exam is that candidates are interested in software testing. However, it is strongly recommended that candidates also:

- Have at least a minimal background in either software development or software testing, such as six months of experience as a system or user acceptance tester or as a software developer
 - Take a course that has been accredited to ISTQB® standards (by one of the ISTQB-recognized member boards).
-

0.5 Accreditation

Training providers should obtain accreditation guidelines from the ASTQB or body that performs the accreditation. An accredited course is recognized as conforming to this syllabus and is allowed to have an ASTQB® exam as part of the course.

The accreditation guidelines for this syllabus follow the general Accreditation Guidelines published by the Processes Management and Compliance Working Group of the ISTQB.

0.6 Handling of Standards

International standardization organizations like IEEE and ISO have issued standards associated with quality characteristics and software testing. Such standards are referenced in this syllabus. The purpose of these references is to provide a framework (as in the references to ISO 25010 regarding quality characteristics) or to provide a source of additional information if desired by the reader. Please note that ASTQB® syllabi use the standard documents as reference. Standards documents are not intended for examination.

0.7 Level of Detail

The level of detail in this syllabus allows internationally consistent courses and exams. To achieve this goal, the syllabus consists of:

- General instructional objectives describing the intention of the CT-SS Level
- A list of terms that students must be able to understand
- Learning objectives for each knowledge area, describing the cognitive learning outcome to be achieved
- A description of the key concepts, including references to sources such as accepted literature or standard

The syllabus content does not describe the entire knowledge area of software testing; it reflects the level of detail to be covered in Soft Skills for Testers training courses.

0.8 How this Syllabus is Organized

There are four chapters with examinable content. The top-level heading for each chapter specifies the time for the chapter; timing is not provided below the chapter level. For accredited training courses, the syllabus requires a minimum of 16.75 hours of instruction, distributed across the four chapters as follows:

- Chapter 1: Professional Communication – 5 hrs 30 mins
 - The tester learns how to communicate effectively and professionally, even in difficult situations
 - The tester understands communication via commonly used tools
- Chapter 2: Business Writing and Communication – 3 hrs 45 mins
 - The tester learns how to incorporate good business writing practices into test documentation
 - The tester learns how to communicate effectively in various meetings
- Chapter 3: Promoting the Value of Testing – 4 hrs 45 mins
 - The tester learns how to recognize opportunities for testing
 - The tester learns how to effectively promote testing to a project team
- Chapter 4: Professional Development – 2 hrs 45 mins
 - The tester learns how to apply for and excel in a promotion
 - The tester learns how to interview for a new role

1. Professional Communication – 5 hrs 30 minutes

Keywords

acceptance criteria, Service Level Agreement (SLA), Subject Matter Expert (SME), User Acceptance Test (UAT)

Learning Objectives for Chapter 1:

1.1 Staying Professional

- SS-1.1.a (K2) Summarize the characteristics of effective communication
- SS-1.1.b (K2) Differentiate between professional and unprofessional communication
- SS-1.1.c (K2) Explain why it is important to pause before sending an email

1.2 Showing Empathy

- SS-1.2.a (K2) Explain how viewpoint affects the receipt of a message
- SS-1.2.b (K2) Describe the value provided by Business Analysts
- SS-1.2.c (K2) Explain how a tester can facilitate Business Analyst review of test cases
- SS-1.2.d (K2) Give examples of how PM focus differs from testing
- SS-1.2.e (K2) Explain how schedule-impacting events can be tracked and presented

- SS-1.2.f (K2) Summarize the considerations when working with developers on defect resolution
- SS-1.2.g (K2) Explain how testers can improve efficiency and teamwork with other testers
- SS-1.2.h (K2) Explain how problems should be presented to management

1.3 Communication Modes

- SS-1.3.a (K2) Summarize potential issues with video conferencing
- SS-1.3.b (K2) Summarize potential issues with using chat tools
- SS-1.3.c (K2) Summarize potential issues with email
- SS-1.3.d (K2) Explain the benefits of face-to-face communication
- SS-1.3.e (K3) Select and use the best environment to convey a particular type of message

1.4 Deflecting Anger and Frustration

- SS-1.4.a (K2) Summarize causes of frustration in a work environment
- SS-1.4.b (K2) Explain good techniques to defuse anger in a business situation
- SS-1.4.c (K2) Explain how fairness can be established

Professional Communication

Testers spend a good part of their time communicating. Communication is conducted through various means, including email, text, meetings, and face-to-face interactions. Each communication is an opportunity to promote the value of testing, but also a potential pitfall for testing to be regarded as unnecessary, frivolous, or even lacking professionalism.

Effective communication must consider the media being used and the intended recipient, with the goal being for the recipient to receive what the sender was intending. This can be surprisingly difficult in some situations and with some people, but testers must always be aware that their communication and its effect matter.

1.1 Staying Professional

Testers work in a high-pressure environment, conveying information to various stakeholders who may not want to hear what must be said. While it is generally easier to convey good news than bad news, it is essential to consistently communicate professionally.

1.1.1 Safest Approach

A professional approach is always best, even when communicating with a close colleague. One thing to keep in mind is that the intended recipient may not be the only recipient. It is easy for email to be forwarded, for defect reports to be shared, and for meetings to be recorded. In these situations, the context can be lost. Saying something in an email, such as “The code is broken again”, might be understood by the developer who is familiar with the issues but might not be appreciated by their manager who is working to improve overall quality, or the project manager who is worried about the schedule.

Effective communication gets the message to the recipient accurately, quickly, and concisely. Everyone is busy. With defect reports, no one appreciates having to read through the 25 steps performed to get to the one that failed. Getting to the point quickly but with an adequate amount of information is not easy. Concise is good, but not if it has created a defect report that requires the developer to request additional information. Getting the right information communicated clearly with the right level of detail on the first attempt is a learned skill.

Another consideration in safe and effective communication is to think about how the message will be received. Will the recipient be happy? Informed? Curious? Annoyed? Angry? There are two keys to help guide communication so it will be received as intended: ensure the tone is objective by reporting observations, not opinions; and concentrate on informing rather than guiding. By supplying objective information, the potential for slanting the information, particularly toward the negative, is avoided. By concentrating on informing, it is easier to review the message and ensure that all the necessary information has been included. This will save frustration and wasted time required for additional communication and clarification.

1.1.2 Keep it Professional

A common mistake in business communication occurs when the sender assumes the recipient is a buddy. While working closely with a project team is normal, and while in-person communication may be informal, all written communication should be professional. Written communication and recorded communication can have a long life and can easily be reviewed by someone other than the intended recipient. When this occurs, misinterpretation is easy, and something that was intended as a joke can be taken as an insult.

Professional does not mean sterile. Communication can still be informal, but it must be professional and not open to interpretation. This is one of the reasons that objective statements are preferred. A developer would rather see:

“I expected the software to do xxx based on the requirements, but it did xxx. Is that correct?”

than

“The software isn’t working right, it was supposed to do xxx, but it did xxx”.

And, even though it might be an inside joke, saying the following is just not acceptable:

“The software doesn’t work, and apparently neither does the developer!”

While that may seem extreme, any experienced test manager will have seen something similar and will have spent significant time doing damage control. Even though the temptation may be there, sticking to professional communication will always be the safer alternative.

1.1.3 Check Before You Send

Testing can be a frustrating job, and sometimes emotions cloud judgment. Sometimes the tester is not even aware that their emotions are showing through. This is when it pays to pause before hitting the send key. It is easy to dash off a quick and less-than-professional message, or defect report, or weekly status, but the best approach is to review, re-write and maybe not send it at all. It is very easy to include a negative comment in a status report that is then incorporated into a larger status report – and this could go undetected by a busy manager who is compiling the report. Similarly, a subjective comment in a defect report can end up being reviewed by the entire triage team. Be careful and review before sending any type of communication as it will reduce the time needed for apologizing later.

1.2 Showing Empathy

Testers have a different viewpoint from most of the other project team members. Testers are not paid to be overly optimistic or unrealistic about schedules or miracles. They deal with the reality of software that does not work correctly, environments that are broken, and schedules that are impossible. The net result of this is an outlook that can be considered to be negative, even though it is only realistic. This tester's outlook can bleed into communication and can result in the tester being classified as negative, uncooperative, not a team player, or just resistant. This section looks at how to communicate effectively with the rest of the project team.

1.2.1 Read the Room

“Read the room” is a common phrase used in effective communication. This may apply literally when speaking in front of a set of people, but it also applies to individual communication. A common mistake of testers is to forget that other people do not come from the same viewpoint. That does not mean the testing viewpoint is wrong, just that it may be unexpected.

Testers are often seen as having a negative view of a project, of the software, and of other members of the team. It is not that this view is necessarily unfounded, but rather that a tester and test manager need to understand that others may have a completely different viewpoint. This viewpoint may arise from unrealistic optimism or from operating from a different set of information. For example, when the development team is reporting that a project is on schedule, an announcement from the test team that no progress has been made for the last six weeks will not be expected and certainly will not be seen in a positive light.

In addition to the context of the message, it is also important to remember that people may react differently depending on who is around them. An individual developer may be honest about quality issues when talking just to a tester but may have a completely different voice when their manager is present. An entire project team may agree that a schedule is unrealistic but may be reluctant to convey that message to the project manager. This situational context may result in a tester presenting a message that should have good support, only to sound like the lone voice of dissent with a negative message.

When presenting information, particularly information that will likely be interpreted as negative, it is important to present the background information as well. If there has been a six-week blockage in testing, presenting the reasons for the blockage is vitally important

if the message is to be received as intended. Without an adequate background, it is easy for denial or message dismissal to occur.

It is not unusual for the tester to find themselves in the role of trainer, familiarizing the rest of the team with how testing works, so they can understand the testing viewpoint. Each team member will come with their own view, so understanding that first will be helpful to build both a good relationship and a pathway for effective communication. Testers who are well-rounded with development, BA, and project management experience can help others who have not had as wide an exposure to the other viewpoints.

1.2.2 Working with Business Analysts (BAs)

BAs have a difficult role. They are expected to take the nebulous explanations from the business Subject Matter Experts (SMEs) and translate those into technical requirements that can be implemented. In most roles, they are pelted with questions from users, developers, data analysts, project managers, and, of course, testers. This can make them less than patient when answering questions, and if they do not fully understand the role of testers, they can be insulted that the requirements are being “questioned”.

Any experienced tester values a good BA (BATimes, 2022). They have knowledge that testers do not have – particularly regarding what the

users need, how they will use the software, and what their expectations and concerns are for the final product. This knowledge is precious when used to create valid test cases, to focus and prioritize the testing, and to ensure a successful User Acceptance Test (UAT). BAs can be a treasure trove of information, but they are busy, and they need to understand how their information can help testing, which in turn helps the project.

The acceptance criteria define the requirements. Some methodologies require the acceptance criteria to be expressly documented, but others allow them to be implied in the written requirements. Ideally, the acceptance criteria are captured and reviewed by the team. If the BA has not done this and does not intend to, it may be helpful for the tester to capture these. This is often done in the form of test conditions which are later used to build test cases. (Conciu, 2023)

When working with a BA to frame out the acceptance criteria, it is good to approach this as a task that is needed for testing, not something the tester has to do that the BA should have done. If the BA feels the tester is encroaching on their area, they are likely to be resistant. By instead making them understand how the information is needed for testing, the BA can be turned into an ally who is looking to solve the same problems. It can also help to work out communication preferences with them. For example, if the tester has a question, which approach is preferred? Some examples include:

- Approach the BA with the question (email, phone call, in-person, etc.)
- Approach the users or SMEs directly with the question
- Create a question register with a Service Level Agreement (SLA) for responses
- Any other approach that is preferred

By offering multiple paths of communication, the BA may feel less bombarded. Practically, though, there is an expectation of response from the BA, or the tester may become blocked waiting on answers, particularly if the requirements are poor.

Ideally, the BA could review all the test cases for validity before they are executed, but this is very time-consuming, and it is the rare BA who will agree to this. It is usually better to have a review meeting where the tester walks through a set of test cases. In this way, the review gets done, but it also allows room for additional questions and clarifications.

1.2.3 Project Managers

Project managers have a different focus from the other members of the project team. They are concerned primarily with getting the work done within budget and on schedule. This focus can make them resistant to information that indicates a problem with either the cost or the timeline. Quality issues usually negatively affect both these aspects of the project.

Many project schedules are built with the assumption that there will be no quality issues. The developers will develop the product, the testers will do something, the users will accept the product, and everyone will be happy. Unfortunately, while the testers are “doing something” there is often a discovery of quality issues that are schedule-impacting. Many methodologies have been proposed and implemented to help find quality issues sooner, but the reality is that quality issues still get through to testing and still impact schedules.

It is easy for the testers to be blamed when there are schedule impacts due to quality. It is important that the data is tracked and available to back up any pronouncements of quality issues. For example, if the

test environment has been delayed by five weeks, but the slip has actually occurred on a daily basis (i.e., it will be ready tomorrow), then the test team must be tracking each slip. It is easy to forget to do this, particularly when it does not appear to be a schedule-impacting issue in the near term, but every outage, every delay, must be tracked. This is sometimes done by recording a defect for each delay which can be prioritized by the team. This information must also be recorded in each weekly status report, so the information is not a surprise when the environment delays have added up and caused a five-week slip.

In addition to having the objective information to back up any schedule issues due to quality, it is also important to present any workarounds. It is a fine line with workarounds to show a willingness to deal with the problem, but not be so accommodating that the issues are minimized or ignored. Using a measurement of effective progress is a good approach when there is a blocking issue. For example, if the test environment is not available for some reason, there are still tasks the test team can do (e.g., preparing data, writing test cases, following up on defects), but this is probably not allowing for 100% progress toward the deadline. The team is not idle, but might only be 45% effective vs what they could do if the environment were ready. Testers timeshare their tasks during testing, but this may not be visible to the project manager.

The best way to deal with a project manager is to be factual. The objective information must be available to show why there are issues. If the schedule is slipping due to quality issues with the incoming code, show the defect trends, show the regression rates, and compare those to the expected trends. This will help the project manager understand why this is impacting the schedule.

This section has been all about reporting bad news to the project manager. That is because reporting is expected and good news is easy.

One problem that can occur with good news is when it is presented with no caveats. For example, the team might be ahead of schedule, but that might be because the higher-risk software has not yet been delivered. Even with good news, it is important that objective information be supplied to show the trends. A project might be 50% complete and ahead of schedule, but might have only delivered 25% of the high-risk items. The project manager needs to understand the residual risk to do their job, and it is the test team's job to supply that information.

1.2.4 Developers

Developers usually have a different focus than testers. Developers may say that they focus on creation, whereas testers focus on criticizing. Clearly that is a negative statement, but it is something testers need to keep in mind and to understand. Developers may be driven by unrealistic deadlines where they are being assessed based on code completed. Defects are often seen as impacting forward progress.

Developer schedules should allow time for defect research and fixing, but they rarely do. Testers can work effectively with this situation, understanding that the developers have only a restricted amount of time. The following considerations should be taken:

- Schedule constraints
- Agreed SLAs for defect fixing to set the expectation
- Communication preferences (e.g., email, conference call, in-person)
- Additional data needed (e.g., data, screenshots)
- Use of the backlog to store defect reports
- Prioritization and scheduling

The goal of a tester with a defect report should be to make that defect easy to fix. This means supplying the developer with all the information they need when the report is written and answering any questions

quickly. Early triage saves time for everyone. It is also helpful to discuss the priority with the developer and the expectations for when it will be fixed. If the development team puts all defect reports into the backlog, it is likely they will receive a lower priority than work already scheduled.

Sometimes a defect will be blocking testing, so it is important that the developers understand that and can prioritize accordingly. Some teams use an “urgency” setting to indicate defects that must be fixed soon. Understanding the service level agreements will help set the expectations for defect fix turnaround and will help reduce frustration.

There will be situations where the developers are either unable or unwilling to communicate. When this occurs, escalation to the test manager and the project manager may be appropriate. Ideally, though, communication issues are solved one-to-one rather than involving management.

1.2.5 Other Testers

Some testers become territorial, meaning that they do not want any other testers working in their assigned area. While this shows great responsibility and ownership, it can significantly constrain the flow of work. Testers need to work together, sharing what they will be doing and making sure they are not duplicating work. Daily standups can be effective for ensuring everyone knows who is doing what.

As a tester, there is a responsibility to know what is currently being tested, what defects have already been reported, and who is assigned to which areas. Redundant testing, duplicate defects, and data destruction will impact the efficiency of work within the test team. The test manager is responsible for making assignments and clarifying with the team which areas they will cover. Each tester is responsible for ensuring

that every defect report they write is not a duplicate. Test data can be difficult to create, and the destruction of data created by someone else can cause resentment. The tester should always ensure that the test data they are using is theirs to use.

Testing is rarely confined to just one area: areas overlap, data is transferred between modules, and interfacing systems may interact with multiple areas. Good and clear communication within the team is vital. Using an online conferencing application, such as Microsoft Teams, allows the team to quickly post what they are planning to do to ensure they are not overlapping or interfering with other planned testing. It is also a good way to check if a defect has already been seen and reported.

Another important aspect of being a tester is understanding that mistakes will be made. Defects will be missed. Tests will not be executable as they were designed. This is expected. Working together as a test team will help everyone fill in the gaps. If someone catches a defect someone else missed, the correct comment is “thank you”, not “why were you testing in my area?”. Working as a team and understanding that mistakes will happen helps the team work cohesively and effectively. Recognizing each other’s skills and respecting the work everyone is doing will go a long way toward building a good team spirit.

1.2.6 Your Manager and Lead

The relationship between a tester and their manager/lead is critical for job happiness. It is up to the manager to create a supportive environment where a tester can be successful. It is up to the tester to be honest, to do the best work possible, and to grow in their role.

Anytime the tester brings a problem to their manager, it is a good idea to also bring a possible solution. Even if it is not a perfect solution, bringing a suggested solution shows that thought has been put into the problem and the tester is willing to work to solve the problem. A busy manager is much more likely to listen to a problem when the tester has obviously put some thought into it. This is particularly true about interpersonal problems.

No one likes a complainer, particularly someone who is busy. It is always best to assume that the test manager is busy, so explaining a problem with an interest in resolution will help focus attention. Consider the following presentations of problems:

The developer is ignoring me. Every time I bring him a problem, he says he's too busy. I think he hates me.

I think the developer is too busy. When I bring him a problem, he says he doesn't have time to look at it. Could we start having a short triage meeting each day to go through the high-priority problems?

The first presentation is a complaint. The second is a possible solution to a problem.

Managers can be sensitive to wording, particularly when they are busy. If a tester storms into their office saying the work environment is intolerable and the developers' behavior is unacceptable, the manager must first sort through the emotion to understand the real problem. By wording a problem in professional and objective terms, the tester will be able to bypass the manager's natural inclination to

avoid confrontation, or at least the large internal sigh they are going to give when the complainer arrives at their door. A tester needs to be productive, not destructive.

Professionalism is always respected, even when the tester is in the wrong. It can be hard to be professional, particularly when under pressure and frustrated. Professionalism grows careers. Hot heads rarely progress up the chain, even though they may be sincere. Business people have long memories and they will remember emotional outbursts, even if the outburst was justified at the time. There is always a professional way to deal with a situation.

1.2.7 Cross-Culture Awareness

This section has discussed the various communication issues between roles in a project team, but these may be further complicated when teams are cross-cultural. People from different cultures may have different expectations for the format and content of communication. An email that may seem straightforward and factual to one culture may be considered confronting and impolite to another. Being aware of these differences is critical to ensuring a message is received as intended. With any communication, cross-cultural awareness must be applied to ensure the message is understood and taken on board as intended.

It is important to remember that both the media and the content matter in communication. The following list shows some of the issues to consider:

- A phone call may be seen as more confronting than an email because it is more disruptive
- An email could be misunderstood because of language issues

- It may be hard to get negative feedback as a manager because some cultures would consider that to be disrespectful
- A request for overtime work may be taken as a requirement, an inquiry, or a suggestion depending on the team member's background

These are just a few examples of how cultural and background differences can affect the interpretation of communication. The best approach is to follow up to be sure the message was understood as intended and to keep an open dialog to ensure that any potential bad feeling is immediately addressed and resolved.

1.3 Communication Modes

As remote working becomes more popular, communication is no longer primarily face-to-face. This has introduced some new challenges to effective communication.

1.3.1 Video Conferencing

As with all forms of communication, professionalism is a requirement. Video conferencing, particularly with cameras on (Integration, 2025), gives an opportunity for a more personal connection than just email. While this medium is often used for meetings, it is also great for a quick catch-up with another team member or a developer. Screen sharing means that a developer can see exactly what the tester is seeing, which can be a much more efficient means of communication than multiple emails and screenshots.

When a team is mostly working remotely, a video conference can be a great way to stay connected, so some interpersonal chatter is usually helpful. Before getting too comfortable, though, it is important to remember that most video conferencing tools can record and transcribe what is said. This is sometimes saved for reference, so it is important to remember to keep it professional even when talking with friends.

1.3.2 Chat

Chat tools are commonly used for teams to communicate. This can be individual chats between two people, or group chats that include a set of folks with common interests, such as a project team. Chats tend to be the least formal communication method and can work well to keep everyone informed, both about project information and personal information such as birthdays. Organizations vary with the type of information they want conveyed in chat groups, so it is important to understand what is acceptable. For example, some will have chats that are only for project information and others that are for more personal information. It is important to understand the expected usage and comply with it.

Chat groups can contain a varied set of participants, such as project sponsors and management as well as developers, testers, BAs, etc. Some information may not be suitable to share with project sponsors or may not warrant management escalation, so it is critical to know who is in a chat group before information is posted. Most teams will have multiple groups with at least one being for management and another for only the technical team. This helps to quell the noise that is not interesting to the management team but still allows the technical team to communicate effectively.

Chat usually provides an easy way to check understanding and clarify any issues quickly. But, as with most communication modes, the communication can be copied and sent to other, potentially unanticipated, recipients, so professionalism is needed. Do not assume the chat content is private.

For providing quick information, asking/answering questions, and getting opinions, chat is great. For communicating issues that will need follow-up, it is not as effective as it can be difficult to scroll back to find a particular communication thread. In this case, email is more effective and provides a better record of the conversation. While a chat discussion with a developer about a possible defect can save time and provide clarity, it is not a substitute for a formal defect report. So, while chat is a convenient tool, it should be used for discussion but not as a replacement for more formal documentation that will be needed for further processing or reference.

1.3.3 Email

It is important to remember to think before hitting “send”. Will the message being sent be the one that is received? If it were re-routed to the rest of the team, would that be a problem? Email should be considered a permanent record. Most testers, being natural packrats, keep all emails both sent and received. With this in mind, think carefully before you send. How will the recipient take this message? What about the unintended but possible recipients? Did you really mean to Reply-All? If there are any doubts, the message should not be sent. It is much easier to stop and rewrite a message than to recall it after it has been sent.

Some people put a two-minute delay on their email to avoid the wave of regret that can wash over just after the send button has been pressed (Review, 2015). This is a good way to provide a window for recall, just in case there are second thoughts. If the message is potentially controversial, it is a good idea to get a second opinion, preferably from someone not immediately involved in the situation. This will help to remove the emotion and ensure objectivity.

Another method for pausing is to write the email but send it to yourself. This provides a chance to review the message as the recipient would see it. If it is all good, it can be sent. If it needs some editing, it is easier to do it before it is sent to the real recipient. Because testers work in an environment that is full of time pressures, stress, and frustration, it is very easy to send communication in the heat of the moment rather than after a considered pause. Building in that pause will help to ensure there is a chance for objectivity to intervene. The best rule is that if the message is controversial, do not send it without further consideration

As with all communication forms, it is important to remember that the recipient may not have the full context. It is easy to dash off a quick email, but leaving out important details or background information may result in a misinterpretation or even having the message ignored. Busy team members will usually glance through emails, checking to see if any of them need immediate action. It is important to ensure that the gist of the message is easy to discern, and the priority of action/response is clear.

Some people will respond quickly without reading the entire email. If two questions have been included in the email, this may result in only one being answered. Knowing the recipient helps to tailor the message so it is efficient for both the sender and the receiver. When multiple items need attention in an email, consider numbering them to call attention to the entire list.

It is usually a personal preference whether the response will contain the original message. In general, this is a more convenient and efficient way to correspond because the message is right there without having to search for what was being answered. While this may take up a little more space on the mail server, the time savings for both sender and receiver usually justify the extra space.

As with other forms of written communication, presenting the message clearly and concisely is just polite. It saves everyone time, it helps to eliminate confusion and misunderstanding, and it focuses the receiver's attention on the message.

1.3.4 In-Person

Although it is becoming less common, in-person communication is still a core part of being a human. Some conversations are better had in person so the reaction can be judged, the message can be adjusted, and a more casual tone can be used. In-person communication is particularly good for casual chats and for difficult conversations.

When in-person communication is not an option, video conferencing with a camera on is the next best alternative. Being able to see the recipient and to be able to judge their reaction to the message is invaluable and helps to reduce misinterpretation and offense.

In-person communication does not have to occur in the office environment. It is sometimes better to remove the distractions of the workplace and to have a chat outside the office. Going out for a coffee or a quick snack can help both parties focus on the conversation and reduce some of the formality of the office setting. For situations that

require more interaction, lunch or, if appropriate, after-work drinks are also options. Picking the right environment depends on the following:

- Relationship between the participants
- Nature of the message
- Time constraints
- Personal circumstances
- Location
- Accepted norms

It is important to allow the other person to propose another time/place or to refuse if that is appropriate. If a tester wants to build a relationship with a developer, but the developer refuses lunch or coffee, a meeting at the office during work hours might be more appropriate. Good flexibility is required to create the best environment for an in-person interaction, particularly if the topic to be discussed is controversial.

For test managers, keeping the team working together even when they are on different projects can be facilitated by team lunches and activities. This allows everyone to talk in an informal, non-work environment and will help to build a supportive and interactive environment for the team.

1.4 Deflecting Anger and Frustration

Unfortunately, tight schedules and project pressures sometimes result in anger boiling over during communication within the team and with other teams. This is often a manifestation of frustration and can be difficult to control. With any situation with frustration involved, it is always better to take some time, create some space, and exercise empathy for the other person or team. Anger will not resolve anything, but it is likely to generate bad feelings and resentment.

1.4.1 Anger Targeting

In addition to avoiding bringing anger to a situation, a tester may also be the target of anger from other team members. This again is usually rooted in frustration, but the situation must be diffused professionally and quickly. Exhibiting empathy should help diffuse the situation, but if that is not enough, it may be best to exit the situation and catch up again after there has been time for everyone to cool down (Bernard Golden, 2025). When frustration arises, it can sometimes be challenged with objective information, which takes the situation from being personal to being a problem the team can solve together.

People can lose their jobs over angry outbursts and can certainly harm their careers by being branded as immature or unable to control their temper. Posting frustrations to public forums or communities can have career repercussions. Public professionalism in social media and other public sites is just as important as professionalism within the workplace. While the temptation to air grievances publicly is undeniable, it is better to understand and deal with the source of the problem.

Testers periodically find themselves as the target of frustration and anger, particularly when quality issues are impacting schedules. Developers get frustrated because they are getting pressure to deliver to the schedule, but they are unable to make progress due to defect fixing. Project managers get frustrated when quality problems cause schedule delays and budget overruns. While testers are in the business of presenting factual information, that does not mean it will be happily accepted.

By respecting everyone's time, presenting objective and factual information, and demonstrating empathy, a tester can help reduce frustration within the team. While it is a temptation to sink to unprofessional behavior, it will only be costly in the long run. Testers must stay professional, regardless of the behavior of others. If there is a pattern of abuse and disrespect for testers, this may be an issue that needs to be raised to management for resolution if the individuals have not been able to turn it around. All workplaces should be safe, and testers have a right to expect that the behavior of others will not develop into a pattern that makes the workplace (virtual or physical) uncomfortable.

1.4.2 Know Yourself

The workplace can be challenging at times. Project deadlines and schedule slips can make an entire team unhappy. Bickering can become the norm when defects are reported, and test environments are faulty. This is when the tester needs to rise above. It is important that testers are respected within the team, but respect is earned over time. Not all teams will be compatible, but working professionally and providing consistently objective information will help to establish a working platform of fairness.

Everyone has their triggers, and it is important to understand these and watch for situations that may cause an angry response. Testers often feel targeted, and when this happens, talking together as a test team and including management as needed can help to re-establish the dedication to testing as a profession and a commitment to reporting objectively, regardless of whether the information is happily accepted.

2. Business Writing and Communication – 3 hrs 45 mins

Keywords

priority, severity, standup meetings, tags, test management tools, urgency

Learning Objectives for Business Writing and Communication:

2.1 General Rules for Business Writing

- SS-2.1.a (K3) For a given passage of text, apply good business writing practices to find errors
- SS-2.1.b (K2) Explain how versioning can be important for test artifacts
- SS-2.1.c (K2) Explain the differences in purpose and level of detail between a test strategy and a test plan
- SS-2.1.d (K2) Explain the purpose of a test strategy and a test plan

2.2 Business Writing in Testing

- SS-2.2.a (K2) Summarize the important characteristics of test reports
- SS-2.2.b (K2) Give examples of how traceability is important in test cases
- SS-2.2.c (K2) Summarize the important characteristics of test cases
- SS-2.2.d (K2) Summarize the important characteristics of defect reports

2.3 Miscellaneous Meetings

- SS-2.3.a (K2) Summarize ways to make defect triage meetings effective
 - SS-2.3.b (K2) Explain how daily standups can help improve the perceived value of the testing team
 - SS-2.3.c (K2) Explain how the test team should present status in project status meetings
 - SS-2.3.d (K2) Summarize the opportunities for testers to communicate in retrospectives
-

Business Writing and Communication

A big part of a tester's effective communication is writing clear and concise professional documentation. This can be in the form of plans, strategies, reports, test cases, and defect reports. This section will look at each type of documentation and how to ensure it communicates the intended message.

2.1 General Rules for Business Writing

Anything written should be considered as a record that will be kept, potentially for a long time. It may also be read by multiple people, not just the intended recipient. This section provides some guidelines to help a tester communicate professionally.

2.1.1 Be Professional

Written business communication should be professional, not casual. While a tester may have a friendly relationship with another team member, project communication must be professional. The following list will help ensure communication is professional:

- **Use complete sentences** – while texting communication may have abbreviations and partial statements, business communication should use complete and properly formatted sentences. A tester risks being classified as unprofessional by sending a report that does not use proper sentences and paragraphs.

The rule of thumb for business communication is that a sentence should contain 15-20 words, and never more than 25 (Wylie, 2025). This helps the reader to quickly determine the meaning without having to wade through extra text. Similarly, paragraphs should contain only 3-8 lines, with six being the average.

- **Check grammar and spelling** – Nothing makes a tester look less professional than grammar or spelling errors in their documents. Spelling and grammar checkers, as well as AI, provide ways to quickly check documents for these types of errors. Releasing a document with these types of errors implies that proofreading was just too much trouble, and the recipient's time is not valued. Some readers will be insulted when an unedited document is presented to them.
- **Proofread** – Much like checking the grammar and spelling, proofreading is necessary for all professional written communication. This is not to say that all text will be perfect – there will always be some type of mistake – but proofread to remove the obvious errors that are missed by the grammar and spelling checkers. Proofreading will also help identify any gaps in the information being provided and will help to adjust the tone if needed.

- **Format Correctly** – Some organizations have well-defined requirements for the formatting of the various documents that are created. Some do not have anything. It is important to provide consistent formatting, even if that format has to be created. For example, once a test plan format has been created, all subsequent test plans should use that template. The same applies to test reports, defect reports, and test cases. Consistent formatting makes document review easier for the reader.
- **Complete the Document** – If the document has a Table of Contents, Index, or History, be sure they are updated each time the document is released. The same applies to versioning and the naming of versions. This helps to avoid confusion regarding what has already been sent and reviewed vs what is a new version. An out-of-date Table of Contents or document history can be confusing.
- **Versioning** – Ensure that the document name contains the version if there will be multiple versions. This helps the recipient know which version is being discussed and reviewed. Versioning may already have a standard format in an organization, but in general, all drafts prior to initial release are usually numbered from .01 onward, with 1.0 being the first release version.

Versioning for test cases and defect reports is often handled by the test management system. It can be important to understand if older versions must be maintained (such as a set of test cases that was used for a particular release) or if updates are made directly to the master version, with previous versions being lost. Defect reports often contain a section for comments and tracking status changes and dates. This allows the tester to scroll back through the history of a defect report to understand what was changed, by whom, and when.

- **Peer Reviews** – For formal documents such as test strategies, test plans, and test reports, it is helpful to have a peer review the document prior to release. It is difficult for the author to review their own work, and items will be missed that a peer will be able to detect. Peer reviews also serve to let other people see what is being published and can be used to educate the wider team. Just as testers want developers to have their code peer reviewed, formal testing documentation should also be reviewed.

These generic recommendations apply to all professional documentation. The following sections review the guidelines for particular types of documents.

2.1.2 Test Strategies and Test Plans

The requirements for the level of detail and depth of test strategies and plans vary by organization and may even vary by project. In general, a test strategy describes why and how testing will be conducted across all projects, and the test plan describes the testing approach for a particular project (BrowserStack, 2025). Test plans should highlight any areas in which the testing will vary from what is described in the test strategy (e.g., test automation will not be implemented for a particular project because it is a short-lived product).

There may be a need to educate a product team regarding the purposes of the two documents. If there is a precedent set, a decision needs to be made to either follow that precedent or educate the team on a better approach. It is not unusual for there to be no test strategy and for the test plan to be just a set of test cases. When this happens, the “planning” aspect of the test plan is lost, and the test plan is just an execution schedule.

Determining a good time to introduce a true test strategy and plan depends on the following:

- **Criticality of the project** – more critical projects should have a formal strategy and plan that is agreed to by the project team
- **Precedent** – some organizations are resistant to what they perceive as excessive documentation
- **Need for guidance and agreement** – previous projects may have suffered because of a lack of a unified plan
- **Purpose of the documents** – at times, a test strategy is only a checkbox on someone’s schedule, which may make it meaningless for real use
- **Commitment to good testing practices** – good practices will dictate the development and use of an effective test strategy and plan

While testers agree these documents are needed, they must be reviewed, accepted, and followed by the wider project team in order to be effective. Pairing the test plan with the test reporting will help gain buy-in from the senior stakeholders who want to track progress throughout the project.

In the event that the wider project team will not accept the development of these documents (i.e., considering them a waste of time), it is still a good idea to develop at least the test plan, even if only for use by the test team. Wider adoption may take more time than is available, but it should definitely be in the long-term plans of the team. Good planning is a factor in selling the benefits of testing, but forcing unwanted documents onto the team may be counterproductive.

If the team is willing to review and adopt a test plan, it should be routed for review by the wider team. It is important that the document is mostly complete when it is routed – sending out an outline full of TBDs

is not a good idea and will tend to annoy the reviewers. If there are missing questions, write comments regarding what is missing and who needs to supply information.

The test plan should align with the test summary report, which indicates that the test plan will be in use throughout the testing process. It is important to keep the plan up to date and to review any changes with the project team. Since this plan is guiding the testing, any changes should affect the wider project team.

As with any formal documentation, it is important to ensure the test plan is a professional document. This means it should be peer reviewed and checked for the following:

- Suitable content that is project-focused
- Grammar and spelling
- Consistency in formatting using templates if available
- Good business writing
 - Sentence length
 - Paragraph size
 - Consistency in bullet lists (e.g., start each item with a verb, only use periods for each list item if any one item in the list is a complete sentence)
 - Completeness and accuracy of links (e.g., link to the project schedule)
- Readability by non-testers

Producing a professional and targeted test plan will help promote the professionalism of testing. The test plan should guide the testing for the project and should be used when estimating project completion and testing coverage for reporting.

2.2 Business Writing in Testing

A significant part of the testing effort is creating test documentation. This varies from defect reports to full test strategies. These are living documents that may be used beyond the lifetime of a current project. The professionalism of the test team is strongly reflected in the professionalism of their test documentation. Each of these document types is reviewed below with a particular focus on effective communication.

2.2.1 Test Reports

Like test strategies and plans, test reports must also be professional in appearance and content. Test reports are likely to be routed beyond the intended audience, so they must contain enough explanation that the data is clear to the reader. Some projects will use only test management dashboards to record progress, but these are usually lacking a way to explain the numbers. For example, if test progress is below expectations, the reader may need their attention called to the critical defect reports that have been logged regarding test environment blockages. Publishing only the numbers without an explanation can be a dangerous exercise and can result in a negative impression of the testing team.

Most test management tools have excellent charting capabilities. These charts allow the reports to have colorful summary pictures showing the testing progress. Trend charts can be particularly helpful to show progress toward the goal. It is important to ensure that the charts match the words explaining the charts. For example, all the tests may have been executed, but have not passed due to some outstanding defects. Presenting just the execution numbers will give the wrong impression.

Understanding how the message will be received is key to developing good testing reports. The recipient may not have any background in testing, so they may need an explanation of why a P1 defect is more impactful than a P4. It is always important to include defect metrics when test coverage/execution metrics are reported. This helps to give a balanced view of the testing status. Risk mitigation metrics are also helpful and show that testing is accomplishing its goals.

It is also important to always verify that the charts and numbers are correct. Mistakes can be made in queries that result in invalid information being reported. A sanity check of the numbers and the presentation of those numbers should always occur before a report is published. Indeed, any changes to queries should be thoroughly tested to ensure errors do not appear in the dashboards.

As with other forms of communication, the information being conveyed in a test report must be objective. Subjective impressions, if appropriate, should be clearly stated as subjective. Making go-live decisions is usually inappropriate from the testing team – the team’s job is to objectively report the status of the project. The go-live decision is a business decision that should be made based on the evidence supplied to the decision makers.

The key to good reporting is understanding the message that will be received. The sender must think like the recipient to judge how the message will be interpreted. This sometimes requires that the test manager work with the project manager or the development manager to tailor the reports so that a clear message is conveyed.

Effective reporting is a key capability for advancement in testing organizations. Analyzing the data and capturing that analysis in a clear and concise way is an important factor in communicating the message.

2.2.2 Test Cases

There are many debates about the proper level of test cases, varying from defining every step explicitly to providing only a high-level description of what should be tested. In general, there are three minimum characteristics of good test cases:

- A test case should be traceable to its origin. This can be a requirement, a test condition, an acceptance criterion, or an identified risk.
- A test case should be repeatable by the target tester. This means that the test case has sufficient detail for the person who will execute it (the target tester) to execute it and get the same results as another target tester executing the same test.
- A test case should define the expected results of the test. This clarifies the expectations and helps the team agree on what should happen.

From these three characteristics, more details can be provided to suit the industry, the organization, the project, and the test team. Safety-critical applications that are subject to audits may need very clearly identified test steps and many more fields defined for each test case, whereas other projects may be fine with just the three minimum characteristics.

Traceability is an important characteristic of a test case. Test cases can trace to the requirements either directly or via tracing to the acceptance criteria of a requirement (or story) or to a test condition that has been defined from a requirement. The point of the traceability is threefold:

- By tracing to a requirement, test coverage can be determined, and priority can be inherited so that the test cases with the highest priorities are testing the requirements with the highest priorities.

- By tracing to a risk item, risk mitigation can be tracked to show how the testing is proceeding with mitigating the highest priority risks.
- By tracing to a requirement, when/if the requirement changes, it is easy to identify which tests will need to be re-executed or modified to match the new requirement.

Test cases are usually stored in test management tools. These vary in capability and usability, but most can label or “tag” test cases with particular information. These tags can be used to quickly find particular test cases or sets of test cases using the tool’s query capability. Tags are in addition to the fields of a test case that carry identifying information. For example, a field in the test case could be the priority of the test case – something that is known when the test case is created and is unlikely to change. A tag is more temporary and could be something such as the automation status of a test case (e.g., can be automated, automation in progress, automation completed).

Folder structures are often used to categorize test cases, often by the area of software they test. For example, in an ERP system there might be a folder for the finance test cases and another one for supply chain tests. In some tools, particularly task tracking tools such as Jira, it is easy to create the test cases as tasks within a requirement or a story. The problem with this approach is that once the story is “closed”, the test case goes with it. When the test case is needed later due to an update in some other area of the software, it can be very difficult to locate. It can also be difficult to create a regression suite from these “closed” test cases because they may show up as already executed.

The easy solution to this problem is to build test cases so they are independent of the requirements. Each test case can contain a link back to the requirement it is testing, which preserves the traceability, but it lives as a separate entity, independent of the requirement, rather than a

task within a requirement. Most test management systems support this, and others, such as Jira, have plug-in test management modules that support this capability (Management, 2025). By building the test cases as independent entities, they can be exported easily into other projects, reused, and reorganized without concern of losing the linkage.

Repeatability and maintainability are always a tradeoff. A checklist is quick to write and easy to maintain, but it may be difficult for someone else to follow. Some people work well with mind maps, but mind maps are often difficult for someone else to decipher. Detailed step-by-step test cases are easy for anyone to follow, but costly in terms of creation and maintenance. Test cases that must be audited or will be used as the basis for automation have their own requirements for the detail level.

As with all formal documentation, test cases must also use proper grammar and punctuation. This improves readability and also promotes the professionalism of testing. People other than testers will read these documents, including developers, other testers, users, and project managers. Having others review test cases is a proven method of improving the quality of the test cases and the testing, but this requires that they be readable for the reviewer (BrowserStack, Test Case Review Process, 2025). As with all testing documentation, test cases must be professional, clear, and conform to the established standards.

Agreement among the team regarding the level of detail in test cases is important. This helps to set the standards and expectations, and also results in a more reliable schedule. Sorting out the identifying fields and labels before starting test case creation will save re-work later. Tester time and effort can be reduced when test management tools are leveraged. Test cases will live beyond the life of the project and must be retrievable for future use. Test cases are artifacts that represent the core of what testers do.

2.2.3 Defect Reports

Objectivity and accuracy are critical for effective defect reporting. An effective defect report leads to a quick resolution of the issue, either via a fix, an explanation, or an adjustment to the requirements. The following is a list of defect report characteristics that help improve objectivity and accuracy:

- **Record what happened compared to what was expected to happen** – this will help quickly identify any misunderstandings on the part of the testers or the developer
- **Include the steps necessary to reproduce the issue** – this clarified understanding of the defect, proves it is reproducible, and gives the developer an easy path to see the defect
- **Include screenshots** – this will help the developer locate where the defect is occurring and will provide additional information on how the defect is manifesting
- **Avoid indicating the developer made a mistake** – the tester might have made the mistake, the environment or data may be flawed, or the requirements may be unclear
- **Include a clear summary of the defect that indicates the nature and severity of the issue** – this summary will show up on defect summary reports, so it needs to be clear to ensure proper treatment of the defect
- **Get the priority/severity right** – follow the published guidelines and do not under- or over-prioritize the issue
- **Ensure the defect report is clear, concise, concrete, and correct** – save everyone time by getting all the necessary information into the report when it is first written

Assigning defect priority and severity can be difficult, particularly when there is a push to lower the priority of defects to meet the project exit criteria. In general, severity indicates the impact the defect has on the system. Priority indicates how important the defect is to the business. Urgency, if used, indicates how quickly the defect should be fixed. Urgency is sometimes used to highlight issues that are blocking testing (ISTQB, ISTQB Glossary, 2025). A tester should be able to determine severity and can usually make an educated guess regarding the priority of the defect.

In some cases, priority must be assigned by a business representative who could be a SME or a product owner. Since this assignment may lag behind, it is sometimes more expeditious to have the tester assign a preliminary priority that can be updated later. While priority and severity may seem clear to a tester, they are often points of contention with the rest of the team. In general, high-priority/severity issues are fixed more quickly, with the lower-level issues languishing until more time is available. High-priority/severity counts are often cited as exit criteria for a project, which may inspire the project team to lower the levels to meet the deadlines.

A tester must stand strong with their priority/severity assignments, while ensuring adherence to the published guidelines, but must also be flexible. Some defects look quite bad but have easy workarounds. Conversely, some defects look benign but are indications of much more serious problems. A defect's priority/severity should not be changed for political reasons, but should be changed when a compelling argument is made for the change. It is a fine line the tester walks when assigning and updating these fields, but it is a reality in testing life. Good communication, understanding, and a lack of blame will help get the right values assigned to the priority and severity of a defect and will guide the fixing.

2.3 Miscellaneous Meetings

Another form of business communication for testers is meetings. Depending on the organization, there may be many or few. Each of the following meetings is an opportunity to communicate in a way that promotes testing.

2.3.1 Defect Triage Sessions

Defect triage sessions should only be held when needed, such as nearing a production release or when the backlog is becoming unmanageable. Defect triage sessions should not be an excuse for people to not do their jobs in managing and updating the defect tracking system.

The following criteria can help make defect triage meetings effective:

- The attendees should be limited to those who need to be there
- The time for the meeting should be constrained to avoid excess re-discussion of issues
- Guidelines for which defects will be discussed should be established
- Each qualified defect should be objectively presented, including any environment, configuration, data, and reproducibility information
- Agreement should be made regarding the fixing of a defect, the timeline, the retesting requirements, and, if pertinent, release to production plans

It is important for testers to remember that not all defects will be fixed and that this is a business decision. Testers are advocates for defect fixes and are expected to lobby for the fixes they feel are important, but in the end, business relevance will take priority. The tester has to remember that this is not a personal issue, but rather a business decision to use the limited resources wisely.

If a tester is feeling ignored about a particular defect, the defect triage meeting is probably not the right place to air the problem. This is usually better dealt with between the tester and the developer, or if it needs escalation, between the test manager and the development manager with input from the project manager and the business. Compromise is always needed when some defects will not be fixed. This can be in the form of a defect status that indicates a defect can be skipped now but must be fixed in the next release (e.g., via using the urgency field) or by publishing an accepted workaround.

A tester's responsibility for a defect being fixed ends when the defect has been presented objectively and factually, when all the necessary information for a fix has been supplied, and when the proper priority and severity have been assigned. After that, it is up to the stakeholders to decide if and when it will be fixed.

2.3.2 Daily Standups

Standup meetings are a part of the Agile lifecycle model but are being commonly used with other lifecycles as well. The purpose of a standup is for each team member to communicate the following:

- What was done yesterday
- What is planned for today
- Any blocking issues

This simplistic format guarantees that every team member has time to speak and time to ask for help on blocking issues. It also ensures that the entire team understands what everyone else is doing and where their help is needed. Communicating openly and honestly in this meeting elevates testers to the same level as everyone else. They

are given equal time and equal attention. By participating in these meetings, testing is brought into the view of everyone as an important component of the project.

Testers are sometimes not included in the planning conducted by the development group. Standup meetings are a great way to understand what is planned, when software is likely to arrive, and what the developers are working on today and tomorrow. Knowing this helps testers plan what they need to be doing and opens the communication between the teams.

2.3.3 Project Status Meetings

Most projects have weekly status meetings, which are used to review the schedule, budget, milestones, and issues. These are usually conducted by the project manager and generally include test and development management as well as other stakeholders. Whether a tester is attending or is supplying information to their manager who is attending, the accurate and current status of the testing effort must be reported. This information should be supported by metrics and should reflect what the test team is reporting.

Project status meetings are a good opportunity to raise any systemic problems, such as slow defect turnaround or lack of documentation. The focus of these meetings is to talk about what will affect the project as a whole, so if there are problems that are likely to cause significant delays in testing, this is the right forum. Trend information is good for this meeting as it will show longer-term issues rather than just a point-in-time problem. For example, a trend that shows defect fixes are taking longer and longer is much more effective than complaining about one defect that has taken four weeks to be fixed.

Communication is important in these meetings as well as speaking to the data. Showing empathy to other project team members helps to build relationships that will help the testing effort. Offering to help where there is time available can also build a good reputation for cooperation. Project status meetings are an opportunity to work with the other project members in a productive way.

2.3.4 Retrospectives

Retrospectives should be conducted regularly at the end of periods of the project – such as the end of a sprint, the end of a release cycle, or the end of the project itself. Retrospectives provide an opportunity for the team to discuss what went well and what should be improved for the next time.

Testers often have a clear understanding of what could be improved. It is important to be honest in these discussions, but also to be productive. It is easy to criticize, but is that going to help the next phase of the project? Just as with presenting a problem to a manager, it is good to go into a retrospective with some suggestions for improvements rather than just complaints. For example, it is better to ask that developers put expected fix dates on defects rather than just complaining that testing does not know what is coming and is forced into reacting rather than planning.

If there are areas where testing can be improved, that information should be taken on board and discussed within the team after the meeting. That will allow everyone to absorb the suggestions (or

complaints) and come up with a constructive approach to fixing the issues. It is also good to report back to the team if changes will be made so that everyone is aware that the feedback has been received and acted upon.

Testers should also be sure to provide positive feedback, particularly to the developers and business analysts, wherever possible. These are people the testers work with daily and are the primary source of information. Positive feedback will help reinforce the relationship and will also soften any requests for changes. A retrospective should be a positive focus on improvement, and part of effective communication is supplying good and constructive feedback.

3. Promoting the Value of Testing – 4 hrs, 45 mins

Keywords

acceptance criteria, accessibility, Cost of Quality (CoQ), quality characteristics, testing mindset, usability, Web Content Accessibility Guideline (WCAG)

Learning Objectives for Selling Testing:

3.1 Level of Understanding

- SS-3.1.a (K2) Explain the three steps of risk management
- SS-3.1.b (K2) Explain how aligning testing with risk promotes the value of testing
- SS-3.1.c (K2) Summarize the ISO 25010 quality characteristics
- SS-3.1.d (K3) Apply the Cost of Quality formula to a set of defects
- SS-3.1.e (K2) Explain how schedule-driven projects can impact the test effort

3.2 Opportunities for Testing

- SS-3.2.a (K2) Summarize ways in which testers can close testing gaps
- SS-3.2.b (K2) Explain approaches that can be used when the “wrong” people are doing the testing
- SS-3.2.c (K2) Explain how SMEs can contribute to a testing effort

- SS-3.2.d (K2) Give examples of how production escapes can highlight the value of testing
- SS-3.2.e (K2) Explain how lifecycles can impact testing
- SS-3.2.f (K2) Summarize the list of people who will need to be involved in an automation solution

3.3 Empathy with the User

- SS-3.3.a (K3) Using the questions in the syllabus, determine the informal usability requirements for a project
 - SS-3.3.b (K2) Explain why accessibility is an important requirement
-

Promoting the Value of Testing

Even though it seems counterintuitive to testers, software testing often has to be sold to the project team and to management. There is an industry trend to regard testing as a “nice to have” and even as something that can easily be done by the users. Testers know this is not true, but they often find themselves in the role of explaining why testing is needed and defending the value that it provides – in short, promoting the value of testing.

3.1 Level of Understanding

In many situations, the value of testing is not understood. It is seen as a costly exercise with significant schedule impact. After all, if the developers just write better code, or if AI writes the code, why is testing needed? Every tester, including test management, must be involved in selling the benefits of testing.

In most cases, testing is not valued because of the following:

- The risk of not testing is not understood
- The complexity and technical aspects of testing are not understood
- The quality characteristics of the software have not been analyzed
- The cost of quality is not tracked
- The schedule is more important than the long-term cost of ownership

Each of these will be examined more closely in the next sections, but it is good to understand the expectations upfront.

3.1.1 Risk

All software deployments have risk. Assessing the risk honestly and realistically helps a project team understand the impact of quality-related decisions. Risk should be managed as follows:

- **Identify the risk** – the identification effort requires all members of the project team to identify risk items. According to the ISTQB Glossary, risk is a factor that could result in future negative consequences (ISTQB, 2025). For an accounting system, a risk item could be that an invoice is calculated incorrectly.
- **Assess the risk** – all the identified risks are assessed to determine the impact if the risk should become an actual event and the likelihood of it becoming an actual event. This will set the priority for mitigating the risk.
- **Mitigate the risk** – depending on the assessment, a plan is put in place to either monitor, reduce, or eliminate the risk. Testing is often used to reduce, if not eliminate, risk items.

Organizations that do not recognize the value of testing usually do not bother to identify or assess risks at the product level. One of the ways testers can help to assert the value of testing is to conduct a preliminary risk analysis and then share that information with the wider project team. While this will not be a thorough risk analysis, because it is only from the testing viewpoint, it will serve as a good starting point to call attention to the fact that there is risk and it must be addressed.

Once an organization starts to recognize that there is risk, the mitigation approach can be reviewed. If the assumption was that the users could do all the testing, once the individual risks are identified, it will quickly become clear that a user acceptance test will not accomplish the level of mitigation needed.

Organizations and project teams will often approach a project with the assumption that the software will work correctly, and the users will be happy with it. This is patently incorrect, as has been proven many times over the history of software development. Perfect requirements, perfect code, and perfect implementations rarely, if ever, occur. By identifying and highlighting risk, it will help draw attention to the many areas in which quality issues and implementation issues may be found, and will introduce reality into the planning. Reality highlights the need for testing by professional testers. Without testing, the risk is just too great.

3.1.2 Testing is Hard

Testing is often under-valued and under-appreciated. It is every tester's job to ensure that testing is understood, valued, and considered to be a vital part of a successful project. This requires educating the team regarding how difficult it is to perform good testing, and how it does not happen without careful planning and appropriate resourcing.

Good software testing requires discipline, technical knowledge, creative thought, excellent communication, and perseverance. Because of the mix of technical knowledge and excellent soft skills, it is difficult to find people who are really good at it and want to pursue testing as a career. The ISTQB has done great work in professionalizing testing and providing the basis for solid practices and training. Unfortunately, the universities still lag behind in providing good testing training for computer engineering students. Good testers seek their own training via seminars, courses, online learning, and books.

As a profession, though, testing still tends to be undervalued and minimized, and this must be combatted daily. Test managers can help the cause by specifying hiring criteria that require training, experience, and, if appropriate, certification to prove the training has been absorbed and can be put into practice. Working with the wider testing community and hosting test practice gatherings can also help to professionalize the industry. Individual testers can highlight the difficulty of testing by finding interesting and unexpected defects. Testing beyond the “happy path” helps to prove the value of testing.

When testing coverage and findings are aligned with the risk assessment, the value of testing is highlighted. If the organization agrees that there is risk and testing can prove risk mitigation in terms of test cases passed and defects found, then the value is undeniable. Demonstrating ingenuity and thoroughness in testing will also help to prove that good testers identify the difficult-to-find defects.

While professional testers should conduct the bulk of the testing, there is still a place for business subject matter experts (SMEs) to add their viewpoint. SMEs test end-to-end business flows and understand the nuances of the business processes. This testing may uncover defects missed by the professional testers, primarily because of the different approach to testing. This is valuable testing, but it should not be all the testing.

3.1.3 Testing the Quality Characteristics

The ISO (ISO, 2025) defines quality characteristics that should be used to evaluate the quality of a software product. These are:

- Functional Suitability
- Reliability
- Performance Efficiency
- Usability
- Security
- Compatibility
- Maintainability
- Portability

Testing approaches for these areas are covered in the ISTQB syllabi (ISTQB, ISTQB Certifications, 2025) for Certified Tester Foundation Level, Certified Tester Advanced Level Test Analyst, and Certified Tester Advanced Level Technical Test Analyst.

Professional testers include the quality characteristics in the risk assessment. This helps ensure that no major area is missed and highlights the importance of testing for characteristics such as usability and performance. It is easy for a project team to assume that all these areas will just work, but professional testers know better and can target the testing to mitigate the risks.

Testing for these characteristics can require special environments, configurations, and skill sets. For example, testing for performance requires a dedicated environment that is representative of production. Operational profiles must be constructed so that the testing reflects a realistic system load. Security testing requires an environment that has the same security characteristics as production, which may be difficult and expensive to set up. Formal usability testing may require usability

labs, scripting and recording, and a set of selected usability testers.

Because there may be additional cost and schedule requirements in order to adequately test these characteristics, the project team must be aware early in planning that this testing will be needed. External specialists may be required for some testing, such as security testing. Websites that must meet certain accessibility requirements may require accessibility certification testing to be conducted.

By highlighting these areas of testing, the project team will understand that testing cannot be conducted as an afterthought by the users. It must be planned, budgeted, and scheduled, and must be conducted by professional testers in order to accomplish the goals of the project.

3.1.4 Cost of Quality

One of the easiest ways to justify testing is to track and report the cost of quality. The team must understand that quality has a cost – it is never free. With this as a basis, the team can be shown how testing early is more cost-effective than testing late (such as having only UAT). Cost of quality is one of the best ways to sell testing as a cost-effective approach to product development and deployment.

Cost of quality is the total cost of all quality activities (cost of conformance) plus the cost of defects (cost of non-conformance) (ISTQB, Test Manager, 2024). A simple model sometimes only tracks the cost of defects that are found in the software. The cost assigned to each defect depends on when it was discovered in the lifecycle, with the cost being higher the later the defect is found. For example, an organization may choose to assign the following values:

Phase Found	Cost Assigned per Defect
Requirements	\$1
Unit Testing	\$10
Integration Testing	\$50
System Testing	\$100
User Acceptance Testing	\$500
Production	\$1000

The numbers should be varied based on the actual costs incurred in an organization, but this list can be a starting point. The cost of a defect includes the time spent finding it, reproducing it, documenting it, fixing it, retesting it, updating regression tests, and closing it. Production defect cost can include the cost of a help desk and the deployment of a fix, as well as more intangible items such as loss of goodwill.

The way the Cost of Quality matrix can help promote testing is by showing what a defect could have cost (assuming it was caught at the same level at which it was introduced) and the actual cost based on when it was found. For example, if a defect was introduced in the requirements, but was not caught until UAT, it cost \$499 more than it should have. It is simple for a test team to track when a defect was introduced and when it was caught. Once that information is known, the cost of quality can be computed, which will show the value that the test team has provided in easy-to-understand dollar numbers. If there is no test team, all defects will either be caught in UAT or production. Any defect caught in the earlier testing levels is money saved for the organization.

Justifying the testing effort by showing the money saved just in terms of defect capture and resolution is an easy way to quantify the benefits of testing. More complete cost of quality models will also include all

the costs for defect prevention, such as the cost of having the test team, the cost of tools, and the cost of preventative actions taken by the developers, such as test-first development.

3.1.5 Schedule Over Quality

Some projects are driven primarily by schedule. This is particularly common when a project manager's goal is to deliver on schedule, and the subsequent support of the product is someone else's problem. When this occurs, the motivation is not to deliver a quality product, but to deliver a product on time. Quality will take a back seat in this situation, and the overall cost of quality will be borne by the team responsible for support and maintenance.

This is a situation where it is particularly difficult to sell testing because the project's motivation does not support delivering quality. Testers are likely to be seen as obstructing progress. When this occurs, the best approach is to remain professional, apply good practices, and dutifully report all the defects found. They may not get fixed and indeed may be downgraded in priority, but it is important to remember that the testing job is to accurately report defects. If they do not get fixed, the job has still been done, and the testers can be proud of the work they have done.

In general, projects with this motivation are usually followed by subsequent projects with a much better focus on quality. There are times when testers just have to do their job, even if it is not appreciated at the time. In the long run, the work will be respected and valued.

3.2 Opportunities for Testing

All testers should be seeking to advance testing. It is a rare organization and a rare project that has enough time and money allocated for testing. As a result, testers need to always be ready to champion opportunities for more testing. The following are examples of some opportunities and how they can be explored:

- Testing is not happening at all in some areas
- The wrong people are doing the testing
- Busy business SMEs are expected to do the bulk of the testing
- Production issues have escaped that should have been caught earlier
- The product lifecycle could be streamlined and improved
- Test automation is not being used effectively

These opportunities will be explored in more depth in the next sections.

3.2.1 Testing Gaps

It is not unusual for there to be gaps in testing, particularly when the testing team was not involved in the initial test analysis. If available, a tester can trace back through the identified risks and test conditions to determine if coverage is being achieved. When that information is not available, it is sometimes necessary to use a combination of the requirements and the quality characteristics to determine what needs to be covered. It is often helpful for a tester to review the requirements with the author (e.g., a business analyst) to help identify and document the acceptance criteria. These acceptance criteria can be converted to test conditions, and test cases can be targeted to ensure each is covered.

One of the most difficult environments for a tester is joining a project that is already in progress. If adequate test planning has not been done, the tester can be forced into a reactive mode. The problem with this approach is that there will be gaps. The process described above is a better way to figure out what has been tested and what still needs to be tested. While this may seem like backward progress, it will save significant time by eliminating redundant testing and helping to focus the testing on the highest risk areas.

Working with the rest of the team is critical to succeeding in finding and alleviating gaps. Some areas in testing may have been skipped because the necessary environment or data was not available. This will require working with system administrators and data analysts to understand the blockages and get a resolution. Because everyone is busy, it is important to use the soft skills described in the communication section to familiarize the parties with the testing requirements and to work with them to achieve a solution.

No one intends to leave gaps in testing. There may be gaps because there is insufficient knowledge of how the software works, because there is a lack of skills to perform the testing (such as security testing), or because there just is not enough time. Gaps may be acceptable as long as there is an acknowledgement of the lack of coverage and an agreement that the resultant risk is acceptable.

Approaching testing gaps with a willingness to understand and work with the rest of the team will go a long way toward finding and resolving the issues. Showing a willingness to learn and to help will improve the speed to a resolution. If the test data is not available, perhaps a smaller set will work. Maybe a data analyst can help build new data rather than importing production data. There will be multiple solutions to the problems, but the gaps must be identified and either consciously accepted by the team or closed with testing.

3.2.2 Right Testing, Wrong People

In some cases, the testing will be happening, but it is not being planned or executed by the right people. For example, expecting the developers to do all the integration testing is probably not the best approach. It is difficult for a developer to test their own work because they lack objectivity and a testing mindset. A user with the best intentions will still tend to emphasize the positive path and end-to-end flows rather than exploring the error scenarios and processing that occurs in the background.

It is important to recognize that the testing that is occurring is likely being done with the best intentions. People are often forced into the testing role with little experience other than using the software (or a previous version of it). This does not mean their testing offers no value; it just means that there will be gaps.

When this situation is encountered, the tester should review what has been tested with the person doing the testing. Thoughtful questions can be asked regarding why some areas have not been considered, such as batch processing. This will help the tester to create a gap list, similar to what was discussed in the previous section.

When the “wrong” people are doing testing, the tester should determine in what way their skills could be better used. It may be that they can do testing, but they should be doing a particular type of testing. For example, a willing developer should be conducting unit testing and could be doing unit integration testing. They could also be looking at building test automation into the release pipeline to help build in continuous testing. A business user can conduct UAT and advise on acceptance criteria. Functional testing should be left to the professional testers, if at all possible.

One of the problems that can occur when non-testers are testing is

that the documentation does not follow the standards. Defect reports may not contain all the necessary information and may be prioritized incorrectly. If non-testers will be testing, they need to be instructed on how to use the tools, how to document correctly, and to understand the level of follow-up that will be needed in the event of defect fixes or requests for more information. Non-testers who are willing to test should not be discouraged but should be trained and focused on the areas where they can make the best contribution.

3.2.3 Busy SMEs

Sometimes SMEs are assigned to help with testing for a project, but they are also expected to complete their day job. This puts significant time pressure on them. When this occurs, it is best to have a frank discussion regarding how much time they can devote and how their schedule will work (e.g., one day a week, only in the afternoons). This will help set the expectations for the kind of work they can do and how much they can do.

SMEs have insight into the business that testers usually do not have. This knowledge should be leveraged by the testing team. When SMEs are tight on time, they may not be able to write test cases, write defect reports, and execute tests. They can still be well employed on the project, doing the following:

- Reviewing and prioritizing test conditions
- Reviewing test cases
- Validating test data
- Explaining roles and access rights
- Reviewing test case results for accuracy
- Prioritizing defect reports

These tasks leverage a SME's unique business knowledge and are less time-consuming than other testing activities. These types of tasks can also be done at a time convenient to the SME rather than having to happen at a certain time or even at a certain pace.

The tester must leverage the SME's knowledge while also respecting their time. This will make SMEs much more amenable to helping with the project and lending their knowledge when they know their time is appreciated by the team.

3.2.4 Production Escapes

One of the most effective ways to raise awareness of the lack of testing is for defects to reach production. In addition to this being a costly issue, it also highlights that testing was not sufficient. This presents an opportunity to do some forensic analysis to determine how each issue was missed and to explore how testing could be enhanced in the future to close any gaps.

While testers may be inclined for an "I told you so" moment, particularly when the project team was warned of the risk of the same type of issues escaping, that would be considered an unprofessional response to an already problematic situation. The test team needs to be seen as part of the immediate solution as well as the long-term solution.

"Production escapes" provide an opportunity to update the cost of quality figures with some actual data. While everyone knows a production defect is bad, being able to point to a particular issue and its actual cost helps everyone understand and respect the defect cost numbers.

There is no question that “production escapes” are significantly costly to an organization. The test team must use this information to promote the value of testing in risk reduction. In this way, the organization can learn, and testing can become an important part of every project.

3.2.5 Inefficient Product Lifecycle

Organizations embrace various product lifecycles in order to meet their goals, which may be faster releases, lower costs, or better inclusion of the business. However, if a lifecycle is not implemented correctly, the result may be chaos. Some lifecycles, such as Agile, de-emphasize the tester role and encourage all team members to own quality. While this is good in theory, the removal of a test organization can result in no one focusing on quality.

Test-driven development and some forms of DevOps emphasize testing occurring as early as possible in the lifecycle (ISTQB, Agile Tester, 2025). This is a great approach and can significantly reduce the defects found later in the pipeline, but it must be implemented correctly, and time must be allowed for the testing assets to be developed early. When there is an implementation of one of these test-first and general quality ownership models, the testing must occur early because there is not sufficient time budgeted for testing to occur after development is complete.

Any product lifecycle can work if it is implemented correctly and embraced by the team. Lifecycle changes are often made because the existing lifecycle is not working, but that is usually due to incorrect implementation or a lack of participation by all members of the team. When this happens, the change is not going to magically fix the problems – it may still be inefficient and result in significant quality issues.

Even when a lifecycle indicates less testing at the system level, that does not mean that testers are not needed. Indeed, testers are needed to audit that the upstream testing is actually occurring and that the coverage is sufficient.

Developers can write unit tests before developing the code in a test-driven development model, but if they do not constantly update those tests as new code is developed, the coverage will drop. Similarly, in-pipeline tests in continuous testing models become out of date, and the coverage becomes insufficient as more integrations occur.

With some lifecycles, test automation is a requirement. This means that the testing emphasis shifts from manual to automated testing. There is always a need for manual testing, but the balance of the test team may need to change to include more test engineers to develop and maintain the automation.

Regardless of which lifecycle is selected, in order to make it efficient, good communication is needed. Testers need to talk freely with the developers to understand who is testing what. Automation frameworks may be shared between the developers and testers to promote integration and a fluid testing pipeline. Working together will help to reduce potential gaps and will set expectations for the level of test coverage to be obtained.

3.2.6 Test Automation Use

As discussed in the lifecycle section, test automation can be critical for early and repeated testing, particularly in a continuous testing model. Automation development and implementation can be time-consuming and costly, so it is important to communicate clearly with the wider

project team to ensure everyone is on board with the effort. Developers need to agree with the tool selection and the integration into the test automation they may have developed. Project managers have to agree regarding who will bear the cost – the current project development effort or the longer-term maintenance effort. Other testers need to understand how manual testing will work with the automation. Communicating openly about the test automation plans will help reduce misunderstandings, resentment, and potentially wasted effort.

When planning an automation effort, getting support from other team members is very helpful. This includes the following:

- Support from other testers and the maintenance team to reduce repetitive testing
- Support from the developers to automate the release pipeline and the quality gates
- Support from the maintenance team for long-term usage and maintenance of the automation, particularly for SaaS products that will have forced releases
- Support from the business for automated testing of critical business processes to reduce repeated UAT burden

By building support and buy-in with the wider team, test automation will be easier to build and more effective. Any test automation effort should be treated as a full development project with its own design, implementation, and testing phases. Involving the team during each phase will help ensure buy-in and long-term usage.

3.3 Empathy with the User

Users expect software to work reliably and typically have little interest in testing beyond basic end-to-end checks. This expectation aligns with the value that testing provides, even if it is not always recognized. In organizations where business units fund projects, making this connection clear can help secure long-term support for testing efforts. When quality is consistently delivered, testing plays a key role — even if that role often goes unnoticed.

3.3.1 Usability

Working closely with the users to develop good test cases, understand usage, and prioritize defects helps to build a relationship with the testers. This relationship can also be leveraged to understand the usability requirements. Usability requirements are rarely clearly documented and tend to be only vague references, such as “must be user-friendly”. Only the users can define what is “friendly”, and this is usually closely aligned with their normal usage of the system.

Informal usability requirements can be discovered by determining the following:

- Can the user figure out what to do?
- Is there help text or some other assistance for the user?
- Does the software do what the user needs it to do?
- Are there extra steps that are not needed or are not logical?
- Does the UI help the user?
- Are the colors/layouts /workflow as expected by the user?

This information, in addition to formal usability requirements (see the ISTQB Usability Testing Syllabus), can be used to tailor the testing.

By working with the users to understand the answers to these questions, a strong relationship is built, and the users understand that the tester is on their side. By working with the user, suggestions for improvements can be filed into the defect tracking system to help improve the overall usability of the system. Functional testing is also helped by understanding how the users will use the system.

Usability defects are often contentious with the developers. Usability is sometimes considered more of an opinion than an actual requirement. In the absence of clear requirements, testers must trust their professional judgement and the input they get from the users. This will help determine what usability issues are defects and how they should be prioritized. Whenever in doubt, usability issues should be documented as defects or improvement suggestions. In this way, while they may not get fixed for the current release, the fixes can be incorporated into future releases.

3.3.2 Accessibility

Accessibility is a component of usability with the goal that the software be usable by all potential users. Accessibility requirements are often driven by legislation that states that a certain level of accessibility must be achieved. This is frequently dependent on a Web Content Accessibility Guideline (WCAG) assessment (WCAG, 2025) This is a widely accepted international standard for usability and has set requirements that must be met for the software to achieve a conformance level (e.g., A, AA, AAA). There may also be local requirements for accessibility that must be considered.

In some cases, accessibility testing helps to mitigate the risks of violations that may result in lawsuits if the software is not compliant with the applicable standards and regulations. Because accessibility

requirements are usually stated more clearly than usability requirements, there is less need to interact with the rest of the project team to understand what needs to be tested. It will be necessary, though, to interact with the developers when issues are found. Because this testing is usually clearly specified, and the defects are often classified as high-risk, there are fewer issues with the resolutions. As with all defects, the defects must be clearly stated along with a reference to the part of the standard that is not being met.

Accessibility testing is an opportunity to sell the value of testing. Some products cannot go live without the right level of conformance, so that makes testing a critical component of product release. Any opportunity to highlight the importance and value of testing helps to “sell” testing.

4. Professional Development – 2 hrs 45 minutes

Keywords

None

Learning Objectives for Professional Development:

4.1 Rising in the Organization

- SS-4.1.a (K2) Summarize how to prepare for a promotion
- SS-4.1.b (K2) Summarize approaches that can help smooth transition to a new position

4.2 Interviewing for a New Position

- SS-4.2.a (K2) Explain the purpose of a cover letter
- SS-4.2.b (K2) Summarize ways a tester can align with a potential job
- SS-4.2.c (K3) Prepare for an interview for a job that will stretch current capabilities
- SS-4.2.d (K2) Summarize interview questions the interviewee should ask
- SS-4.2.e (K2) Explain areas to be explored to ensure a good fit
- SS-4.2.f (K2) Explain how the testing industry continues to offer opportunities

Professional Development

A tester should always be planning for their next position. How will they prepare for it? How will their current work make them ready for the next role? This may involve changing roles within the current organization or moving to a new one. Either way, a tester must be thinking about where they want to go and how they will get there.

4.1 Rising in the Organization

Promotions are a way to grow in a role and expand capabilities. In testing, there are generally two paths available: technical and management. Ideally, an organization has two distinct lines for promotion so that people who are technically inclined are not forced into management roles they do not want just to be promoted. Strong technical skills and strong management skills are frequently not shared in a single person – there is usually a preference for one or the other. This does not mean that a manager cannot be technical and that a technical person cannot be a manager; it just means that they should not be forced to be experts in both areas if they are not interested in both.

4.1.1 Preparing for Promotion to a New Role

Most people like the idea of being promoted. This does not mean they are ready for promotion or that they will even be happy in the new role. The first step in preparing for promotion is to understand the requirements for the desired position. This includes the following:

- Education/training requirements
- Responsibilities
- Alignment with skills
- Alignment with career goals
- Availability of support for the new role
- Changes in reporting structure

Once the new position has been assessed and the job description reviewed, the tester can determine how closely aligned they are with the new role. There may be a significant delta that has to be closed with additional training, experience, and time. Having a realistic understanding of the requirements of the new job and the tester's current capabilities will help create a realistic timeline for promotion.

After the requirements have been assessed and the tester has made their own training and skill development plan, it is time to talk to management. When a tester approaches their manager about a promotion, they can make a good impression by presenting their understanding of the requirements for the role, matched with their current skills and career plans. This will pave the way for an open discussion of what is needed to achieve the promotion.

Even when deserved, promotions are not necessarily available. This may be due to constraints within the organization that limit the money available, the number of people who can hold particular positions, or no current openings. This does not mean that the tester should not

continue to develop their skills to be ready – indeed, they want to be sure they are ready when the position becomes available.

Not everyone should manage or lead people, and not everyone wants to have that responsibility. If the promotion path requires management experience, the tester may want to explore if there are opportunities for coaching/mentoring responsibilities without full administrative management responsibilities (e.g., hiring, firing). This can build leadership skills and help determine if this is a good career path for the tester.

4.1.2 Receiving a Promotion to a New Role

In some cases, a new promotion will actually mean very little change to the day-to-day work. This happens when the tester was already performing most of the duties of the higher position. However, it may be a substantial change, particularly when there are new responsibilities for leading people. Former peers may now be direct reports. The following approaches can help smooth this transition:

- **Do not assert authority** – respect is earned
- **Do not try to be a buddy** – the role is different now, and being “one of the gang” is probably not possible
- **Maintain good relationships and friendships** – those are real and will be needed
- **Remember that some information can be shared, and some cannot**

It is important to remember that the role is different, and that peers will take time to adjust to the change. If the promotion is from a senior test analyst to a principal test analyst, the responsibilities may be

greater, but not substantially different. When the promotion is to a management position, the transition is more significant for everyone.

In the end, a title is just a title. Performing the job well, earning respect from peers and co-workers, and demonstrating consistent professionalism will determine a tester's success regardless of the role.

4.2 Interviewing for a New Position

A new position may be within the existing organization or may be with a new organization. The preparation for the interview should be the same for either situation. The tester should approach any interview with the same level of interest, preparation, and enthusiasm.

4.2.1 Preparation

The first step for any interview is to ensure the resume is correct. This means that it needs to be up to date, use industry terminology, and accurately represent the various job responsibilities. Some organizations use unusual position titles for testers, such as programmer analyst. These are not usually descriptive of the real role, so it is important to use industry-accepted terms such as senior test analyst rather than a term that might be misleading. Good interviewers will review resumes before the interviews, and unusual job titles can result in resumes being rejected.

A resume should present the tester in the best possible light. Experience that is relevant to the new role should be highlighted, as well as any other noteworthy accomplishments. It is important that a resume be

an accurate statement of ability. Claims of “expert” status are usually disregarded by interviewers due to the prevalence of the term on resumes where the claim is unfounded. Resumes must be accurate, as any inaccuracy that is detected during an interview will lead to distrust.

When applying for a job, a targeted cover letter can differentiate between similar resumes. In a competitive job market, just getting an interview can be difficult, so using a cover letter to help open the door is a smart approach. Researching the company and role will help target the cover letter and make it stand out. Cover letters are also an opportunity to demonstrate good verbal communication skills, and that can be a significant differentiator for the potential manager. It may be tempting to use AI to write the cover letter, but a personal letter is a safer and preferable option. Many managers routinely screen for AI-generated text, and it is usually considered a negative if the text is not written by the candidate.

In the case where the manager’s name and role are known, the tester should research that person to be able to personalize the interview as much as possible. Researching the organization and manager, as well as supplying a well-constructed cover letter, shows a real interest in the role. This can help get that critical first interview and make a resume stand out from the rest.

4.2.2 Promoting your Skills and Capabilities

The better the tester’s skills match the desired position, the more important it is to highlight those skills to the manager. The tester should also demonstrate curiosity about the role and discuss how they will fit into that role to the benefit of the organization. Interviewees sometimes concentrate more on how a position will benefit them than how their taking the position will benefit the manager and the

organization. Keeping that focus during the interview will show a strong interest in the position and the longer-term interest in the organization.

During an interview, the tester should recount situations they have encountered that are relevant to the new position. This will help align the tester with the position in the manager's mind and will help them picture the tester in that role. Ideally, the tester's skills are already a good match for the position, but if they are not, it is up to the tester to demonstrate how they can flexibly fulfill the responsibilities of the position.

Getting a new position is not all about skills fit. It is also about personality fit – both into the team and into the manager's organization. It is important to show interest and curiosity about the role. Enthusiasm is critical, but this needs to be a mature enthusiasm, not silliness. The tester needs to be engaging throughout the interview, making a good connection with the manager. Above all, and particularly as the workplace changes, the tester must show flexibility. This may mean showing an expectation that there will be some irregular hours, on-site/off-site work, occasional weekends, and the peaks and valleys of project work. Showing the maturity and flexibility to deal with these realities will help the interviewer understand that the tester accepts the requirements of a testing job.

4.2.3 Asking Questions

Even after reading all the information about a new position within the organization, a tester should still have some questions. These could be any of the following:

- Reporting structure
- “Normal” day
- Project schedules
- Team size and structure
- Management/Lead structure
- Normal team composition and software development lifecycle
- Project constraints
- Feature vs schedule vs quality tradeoffs
- Documentation expectations
- Good and bad features of the job
- Which skills are particularly valued

By asking questions of the interviewer, the tester shows an interest in how they will fit into the job and the organization. The answers to these questions may spawn more questions. It is important to remember that the interview works both ways – the manager wants to get the best person for their position, and the tester wants to make sure they are making a move that is good for their career and their well-being.

4.2.4 Ensuring a Good Fit

It can be hard to decide if a new position is the right fit. Changing jobs is stressful, and a tester wants to be sure the change is going to be a positive move in their career. New jobs should come with new challenges, but the tester must be realistic about what skills will be needed and how those skills can be acquired (e.g., training, on-the-job experience, self-education). If there are areas where the tester is

particularly weak, it will be important that the new job provides support while skills are being learned. The tester will need to learn to use new tools, which may or may not be documented.

This is why it is so important that the tester not oversell themselves, but also not undersell. Everyone must learn, and in the software industry, there is always rapid change. The ability to learn and adapt is critical, not just when changing jobs. With the advent of AI in software testing, everyone is learning again.

Job location and hours must be considered. It is normal for testers to have to work longer hours immediately prior to a release, but that should be balanced with shorter hours at other times. It is also good to understand where the project team will be located. If the testers are required to be on-site but the rest of the project team is remote, that may cause communication challenges. If this is the case, it is good to understand why this is needed. If there is an expectation that a percentage of work time will be work-from-home, that needs to be clearly stated. If there is an employment agreement, that should be documented. Testers will always need to exhibit flexibility, but it is good to understand the expectations up front.

4.2.5 A Career in Testing

Testing is an ever-changing field that will always provide technical and personal opportunities for learning and growth. As the software industry continues to evolve, testers must also evolve. Industry changes have created new positions for mentors, coaches, managers, and technical leaders. Testing can offer an interesting and satisfying career, but it will always have its challenges. Mastering soft skills including the ability to communicate effectively, skillfully advocate for the value of testing, and plan and make career transitions will go far in building long-term satisfaction in a testing career.

5. References

- Adzic, G. (2009). Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing. Neuri Limited.
- Adzic, G. (2009b). Test Driven Development: By Example. Addison-Wesley.
- BATimes. (2022). Retrieved from BATimes: <https://www.batimes.com/articles/did-you-know-that-a-business-analyst-is-as-important-in-software-testing-as-a-qa-specialist/>
- Bernard Golden, P. (2025). Recognizing, Understanding, and Managing Workplace Anger. Retrieved from Psychology Today: <https://www.psychologytoday.com/nz/blog/overcoming-destructive-anger/202501/recognizing-understanding-and-managing-workplace-anger>
- BrowserStack. (2025). Test Case Review Process. Retrieved from BrowserStack: <https://www.browserstack.com/guide/test-case-review>
- BrowserStack. (2025). Test Plan vs Test Strategy. Retrieved from BrowserStack: <https://www.browserstack.com/guide/test-plan-vs-test-strategy>
- Conciu, O. (2023). Comparing Test Cases and Acceptance Criteria. Retrieved from Software Testing Magazine: <https://www.softwaretestingmagazine.com/knowledge/comparing-test-cases-and-acceptance-criteria/>

- Enders, A. (1975). An Analysis of Errors and Their Causes in System Programs. IEEE Transactions on Software Engineering, pp. 140-149.
- Grammarist. (2025). Read the Room - Origin and Meaning. Retrieved from <https://grammarist.com/idiom/read-the-room/>
- Integration, G. (2025). Seven reasons to put your camera on in virtual and hybrid meetings. Retrieved from Global Integration: <https://www.global-integration.com/insights/seven-reasons-to-put-your-camera-on-in-virtual-and-hybrid-meetings/>
- ISO. (2025). ISO/IEC 25010. Retrieved from ISO 25000: <https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>
- ISTQB. (2024). Test Manager. Retrieved from ISTQB : <https://istqb.org/certifications/certified-tester-advanced-level-test-management-ctal-tm-v3-0/>
- ISTQB. (2025). Agile Tester. Retrieved from ISTQB: <https://istqb.org/certifications/certified-tester-foundation-level-agile-tester-ctfl-at/>
- ISTQB. (2025). ISTQB Certifications. Retrieved from ISTQB Certifications: <https://istqb.org/certifications/>
- ISTQB. (2025). ISTQB Glossary. Retrieved from ISTQB Glossary: <https://glossary.istqb.org>
- Management, T. (2025). The Best Test Case Management Tools for Jira. Retrieved from Test Management: <https://www.testmanagement.com/the-best-test-case-management-tool-for-jira/>

- Marick, B. (2003). Exploration through Example. Retrieved from <http://www.exampler.com/old-blog/2003/08/21.1.html#agile-testing-project-1>
- Review, H. B. (2015). Prevent Email Horror with a 2-minute Send Delay. Retrieved from <https://hbr.org/2015/07/prevent-email-horror-with-a-2-minute-send-delay>
- WCAG. (2025). WCAG Overview. Retrieved from W3C Web Accessibility Initiative: <https://www.w3.org/WAI/standards-guidelines/wcag/>
- Wylie. (2025). How long should a sentence be? Retrieved from Wylie Communications: <https://wyliecomm.com/how-long-should-a-sentence-be/>

5.1 Standards

- ISO/IEC 25010 (2011), Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models

6. Trademarks

ASTQB® is a registered trademark of the American Software Testing Qualifications Board

ISTQB® is a registered trademark of International Software Testing Qualifications Board

7. Appendix A – Learning Objectives/ Cognitive Level of Knowledge

The specific learning objectives applying to this syllabus are shown at the beginning of each chapter. Each topic in the syllabus will be examined according to the learning objective for it.

The learning objectives begin with an action verb corresponding to its cognitive level of knowledge as listed below.

Level 1: Remember (K1)

The candidate will remember, recognize and recall a term or concept.

Action verbs: Recall, recognize.

Examples

Recall the concepts of the test pyramid.

Recognize the typical objectives of testing.

Level 2: Understand (K2)

The candidate can select the reasons or explanations for statements related to the topic, and can summarize, compare, classify and give examples for the testing concept.

Action verbs: Classify, compare, differentiate, distinguish, explain, give examples, interpret, summarize

Examples	Notes
Classify test tools according to their purpose and the test activities they support.	
Compare the different test levels.	It can be used to look for similarities, differences, or both.
Differentiate testing from debugging.	Look for differences between concepts.
Distinguish between project and product risks.	Allows two (or more) concepts to be separately classified.
Explain the impact of context on the test process.	
Give examples of why testing is necessary.	
Infer the root cause of defects from a given profile of failures.	
Summarize the activities of the work product review process.	

Level 3: Apply (K3)

The candidate can carry out a procedure when confronted with a familiar task, or select the correct procedure and apply it to a given context.

Action verbs: Apply, implement, prepare, use

Examples	Notes
Apply boundary value analysis to derive test cases from given requirements.	Should refer to a procedure/technique/process etc.
Implement metrics collection methods to support technical and management requirements.	
Prepare installability tests for mobile apps.	
Use traceability to monitor test progress for completeness and consistency with the test objectives, test strategy, and test plan.	It could be used in an LO that wants the candidate to be able to use a technique or procedure. Similar to 'apply'.