ASTQB Certified Tester Syllabus
IoT and Mobile Testing

ASTQB®

American Software Testing Qualifications Board

Table of Contents

0. Introduction to this Syllabus

0.1. Purpose of this Document

This syllabus forms the basis for the ASTQB certification for IoT and Mobile Testing. ASTQB provides this syllabus as follows:

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

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

02. Examinable Learning Objectives

The Learning Objectives for each chapter are shown at the beginning of the chapter and are used to create the examination for achieving the IoT and Mobile Testing Certification.

1. Introduction to Testing IoT and Mobile Devices - 75 mins

Keywords

Internet of Things, hybrid application, mobile application testing, mobile web application, native mobile application, wearables testing

Learning Objectives for Introduction to IoT and Mobile Devices Testing

1.1 Introduction

None

1.2 What is an IoT or Mobile Device

CD-1.2.a (K2) Understand the challenges that are faced when testing IoT and mobile devices

1.3 What is an IoT or Mobile Device Application

CD-1.3.a (K2) Understand the difference between a traditional mobile application and an application that supports an IoT or mobile device in the Internet of Things

1.4 Expectations from IoT and Mobile Device Users

CD-1.4.a (K2) Explain the expectations for an IoT or mobile device user and how this affects test prioritization

1.5 Challenges for Testers

CD-1.5.a (K2) Explain the challenges testers encounter in IoT and mobile device application testing and how the environments and skills must change to address those challenges

CD-1.5.b (K2) Summarize the different types of mobile applications

1.6 Necessary Skills

CD-1.6.a (K1) Recall the necessary skills needed by an IoT or mobile device tester

1.7 Equipment Requirements

CD-1.7.a (K2) Explain how equivalence partitioning can be used to select devices for testing

1.8 Lifecycle Models

CD-1.8.a (K2) Describe how some software development lifecycle models are more appropriate for IoT and mobile device applications

1.1 Introduction

The world of IoT and mobile devices is expanding. What started with mobile phones soon became smartphones and tablets, and then blossomed into the Internet of Things (IoT). Smartphones and tablets are IoT and mobile devices and are often simply referred to as mobile devices (that move around and connect from different points and via multiple protocols). Because they connect to the Internet, they are also a part of the IoT. Some devices, such as smart refrigerators and home security systems, are IoT and mobile devices and a part of the IOT, but are not “mobile”. In this syllabus, the general term “IoT and mobile devices” is used for the inclusive set of all devices that have some ability to connect to the Internet. Mobile devices and mobile applications generally refer to phones and tablets. Some devices, such as automobiles, are certainly mobile, but are more aligned with the IoT than with mobile devices and applications.

As the software for IoT and mobile devices has adapted to be faster and smaller, testing must adapt to be quicker and more lightweight. That being said, good testing practices still apply and quality characteristics still matter to the users, particularly when an application is safety-critical. It is important to remember that the users have expectations that their mobile applications and IoT devices will “just work”.

1.2. What is an IoT or Mobile Device?

Devices in the connected world vary from smartphones to refrigerators, from tiny humidity detectors to cars. Anything that is capable of supporting an Internet enabled component is capable of joining the IoT. The same software may be supported on a variety of devices and operating systems (OSs), making compatibility testing more challenging and future-proofing difficult. As the industry leaps forward, backward compatibility is often receiving less emphasis when more is actually needed. There is an expectation from users to not be forced to upgrade in order to take advantage of new features and applications.

Personal digital assistants possess the ability to control multiple IoT and mobile devices at the voice command of the user. These devices are manufactured by a variety of companies and must all interact correctly with their respective IoT and mobile devices. Smartphones can also provide this functionality, depending on the interface of the IoT and mobile device.

Copyright ASTQB,
All Rights Reserved

Testers can no longer expect to have access to all the devices that will use the software under test. Selecting a representative sample set is a critical part of defining test coverage and risk mitigation. For instance, refrigerator software that allows the user to increase or decrease the temperature remotely may not need to be tested on all models of refrigerators. It is important for testers to understand what exactly is to be tested. Is it the device that provides the connectivity? Is it the response of the target device? Is it the communication between the two? Realistically, it is all of these. However, if all refrigerators use the same communication interface with the Internet device, then testing one may be sufficient (i.e., applying equivalence partitioning).

1.3. What is an IoT or Mobile Device Application?

With the ever-increasing scope of IoT and mobile devices, applications are no longer limited only to mobile devices. However, mobile applications which reside on smartphones, tablets and other traditional types of mobile devices are still a very important part of IoT and mobile testing. In fact, with the rapid increase in IoT and mobile devices, the need for mobile apps to integrate and interoperate with these devices has only increased. For example, some mobile apps offer versions for smart watches too.

In the context of hand-held mobile devices, applications generally fall into two categories—those developed specifically to be native applications and those that were designed to be viewed through a web browser on a mobile device. From the user's viewpoint, there is no difference, although some browser-based applications may be optimized for the mobile device, providing a richer (or at least more readable) user experience. From the developer's and tester's viewpoint, there are different challenges, goals and success criteria. This syllabus focuses on the applications specifically developed for use by a wide range of IoT and mobile devices, such as the Internet of Things, as well as traditional hand-held mobile devices.

Copyright ASTQB,
All Rights Reserved

There will also be some discussion about applications that have become mobile despite the original intentions. Examples of these include medical devices, appliances and automobiles. Traditional mobile devices include any of the so-called hand-held devices (including (dumb) mobile phones, smartphones and tablets/netbooks), as well as devices that have been created for a specific use (such as e-readers or a device used by a parcel delivery service that allows the driver to record delivery, the customer to sign, and an image to be taken documenting the delivery). Mobile devices also extend to wearable items, such as smart watches and glasses that allow access to specific applications, and may include additional native functionality, such as telling time or improving vision.

The field of IoT and mobile devices is continually expanding as new uses are devised and devices are created to support those uses.

1.4. Expectations from IoT and Mobile Device Users

IoT and mobile devices are becoming critical to daily living. Users expect 100% availability regardless of what they do to the device or the software. They expect usability that allows them to download and immediately use an application with no instructions or training. They expect exceptional response time regardless of what else the device is doing, and regardless of the strength or capability of the network.

One of the most difficult aspects of IoT and mobile testing is defining the users’ expectations. The user group for an application can be quite large and the expectations may vary dramatically, even within the target users. It is particularly important that the requirements for the quality characteristics are clearly defined in a measurable way in the requirements or acceptance criteria for the software. With all quality criteria, there is a range of what is considered “acceptable”. Defining and documenting this range early in a product’s lifecycle is particularly important for the IoT and mobile devices. By defining the criteria for the quality characteristics at the beginning of the project, all architecture, design and implementation decisions can be made to align with and fulfill those objectives.

Users have become impatient with slow software and have no tolerance for software that is difficult to use. Usability and performance testing have become vital to the release of any IoT and mobile device application because of the expectations of the user, not necessarily because of the criticality of the functionality. This is different from traditional software where users are somewhat committed to using an application, even if it was a bit slow or awkward to use, particularly for enterprise software where the employee has no choice but to use it. If an IoT or mobile device application is too slow or unattractive, the user will often have an option to download a different application. Organizations can lose customers if their mobile applications are not fast enough or pleasing enough to use. Competition in the IoT and mobile device industry is fierce, raising the importance of good testing to ensure high quality products.

1.5. Challenges for Testers

IoT and mobile device users are everywhere and include everyone. Never before in the history of software has the user community been so vast or varied. IoT and mobile device uses vary from recreational to business-critical. Users expect seamless connectivity and instant access to information. The Internet of Things has put access into the hands of many but has also increased the expectation for all applications and devices to provide a consistent experience [InfoQ]. The Internet of Things includes many items that are not particularly mobile (such as refrigerators) or handheld devices (such as drones).

The set of IoT and mobile devices is continually growing. Software is expected to work well across a growing set of devices with constantly increasing capabilities, while providing an ever-expanding set of functionalities to the novice and expert user.

1.5.1. FREQUENT RELEASES

One of the biggest challenges to testing is the frequency of the release cycles. Because the IoT and mobile device market is so competitive, organizations race to be firstto-market with new features and capabilities. In order to meet these demands, support for development environments and tools has increased dramatically, making the bar for entry into the market much lower than ever before. There are free development kits, free or inexpensive training and free distribution channels. This leads to a large number of developers with the ability to quickly create and deploy an application.

Testing has to adapt to the demands of time-to-market while also meeting the expectations of the users regarding functionality, usability and performance.

In the digital assistant realm, it is common for a vendor to release a new version of operating software which may impact the functionality of APIs. If the manufacturer of IoT and mobile devices controlled by the digital assistant are not prepared for the new release, the users may experience unexpected failures. This is a challenge for testers of the IoT and mobile devices as they must keep track of upcoming releases, some of which may not be well-publicized.

Copyright ASTQB,
All Rights Reserved

1.5.2. PORTABILITY/COMPATIBILITY

Although invisible to most users, there is an expectation that applications will work across devices and that devices will work together. There is an expectation to be able to easily and automatically transfer data between devices, and to use the same applications from any device. The portability of an application is highly dependent on how it was developed and the deployment target.

The typical application types include the following:

- **Traditional browser-based applications –** The application is designed to work in a browser on a PC. It may or may not function well and provide adequate usability (e.g., scaling) when accessed from an IoT or mobile device.

- **Mobile web sites –** The application is hosted on the server but it is designed for mobile access across multiple compatible devices. Portability is a key concern.

- **Mobile web applications –** The application is developed for use by a variety of devices, with the majority of the code residing on the web site. Mobile web applications must have communication with the web server in order to run. In some cases, the applications are the same as those used on the web site, but generally a mobile version of the application is what is used by the mobile device.

- **Native mobile application –** A native mobile application is designed for a specific IoT and mobile device family. These applications reside on the device and communicate directly with the device through the device’s interfaces. Coding is normally done using tools designed specifically for the device. Testing a native mobile application requires either the specific device or simulators for that device and its software. This is especially true with wearables and Internet of Things appliances, including very complex platforms, such as automobiles.

- **Hybrid applications –** Rather than being coded with the native device tools, these applications use a library or framework to handle platform specific differences. Device functionality is accessed via plug-ins that may be unique for different families of devices. Hybrid applications are designed to be more portable than native mobile applications, but are still able to access unique device capabilities. Hybrid applications are often dependent on some level of connectivity with a web server and may also be subject to device/ browser compatibility issues.

It is important for a tester to understand the intended target device(s) in order to know the portability requirements for testing.

1.6. Necessary Skills

Functional testing is required for IoT and mobile device applications. The tester must have the skills necessary for manual functional testing tasks, including requirements analysis, test design, test implementation, test execution, and results recording and reporting.

In addition to the standard testing skills that are needed in any environment, IoT and mobile device application testing also requires good capabilities for testing specific quality characteristics: security, usability, performance, portability/compatibility, and reliability.

These quality characteristics, skills and techniques are covered in Chapter 3.

1.7. Equipment Requirements

Depending on the expected usage of the application, testing needs to cover representative devices. Representative devices are those whose behavior can be determined to be representative of other devices in the same class. For example, it might be determined that all iOS devices will behave the same way when running an application. Therefore, only one of those representative devices needs to be tested. The results from the test on one device is the same as would be seen if the same tests were run on another iOS device. This is equivalence partitioning applied at the device level.

Most devices will not fall into such large categories of similar behavior, so it is likely that a sample set of devices will be needed to determine compatibility for an application across devices. This usually results in having to acquire a large set of physical devices, use simulators, rent a lab full of devices or use alternate testing approaches. These options are discussed in more depth in Chapter 5.

It is important for the tester to approach any IoT and mobile testing project with a clear understanding of the equipment requirements. This is a key part in effective planning to determine the budget and schedule and also requires the proper allocation of test cases across the various devices. Factors such as device location (rural, city), weather (sunny, rainy), usage location (indoors, outdoors), connectivity (Wi-Fi, cellular) and others are significant in selecting the testing approach, people and locations.

1.8. Lifecycle Models

The requirements for fast development and deployment have pushed the software development lifecycle toward the iterative models, including Agile. Rapid prototyping is often used to quickly develop, gain feedback and successfully deploy a new product. Existing products are updated frequently and there is a tendency in the market to “push it out and let the users test it”. This often results in frantic fixes being deployed to unhappy users.

IoT and mobile device testers need to employ testing that will not substantially slow the progress of the product to market, but will help reduce the risk of a catastrophic failure. Risk-based testing approaches are critically important in the mobile application industry because there will never be enough time to test everything. The amount of risk and the corresponding amount of testing are correlated to the usage and criticality of the product. Smartphones, for example, have a wide variety of uses, some of them safety-critical.

It is important to evaluate each application individually for its risk factors rather than to group a set of applications together. Even though the functionality may be similar, the actual usage may determine the criticality. For example, if a viewing application should be able to display images at a certain resolution, not achieving that resolution might not matter for someone’s holiday pictures, but could be safety-critical if those images are used by a remote doctor to analyze skin abnormalities to determine cancer treatment. Once a proper risk analysis has been conducted, the testing can be allocated to mitigate the risk and achieve the desired level of confidence in the released product.

Because many IoT and mobile device applications are able to accept updates "over-the-air" (OTA), sending updates may be relatively easy and fast and it may be possible to force installation of the updates. Other IoT and mobile devices that have to be loaded from a central source (for example, a PC) may not be as easy to update quickly if a significant defect is found. The ability and ease with which updates can be applied may be a factor in determining release risk. It is also a factor in determining how much effort will be needed for maintenance testing.

Many IoT and mobile device applications are developed incrementally. An initial, simple version of the application is developed and deployed. Features are then added incrementally as they become ready, and as the market demands. This type of development allows the product to be introduced quickly without compromising quality, while additional features are developed internally with testing time allocated.

Sequential lifecycle models (e.g., V-model, waterfall) are used less frequently for IoT and mobile device applications due to the need to get a product to market quickly. Documentation tends to be minimal and testing tends to follow more lightweight methods with less documentation. Safety-critical applications still tend to follow sequential models, as do other applications that are under regulatory control.

Testing approaches are discussed in Chapter 2.

2. Test Planning and Design – 60 mins.

Keywords

minimal essential test strategy, operational profiles, risk analysis

Learning Objectives for Test Planning and Design

2.1 Identify Functions and Attributes

CD-2.1.a (K2) Explain why use cases are a good source of testing requirements for IoT and mobile device applications

2.2 Identify and Assess Risks

CD-2.2.a (K2) Describe different approaches to risk analysis

2.3 Determine Coverage Goals

CD-2.3.a (K2) Explain how coverage goals will influence the level and type of testing to be conducted

2.4 Determine the Test Approach

None

2.5 Identify Test Conditions and Set Scope

CD-2.5.a (K2) Describe how test analysts should take the device and application into consideration when creating test conditions

2.6 Regression Testing

None

2.1 Identify Functions and Attributes

Feature-rich IoT and mobile devices are difficult to test. It is important to focus on the functions and attributes that are within scope for the testing effort. For example, if the goal is to release a new application across multiple smartphones, the focus will be on the capabilities of the application, the interaction of that application with the device and the quality characteristics that are important for the success of the application (i.e., usability and performance). If the project is to release a new smartphone, the scope is different. In this case the tester will focus on the capabilities of the phone itself, its ability to support a sample of applications, communication between the device and the network (also Wi-Fi and other forms of communication, such as IPover- USB), and various other quality characteristics. The focus of this syllabus is on testing the IoT and mobile device applications and the interfaces with host devices and other devices, rather than the device itself.

Requirements tend to be brief for IoT and mobile device applications. There may be a specification, a requirements document, use cases or user stories. In general, the tester should not expect comprehensive requirements and should instead plan to work at the use case level where usage scenarios are identified. If the use cases are not available, the tester should seek them out to understand the expected usage and to focus the testing accordingly.

In order to scope the testing, it is important for the tester to understand the attributes of the application that are important to the user and prioritize them appropriately. If security and performance are more important than usability, this will help to identify the risks and determine the amount and type of testing that will be needed in each area. The stakeholders must understand that each attribute to be tested will require an investment in people (with the appropriate skills), tools and environments.

2.2. Identify and Assess Risks

An IoT or mobile device application project that is not safety-critical or mission-critical is usually characterized as being feature-rich but time-poor, meaning that there are many features but little time for implementation and testing. Requirements tend to be brief and informal. As a result, when identifying and assessing risks, it is important to use a lightweight process. One way to approach the risk analysis is to think of the application in two ways - the physical and the functional.

For the physical capabilities, consider the items that are physically touched by the user (e.g., buttons, icons, display, graphics) and physical features of the device that are used by the user (e.g., rotation, accelerometer). These capabilities enable the functionality of the application but are not functions themselves. Once these items are identified, create a grid that lists the category of the item (e.g., display) and list the capabilities that are of critical, high, medium, and low importance to the user. For example, it might be critical that an image loads completely in normal situations, high importance that the resolution is acceptable, medium importance that it loads consistently without retries, and low importance that it retries the load if the connection is dropped. Similarly, it might be of critical importance that an image rotates when the device is rotated, high importance that it resizes upon rotation, and medium importance that the text rotates with the image.

For the functional capabilities, consider the features of the software (e.g., accurate map loading for a navigation application). In this case, it might be critically important that the correct map is loaded, highly important that the map shows the user's location, medium importance that the map shows fuel stations, and low importance that it shows construction sites.

This lighter-weight approach allows the tester to understand the physical aspects of the device that will need to be tested, either on a real device or a simulator, as well as the features that are important to the user. By working through a spreadsheet of this type, the tester can find requirements that may not have been stated and can help discover features that are implemented but not documented.

Examples of lightweight approaches to risk analysis are available from multiple sources. Traditional risk analysis approaches can also be used in a lighter-weight fashion to better fit IoT and mobile device application testing. See [Paskal] for information regarding the Minimal Essential Test Strategy (METS), which is a lightweight approach to risk prioritization.

It is important for the tester to adapt the risk identification and assessment process tofit within the timelines of the project. Heavyweight methods will not be successful in this environment and will tend to delay the testing.

It may also be useful to consider production metrics when defining risk areas. For example, the following metrics could be used [Webtrends]:

- **Total downloads –** Indicates the amount of interest in the application and provides the upper bound for the maximum number of concurrent active users.

- **Application users –** Indicates how many people actually use the application (not just downloaded it).

- **Active user rate –** Provides the ratio of the number of application users to the total number of downloads.

- **New users –** Provides the number of users who first used the application within a period of time (particularly interesting when compared to the attrition rate, which can be derived from the active user rate).

- **Frequency of visit –** Provides the ratio of the number of mobile web site visits to the number of users over a period of time (can be used to gauge user loyalty).

- **Depth of visit –** Indicates the number of screens viewed during the average visit.

- **Duration –** Indicates the average amount of time spent in the application.

- **Bounce rate –** Provides a ratio of the number of mobile web site user visits that had only a single view (people who downloaded the application, tried it, and then never used it again).

These metrics can be used to identify high risk areas that can be addressed by testing Or development. For example, a high bounce rate may indicate usability issues. The active user rates can be used to develop realistic performance testing goals.

2.3. Determine Coverage Goals

Once the risks have been identified and assessed, the coverage goals must be determined. While the tester will need input from other people, such as the test manager or product owner, it is important to consider all the areas that need to be tested and get agreement across the team that the coverage goals are realistic and will accomplish the testing goals for the project.

The following areas should be considered and the desired coverage determined before testing starts on the project:

- **Requirements –** If there are requirements, requirements coverage should be used as one of the testing guidelines. Traceability from the tests back to the requirements is useful because requirements for mobile applications often change as new features are added and existing features are updated or modified. The traceability will help the tester identify which test cases need to re-executed when changes occur.

- **Risks –** The identified risks must be addressed by testing, and traceability may be needed between the test cases and the risk items.

- **Functions –** The capabilities of the software will be tested. They should also be tested in accordance with their associated risks. A complete list of functions will help to set the risk levels and help to track coverage of each of these items.

- **Code –** Because of the speed of development of mobile applications, unit testing is very important. Code coverage goals should be stated before development starts. Automated unit testing, particularly when employed with continuous integration and deployment, will help to improve the quality of future updates, as the same tests can be run each time without significant manual time and effort. Fault metrics and technical debt measures can be used to track the quality of the software.

- **Devices –** Coverage across devices must be known at the beginning of the project so that those devices can be procured or simulators can be bought or built. The developers must provide input regarding the expected variability between devices in order to determine which device behavior is representative. Device-based application testing is usually prioritized by the expected usage of particular devices with particular applications. Since it will not be possible to execute all test cases on all devices (and the permutations of those devices), allocating the test cases across the supported device configurations is an important risk mitigation activity.

- **Connectivity –** Coverage must include the way in which a device connects to the Internet (including cellular, Wi-Fi, Ethernet, and in some cases the ability to switch between them). This should also include access to any additional services (such as loading style sheets) and potential side-effects of network issues (such as latency, jitter and re-tries).

- **Geography –** The geographic location of expected use can influence the testing. If an application is expected to be used only at high altitudes, the test environment will need to take that into account. Devices that must respond despite intermittent or slow networks will be tested differently from those that will only be used in offices with highly reliable, fast networks.

- **User perspectives –** Designing good test cases requires a knowledge of the users, including their expectations, knowledge, capabilities, personas, and operational profiles (what they will be doing). Testing will need to simulate usage by the various expected users.

Understanding the coverage requirements for testing is important for setting the scope and timelines of the testing effort, as well as for determining the types of equipment and environments that will be needed.

2.4. Determine Test Approach

Once the coverage goals are determined, the proper test approach can be decided.

The test approach must consider the following:

- **Environments –** The tests must be conducted in certain environments and may have associated conditions (e.g., outdoors while raining).

- **People –** The product is intended for certain user types. The actions of those users must be built into the tests, including any variability based on the user (e.g., someone with poor eyesight may always zoom images to view them).

- **Industry context –** The target industry can influence the required test approach (e.g., safety-critical, such as medical devices and automobiles; mission-critical, such as those that an organization depends upon; Commercial Off the Shelf (COTS); games; business applications; social networking; Internet of Things; digital assistants).

- **Schedules –** The reality of the schedule must be considered when determining the test approach, with the highest priority (highest risk) tests being conducted first.

- **Scope –** The testing scope must be limited and clearly stated to set expectations for the coverage to be achieved and the risk mitigation goals.

- **Evaluation –** Evaluation of test results tends to be different for mobile projects because much of the non-safety-critical testing is performed with less structured techniques and with simulators and emulators. The evaluation method must be clearly stated and understood by the team members so that they will understand test status reports and the final test summary report.

- **Methods –** Testing methods vary for IoT and mobile device application projects.

Specific quality characteristics, environments and tools are discussed in Chapters 3, 4, and 5.

Depending upon the formality and criticality of the project, the test approach may be documented in a traditional test plan or may be informally documented in a brief project document. Either way, the approach should be documented because agreement for the approach is critical within the project team.

2.5 Identify Test Conditions and Set Scope

The test conditions are the building blocks of the testing to be conducted in an IoT or mobile device application project. There may not be enough time to create test cases in a fast-paced project. In this case, identifying the test conditions, assigning risk-based priorities to each and conducting testing to address each identified condition may be the most efficient method for testing within the limited timeframe.

Test conditions consist of the physical capabilities of the software within the device (e.g., buttons, icons, screen zooming, device rotation, geolocation); the functionality of the application (e.g., displaying an image, displaying a map, accessing a bank balance, summoning help when an airbag is deployed); and the non-functional areas, such as performance and usability. Each of these capabilities and features have a number of conditions that should be tested. Using risk assessment, these conditions can be prioritized for testing and the scope of testing can be set. For example, if the application will access banking information, the application will have a login capability.

To test this login, the tester needs to test a valid username/valid password, invalid username/valid password, valid username/invalid password, and so forth. Each of these combinations is a test condition. Since there can be many test conditions for a single feature of the software, it is important to identify the critical and high risk conditions to be sure those are tested. The low risk items may be left untested or may be tested as part of other tests.

Identifying and prioritizing the test conditions sets the scope for the testing. With limited time, priority/risk-based testing will ensure the most important items are tested to some level of coverage. When time runs out and the coverage is deemed sufficient, testing is complete.

2.6. Regression Testing

Regression testing for IoT and mobile device applications is particularly challenging. Not only does the software change rapidly (including the firmware), but devices are continually changing as well. As noted earlier in the example of digital assistants, changes are frequent and may not always be announced in time for developers to make appropriate changes to their IoT and mobile device applications and for testers to conduct the testing needed, including regression testing.

The more devices supported, the larger the set of changes. Regression testing should be conducted regularly for IoT and mobile device applications, even if the application itself has not changed. Changes in the environment may impact an IoT or mobile device in unexpected ways at any time. As discussed in this syllabus, test automation and access to device labs and simulators is critical to a successful IoT and mobile device application project and are required for good regression test practice. When regression testing is automated and devices are available (via labs or simulators), regression testing can be scheduled to run at regular intervals, such as once a week.

This requires the test devices and simulators to be updated regularly so that regression testing is reflecting the functionality of the software on the target devices.

3. Quality Characteristics for IoT and Mobile Testing - 290 mins

Keywords

geolocation, persona-based testing, perspective-based testing,Teststorming™

Learning Objectives for Quality Characteristics for IoT and Mobile Testing

3.1 Introduction

None

3.2 Functional Testing

CD-3.2.a (K3) For a given IoT and Mobile Testing project, apply the appropriate test design techniques

CD-3.2.b (K1) Recall the purpose of testing for the correctness of an application

CD-3.2.c (K2) Explain the important considerations when planning security testing for an IoT or mobile device application

CD-3.2.d (K2) Summarize the concepts of perspectives and personas for use in IoT and mobile device application testing

CD-3.2.e (K2) Summarize how device differences may affect testing

CD-3.2.f (K2) Explain the use of Teststorming for deriving test conditions

Copyright ASTQB,
All Rights Reserved

3.3 Non-Functional Testing

CD-3.3.a (K3) Create a test approach that would achieve stated performance testing goals

CD-3.3.b (K1) Recall aspects of the application that should be tested during performance testing

CD-3.3.c (K2) Explain why real devices are also needed when simulators are used for performance testing

CD-3.3.d (K3) For a given IoT and Mobile Testing project, select the appropriate criteria to be verified with usability testing

CD-3.3.e (K2) Explain the challenges for portability and reliability testing mobile and IoT and mobile device applications

3.1. Introduction

IoT and mobile device applications, similar to other applications, have functional and nonfunctional quality characteristics that must be tested. This syllabus covers the quality characteristics that are particularly important in the IoT and mobile device application testing scope. While not all of these are applicable to every IoT and mobile device application, each should be considered to ensure that nothing is skipped and that testing is prioritized correctly.

Copyright ASTQB,
All Rights Reserved

3.2. Functional Testing

3.2.1. INTRODUCTION

Functional testing is designed to assess the ability of the application to provide the proper functionality to the user. It tests what the software does. For IoT and mobile device applications, functional testing covers the following:

- Correctness (suitability, accuracy)
- Security
- Interoperability

Each of these is discussed in the sections below.

3.2.2. CORRECTNESS

Correctness testing is done to ensure the software provides the right functionality in a way that works for the user (suitability) and that the functionality is provided correctly, including all data delivery (accuracy). If the capability is there, but it is not delivered in a suitable way, the product may be unusable. For example, if a smart watch application cannot scale an image down to fit on the screen, it is not suitable. If it can scale the image, but it is the wrong image, it is not accurate.

3.2.3. SECURITY

While security testing is best left in the hands of the security experts, all testers should have some awareness of security vulnerabilities and areas that should be covered by testing. Some tools are available that can help with security testing, such as static and dynamic analysis tools, but good security testing requires a current knowledge of security issues, testing methods and tools, and the technical ability to create security tests (which often involve coding).

3.2.3.1. SECURITY IN IOT AND MOBILE TESTING

Security in IoT and mobile device applications poses more threats than traditional applications. The following should be considered when planning security testing or considering what should be tested:

- IoT and mobile device applications are generally more easily attacked by hackers than traditional applications. This is partly due to the lengthy communications over public networks and partly due to the tendency for users to download many potentially vulnerable applications which can expose other applications residing on the device. Another factor is that users tend to be lax in changing default passwords, which has led to attacks through devices as basic as home security cameras and baby monitors to highly critical devices, such as medical and industrial control devices.

- People are too trusting. They tend to download applications without concern, even though they would never open an email attachment from someone unknown. A great deal of personal information is kept on mobile devices, such as smartphones and tablets, because they are convenient and always available. Rather than recording passwords on a sticky note on a desk, passwords are often recorded in the notepad application on the device itself.

- Devices get lost or stolen. People misplace mobile devices frequently. This leaves the device open to tampering, particularly when it is protected by a single short password or pattern.

- Traditionally, mobile devices are often donated, sold or traded-in without the existing data being wiped. This provides a rich opportunity for the recipient to access all types of user data – passwords, user names, pictures, videos, contact information (e.g., name, phone, e-mail). As Internet of Things appliances are sold or donated, similar failures to wipe personal data will continue. Most IoT manufacturers do not provide instructions on how to properly dispose of private information on the device. A very common example of this is when someone connects their smartphone to a rental car and the user's contacts get stored on the vehicle for the next renter to see.

An important part of IoT and mobile device application development is to compensate for the lack of security knowledge on the part of the user. An important part of testing IoT and mobile device applications is to ensure that the security is in place and is working correctly. Testers need to make sure sensitive information, such as passwords or account information, is not stored unprotected on the device. While malware (hostile or intrusive software) will always exist, the application should protect itself from attack and the device itself should have some protection to validate installed applications.

While this list changes year to year, the following are the top ten mobile risks in 2016 according to [OWASP]:

- Improper platform usage
- Insecure data storage
- Insecure communication
- Insecure authentication
- Insufficient cryptography
- Insecure authorization
- Client code quality
- Code tampering
- Reverse engineering
- Extraneous functionality

Knowing the security risks helps the tester know what to test and can serve as a reminder for developers when they are coding.

3.2.3.2. SECURITY TESTING APPROACHES

Because security testing is often carried out by security testing experts, the tester may not need to know how to set up and complete security testing. It is however important for a tester to ensure they are covering the basics during their testing. This includes testing the following:

- **Access –** Ensure the right people and applications have access and those without permissions are denied access. Also ensure that access is limited to only the functions and data that the user should be able to access. This is similar to access security for any type of application.

- **Protecting data while on the device –** When data is stored to the device, it must be secure. This means that information, such as passwords, account information, credit cards and so forth, that are used in transactions are not stored in an accessible format on the device, even during the transaction.

- **Protecting data that is in transit –** Information is passed between the device and servers via the network (e.g., cellular, Wi-Fi) and passed between devices (e.g., Wi-Fi, Bluetooth, SMS). Any data that should be secured must be encrypted during this transition to protect it from interception and misuse. This includes transactions that may encounter errors or have to be re-tried.

- **Policy-based security –** Organizations may have security policies that indicate how data is handled and who may access it. When data is being transferred to/from a device or stored on a device, these policies apply just as they would with a non-mobile application.

- **Malware detection on the device –** A growing security concern, especially with IoT, is the possibility for increased levels of Distributed Denial of Service (DDoS) attacks since the size of the botnet could be much greater due to the proliferation of IoT devices.

As with any testing, it is important to understand the application and its uses. Security for a banking application will not be the same as that used for a memory game.

3.2.4. INTEROPERABILITY

IoT and mobile device applications must be tested for interoperability to ensure they interact properly with other components, devices and systems. These applications must be able to exchange information and images with other software. For example, an image or video captured by a home security camera can be sent via email to a smartphone.

Copyright ASTQB,
All Rights Reserved

Testing for interoperability is highly dependent on the capabilities and interactions of the application being tested. At a minimum, an application will likely transfer data back to a web server that maintains a storage of information (e.g., highest score achieved in a game, the current weather forecast). While applications may act alone on a device, they may also interact with other applications installed on the same device. Since new applications may be added at any time to a device and new IoT and mobile devices can be added to a network at any time, trying to test for the superset of interacting applications will likely lead to frustration. This is why a riskbased approach using a lightweight means to capture this information is a good way to approach the problem of too much to test in too short a time period.

Interoperability testing can be expanded to include verifying compatibility of the application across environments. An application may need to work on a variety of devices operating at different speeds.

This form of interoperability testing is sometimes called compatibility testing. One factor that makes compatibility testing of IoT and mobile device applications and mobile web sites so challenging is the number of browsers and versions supported by each application and device brand/type. It is important to know the list of devices on which the application is intended for use, so that a reasonable testing matrix can be developed and the testing can be divided between the devices, using techniques such as the combinatorial testing techniques. Sources of configuration information include web server logs, web analytics and store analytics (such as iTunes and Google Play) to see which percentage of users for a particular application use a particular device/browser.

3.2.4.1. DEVICE SPECIFIC CONSIDERATIONS

IoT and mobile devices are varied and have a wide range of capabilities. One of the factors in interoperability testing is understanding the commonalities and differences between devices. Specific versions of devices may have different capabilities that may affect an application's capabilities and usability. For example, an application running on a slower device (or one with less memory) may exhibit different characteristics than one running on a fast device. An anti-glare screen protector may render certain user interface designs difficult to read and/or access. This device specific information tends to change rapidly and should be determined immediately prior to testing. It may also be necessary to anticipate new device features so that testing can occur before the features are widely available (e.g., higher speed, bigger memory, different operating system requirements).

3.2.4.2. TESTING PERIPHERALS

It is also important to remember that devices may have peripherals attached or built in, such as scanners, card readers, bio-recognition equipment (e.g., fingerprints scanners), cameras, altimeters, microphones, speakers, and so forth. If the application may use a peripheral, or may be affected by the presence or absence of a peripheral, the application must be tested both with and without the peripheral. This is a consideration if simulators will be used for testing since simulators for peripherals may be needed. In the case of peripherals, actual device testing is usually required to some degree.

Copyright ASTQB,
All Rights Reserved

3.2.4.3. DEVICE DIFFERENCES

IoT and mobile devices have many differences, even within the same type of device, such as a tablet. For example, communication protocols may be different, transmissions may or may not be secured, the device may have the capability to be docked and transfer information. A device may be able to recognize other devices of its type when they are within a certain geographic area. The variability between devices and the commonalities they share all introduce testing opportunities. It is important to understand how an application will interact with a device or set of devices and how differences in those devices may affect the application. For example, a device may share geolocation information with the application, which then shares it with other applications and allows communication to other applications that are running on similar devices in the same area. If the geolocation information is secured on one device, but not on another, what will happen?

As devices add more and more features and more devices enter the market, these differences will become a larger factor in testing.

3.2.5. TEST DESIGN

When designing the tests for an IoT or mobile device application, the following should be considered:

- Functionality of the application
- Functionality of the device
- Risk within the subject domain of the application (e.g., a mobile device used to deliver medical information to ambulances)
- Network connectivity
- Operating systems
- Power consumption/battery life
- Type of application (native, hybrid, etc.)

The functionality of the application can be determined from the requirements, use cases, specifications or even by conducting exploratory testing to learn about the application. The functionality of a device, particularly if the device is made by another organization, must be determined by reading the published specifications, experimenting with the device or from talking with others who are familiar with the device. Designing tests for an IoT or mobile device application requires considering both the features of the application to be tested, as well as the capabilities of the device. The capabilities and features of the target device must be understood, particularly if those capabilities will be utilized by the application.

The following is a list of some of these capabilities, but remember the list is always expanding:

- Screen size and resolution for display (this can be very small in the case of watches and other wearables)

- Geolocation (ability to detect the device's geographic location)

- Telephony (ability to act as a telephone)

- Accelerometer (senses acceleration on three axes – up/down, side to side, back and forth – used for games and orientation)

- Gyroscope (senses orientation based on angular momentum)

- Magnetometer (measures the direction of magnetic fields – can act as a compass)

For example, if the application depends on using a magnetometer, the test design must include tests for devices with various types of magnetometers, as well as for devices without the capability.

In addition to the functionality, test design also needs to include installation of the application. Many applications can be installed over-the-air (OTA) which means that testing needs to include interruptions at any point in the installation, re-installation, upgrades and de-installation. Permissions and payment may also be required to install applications and must also be tested.

The risk of the application is assessed by the means discussed in section 2.2 - Identify and Assess Risk.

Copyright ASTQB,
All Rights Reserved

3.2.5.1. USING CORE TESTING TECHNIQUES

Standard black-box testing techniques are explained in the AT*SQA Essentials of Software Testing Syllabus [AT*SQA_EoST]. These techniques are applicable for IoT and mobile device application testing and are valuable in testing both applications and devices. Use of these techniques will help the tester ensure that desired test coverage is achieved. These techniques are briefly summarized here:

- **Equivalence partitioning (EP) –** Determine equivalences classes based on equivalent processing, then test one item from each class assuming the results for that one item are representative of the entire class. For example, assume all cameras with the same megapixel capabilities will create an image of the same quality.

- **Boundary value analysis (BVA) –** Select tests based on the boundaries of ranges of inputs or outputs. For example, test the maximum number of names that can be stored in a contact list, test maximum + 1, test one and test zero.

- **Decision tables –** Test combinations of inputs and/or stimuli (causes) with their associated outputs and/or actions (effects). For example, test that an incoming email results in the configured sound.

- **Exploratory testing –** Test by simultaneously designing and executing tests while learning about the application. For example, for a new application, use it to accomplish a single task and document any defects found.

- **Combinatorial techniques –** Test across different combinations of characteristics based on the information supplied by the model. For example, use pairwise testing to determine the combinations of devices and device features on which to test the application.

Copyright ASTQB,
All Rights Reserved

Some of these techniques, such as equivalence partitioning and combinatorial techniques, help to reduce the test set down to a manageable size. Exploratory and decision table testing help focus the tester toward real world usage and scenarios.

3.2.5.2. USING TECHNIQUES SPECIFIC TO IOT AND MOBILE DEVICE APPLICATIONS

In addition to core testing techniques, there are also specific techniques commonly used in testing IoT and mobile device applications. There are some overlaps between the two categories, but the best combination of techniques to use should be determined by the features of the application, the capabilities of the device that interact with the application, the criticality of the application, the time available, and the skills/knowledge of the tester.

The following techniques are commonly used in IoT and mobile device application testing: Session-based – These testing sessions are designed to be uninterrupted from start to end, reviewable by other testers or managers, and chartered to ensure the focus matches the goals of the testing.

Exploratory testing can be expanded into the concept of exploring different user perspectives, called perspective-based testing. Because the user base is so varied, it is important to consider the different perspectives those users bring to their usage of the device. The goal is to simulate real usage and concentrate on specific aspects of the software and its interaction with the device [Whittaker].

Copyright ASTQB,
All Rights Reserved

These perspectives and usage scenarios should cover the following:

- Skill level of the user (see persona-based testing below)
- Location of the user (e.g., indoors, outdoors, home, work, in a car, on a plane)
- Lighting in the environment (e.g., dark, bright sunlight)
- Weather conditions (e.g., rain, wind)
- Connectivity (e.g., strong, weak, intermittent)
- Accessories available (e.g., interaction with each accessory)
- Motion (e.g., stable, in hand while walking)

Scenario-based testing – Similar to testing according to defined use cases, this tests the paths that a user is likely to follow to perform a defined task. The validity of the scenario directly influences the effectiveness of the test. Testing scenarios that a user will not follow can result in wasted time that could be better spent testing frequently used paths.

Persona-based testing – Because the user base is so varied, it is important to consider the different types of people that will be using the software. People vary widely in skills, capabilities, and needs.

The following is a list of some of the personas that can be used for persona-based testing:

- First time user
- Casual user
- Frequent user
- Expert user
- Confused user (does not understand any of the software)
- Angry user
- Frightened user (afraid of technology)
- Impatient user
- Malicious user
- Playing user (experimenting with the software)
- Technically knowledgeable user
- Age class user (e.g., over 65)

Persona-based testing requires that tester to understand the viewpoint of the user and be able to interact with the software as that user. This can be a difficult task, particularly when the tester's level of knowledge is much higher than the user persona being tested. Good usability requirements will help to cover the software characteristics that are needed to accommodate the various user personas.

Because device and application usage can be so varied, it is often helpful to approach testing from a viewpoint that is independent from the requirements. For example, Teststorming [Rice] may be used to derive test cases and scenarios via brainstorming or mindmaps. This technique helps the tester to think of new and creative uses of the software that might not have been considered during the design and development of the product. As the usage and capabilities of IoT and mobile devices continues to expand, test case creation requires forward thinking, beyond what is stated in the requirements.

Reminder checklists are helpful when testing IoT and mobile device applications to ensure that all aspects of the software are being addressed. Since requirements for IoT and mobile device applications tend to be lightweight, the tester cannot rely on these documents as the only guide for testing.

3.3. Non-Functional Testing

Non-functional testing concentrates on how functionality is delivered to the user. This section focuses on the non-functional quality characteristics that are of primary importance in testing IoT and mobile device applications.

3.3.1. PERFORMANCE TESTING

In general, performance testing is verifying the response time of the system when it is experiencing a defined load. Performance testing also considers throughput and resource utilization. With IoT and mobile device applications, there are more considerations for performance testing (such as network connection type and strength, device type, device memory and other conditions) that may be difficult to control. IoT and mobile device applications are used for a wide variety of capabilities, but some applications have higher performance requirements than others. For example, GPS location applications that may be using the geographical position of the device to transmit driving directions have critical performance requirements in order to be able to adjust for changes to the planned course. Applications that are used for timecritical transactions, such as stock trading or online auctions, also have stringent performance requirements. Applications that provide media have requirements to provide consistent performance that is sufficient to provide a good user experience, without problems such as pausing during video display.

Performance criteria tend to be loosely defined and are often identified when a product does not meet expectations. Ideally, the performance criteria should be defined and captured early in the requirements phases of a project. Performance has to be architected into a product – it cannot be tacked on at the end. Defining these criteria, setting up the proper personas and benchmarks, and testing to ensure the criteria are met is critical for the success of a connected product. It might be acceptable for a refrigerator to be slow to respond to a request to decrease temperature, but it is unacceptable for a GPS-based guidance application to alert the driver of an excavator that a buried electrical line in its path too late for the driver to avoid it.

Another variable with IoT and mobile devices is the possibility of a lack of connectivity. This can significantly impact both functionality and performance. A poor connection can result in poor performance. A product may not be able to control the quality of the connection, but it can control its response to poor, intermittent or non-existent connections.

3.3.1.1. PERFORMANCE TESTING FOR IOT AND MOBILE DEVICE SOFTWARE

In addition to the normal performance testing that should be conducted regarding the server's ability to handle traffic, there are other considerations for IoT and mobile device application testing. Network connectivity plays an important part in the performance of an IoT or mobile device. Variables such as connection speeds, time required to connect and reconnect if disconnected and network latencies are all factors in the ability of the application to deliver the desired performance. Unreliable or inconsistent network connectivity may result in multiple re-tries or the application trying to proceed with some data loss. This can be particularly critical if the loss is of information the application needs to function.

Performance testing an IoT or mobile device application starts with ensuring that the application itself, and its interactions with the server and other needed resources, are as efficient as possible. A slow application or slow response from the server will only get worse when the application is running on an IoT or mobile device. In addition to the normal tests, IoT and mobile device application performance testing should also cover the following aspects of the application itself:

- Application launch time – This includes the time from the user's first indication that they want to use the application until the time the application is fully usable.

- User interface delays – This is the time that is spent between receiving a user interaction (e.g., pushing a button, moving an image) until the time the application provides the response to the user.

- Irregular performance – This is a problem when the performance is noticeably varied for the same type of transaction.

- Visual indicators – As with any user interface, the user must be given an indicator that processing is occurring and there is a wait. Since these indicators usually appear after a certain length of waiting time has occurred, it is important to test that the indicator appears correctly, even when performance is degraded.

Copyright ASTQB,
All Rights Reserved

- Resource usage – An IoT or mobile device application is likely running in a shared environment. This makes efficient usage of CPU, memory and battery important, not just for the application but for the overall device as well. IoT and mobile devices are rarely running just one application. Performance must be verified with an expected set of other applications and processes running. This will help to verify what the real user experience will be when running the application under test.

- Degraded performance with resource constraints – In the general application world, “graceful degradation” is usually acceptable when memory or some other critical resource is constrained. For IoT and mobile devices that may be dependent on battery function, degraded performance may not be acceptable.

  For example, a pacemaker cannot degrade the number or strength of signals sent to the heart. It must anticipate battery longevity and provide proper notifications when connected.

- Task completion – This is an overlap between usability and performance characteristics. It is important to test the time it takes for a user to complete tasks that they would expect to accomplish with the application.

- Code inefficiencies – While the code contained in an IoT or mobile device application may be compact, it is still possible for the code to have bottlenecks, endless loops or other inefficiencies that impact overall performance. These inefficiencies can be detected by both static analysis (reviews and/or tools) and dynamic analysis (tool-driven).

Copyright ASTQB,
All Rights Reserved

In addition to these, web applications also need to be tested for:

- **Site page loading time –** In the case of a mobile application running from a web site, the time it takes to load the site pages can have a significant impact on the user's experience with the application.

- **Delays –** Delays can occur for many reasons, particularly those related to server and network time.

- **Resource usage –** While a web application will take less memory and CPU from the device, it may require more network bandwidth from the device because of larger amounts of data being passed back and forth from the server.

3.3.1.2. PERFORMANCE TESTING APPROACHES

Identifying valid personas is important for both performance and usability tests. Personas define the characteristics of the user as well as the tasks the user is trying to accomplish. By using this information, scripts can be built that can simulate typical user transactions. These personas can be duplicated by automated tools to provide multiple virtual users who will interact with the system through a device in a prescribed way. Since devices may be difficult to procure, device simulators are often used to supply the same interaction as the device without requiring the actual device to be present during the testing.

Performance goals and testability goals must be established when the application is being designed and the target environments are being specified. These goals can then be used to compare against the actual results of the testing. Performance testing can start as soon as individual components are available. By building and executing performance tests as the application is being developed, particularly in a continuous integration environment, poorly performing components can quickly be identified.

The environment will always have an impact on the application's performance. A device that is inherently slow due to poor design, high overhead, slow communication or any other factor will cause the application to appear to perform slowly as well. When testing with simulated devices, it is important to include a set of real devices to ensure the simulated performance is reflective of the performance that will be experienced by a real user on a real device. It is also important to understand what is being tested. An IoT or mobile device application that is deployed on a user's device has different performance concerns compared to a web application that is accessed from a mobile device. Understanding these differences before designing performance tests will help ensure valid results and maintain proper focus of testing.

Overall poor performance may cause people to abandon the application. What is "slow" is determined by the user. User expectations are continually being refined so that yesterday's fast performance may be unacceptably slow tomorrow. However, some responsibility for performance rests with the user. For example, older devices and operating systems will have slower performance. In addition, having many apps running concurrently will degrade performance.

3.3.2. USABILITY TESTING

IoT and mobile device applications tend to have a wider and more varied user base than traditional applications. This is partly due to the ease of access to these applications and partly due to general acceptance that applications should be available anywhere, anytime. This poses significant usability concerns, particularly on the part of those planning and conducting usability tests.

3.3.2.1. USABILITY TESTING FOR IOT AND MOBILE DEVICE SOFTWARE

In addition to traditional usability areas such as navigation, colors, sounds, accessibility, and others, IoT and mobile device applications have some additional areas for testing. These include:

- **Simplicity –** IoT and mobile device applications must be designed for simplicity and ease of use.

- **Layout –** Interaction with IoT and mobile device applications is sometimes done via a device, such as an electronic pointer or pen. However, many devices require the use of fingers. This means the application must allow space for fingers and perhaps for displaying a keyboard that is large enough to be used for text entry as well. This can be especially important and challenging in devices such as wearable devices.

- **Intuitiveness –** Users expect to be able to load an IoT or mobile device application and immediately use it. They may experiment with it for a little while, but they will make a quick decision as to how intuitive it is to use. If it is too difficult to use, most users will discard the application and look for another. If instructions will be displayed to the user, they must be visible, but not intrusive.

Copyright ASTQB,
All Rights Reserved

- **Navigation –** While navigation is a concern with traditional applications as well, it is even more important for IoT and mobile device applications. The user has an expectation to be led in the direction they need to go rather than having to determine their path from a list of many options. IoT and mobile device applications are expected to be simple and easy to navigate.

- IoT and mobile device applications, more so than traditional applications, will experience a high rate of abandonment if they are not considered usable by their users. If an IoT or mobile device is limited to using certain applications from a particular vendor, then the risk of application abandonment may be eliminated, but the risk of device abandonment may be increased.

3.3.2.2. USABILITY TESTING APPROACHES

Similar to performance testing, personas are needed for usability testing to ensure testing is covering a representative set of user types. It is important to remember that users are not only end users. Applications are sometimes used to increase sales for a company. If a particular campaign has been used across the mobile applications for a company, their sales team may need to see metrics regarding number of downloads, number of responses, and other information.

User expectations are an important consideration. This is an area where expectations will change as more applications become available, new and improved devices are introduced and speed is improved. Usability experts will be challenged to stay current with market expectations for product usability.

When approaching usability design and testing, real users are needed. Observing users actually using the application will help target the testing to cover real usage scenarios and may also highlight areas where the interface is confusing or navigation is unclear. If possible, obtaining user feedback is also helpful for understanding what they like about the application and identifying areas where improvement would be needed. Usability labs and surveys may be used to help obtain this information in a controlled environment.

Simple metrics can help bring objectivity to the subjective topic of usability. Measuring items such as the following can be helpful in understanding usability factors:

- The number of tasks fully completed
- The time taken to perform a complete task
- The mistakes made in performing a task or series of tasks
- The number of clicks (actions) needed to perform a task

These measures can also be obtained by first, second and third attempts, which can then be used to determine the learnability of the application.

Effective usability testing can be conducted with a small set of representative users [Nielsen]. If the users fit the identified personas, they can provide valuable feedback relatively inexpensively, which can be used to improve the application and future designs.

With IoT and mobile devices being used by a wide range of people, accessibility must be considered during testing. If compliance to accessibility standards is required, it is important to obtain tools that can scan the application for compliance or to conduct the manual testing necessary to ensure compliance. Accessibility considerations include items such as readability, color usage, sound usage, human interaction such as typing, ability to resize the screen or change contrast, and so forth. It is also important to consider accessibility in terms of environmental challenges, such as bright sunlight, darkness, rain and so forth.

Mobile devices, particularly smartphones, are feature rich. This feature set is sometimes dependent on accessories to the base mobile device. These include cameras, scanners, credit card readers, headphones, keyboards, and other devices that can be built into or attached to the base device. When testing an application, it is important to consider any accessories that might interact with the application or the environment shared by the application. Consideration must be given to concurrent processing when using accessories. For example, a credit card reader might supply input to a banking application running on the phone, but that card reader may not work when the camera is also in use. These types of interactions between accessories must be considered when selecting test device configurations.

3.3.3. PORTABILITY TESTING

Portability testing focuses on how well an application will function when moved into a target environment. Good portability testing requires good understanding of the target environments and their characteristics.

3.3.3.1. PORTABILITY TESTING FOR MOBILE SOFTWARE

Mobile software is intended to run on a mobile device. Devices are plentiful and the numbers, types and capabilities continue to increase. Mobile testing is usually concentrated on a subset of representative devices intending to cover the most common environments and

Copyright ASTQB,
All Rights Reserved

environment variables that could affect the application under test. The key to good portability is good design. It is important for the tester to work with the developer in determining areas to be tested due to device differences.

For example, a developer is usually aware of modifications that were required for an application to work on both iOS and Android devices. That being said, developers will sometimes miss nuances between devices and between different versions of similar devices. Obtaining a good set of representative devices can be split between procuring some actual devices and using accurate simulators for other devices.

3.3.3.2. PORTABILITY TESTING APPROACHES

Users do not usually have an awareness that an application may have to be modified to work on different devices. This results in an expectation from the user that a mobile application will work on a set of devices and that they can expect the same level of usability from the application on a smartphone, a tablet and a PC browser. This may not be a realistic expectation, but it is often a tester's job to verify which environments enable the application to function and work well.

Portability testing for the present requires testing across a known set of devices and software versions. Portability testing for the future requires anticipating which devices will still be popular, what potentially conflicting or limiting features they may have and how a user will expect to use them. Because the field of mobile devices changes so rapidly, it is important for a tester to understand the new devices and the planned release dates for major new changes. Procuring updated hardware devices can be expensive, however, simulators tend to lag behind the introduction of the physical device. As a result, the market is sometimes testing an existing application on a new device before the test team has had a chance to test it.

Developers must do their best to future-proof their designs. Device manufacturers will normally maintain a level of backward compatibility (meaning that software that ran on previous versions will still run on the new version). However, there is a short life span for mobile devices and old releases will fall out of support more rapidly than is seen with desktop software.

Portability testing must consider whether the application is using a native device interface or a more portable interface. An application using a native device interface will likely encounter portability problems when the application is installed and run on a different type of device. An application with a more generic interface is designed to be ported to different devices.

Software that has been ported to another environment will sometimes exhibit performance and usability differences in the new environment. Any porting project must consider additional performance and usability testing to ensure an acceptable level is still achieved. For example, a mobile application that displays flight information may be quite readable on a tablet, but may not scale correctly or allow resizing on a smartphone. Similarly, software that was written to be fast on a specific device may be quite slow on a different device because it was not optimized for that environment.

3.3.4. RELIABILITY TESTING

IoT and mobile device applications have become an important part of both business and personal life. As people become more and more dependent on them, reliability becomes an important quality characteristic. Some applications are safety-critical and require extremely high reliability. Reliability equates to the robustness of the software, including how well it handles faults (fault tolerance), how quickly it can recover from a problem, and how consistent it is in providing the same result for the same actions.

3.3.4.1. RELIABILITY TESTING FOR IOT AND MOBILE DEVICE APPLICATIONS

Testing for reliability requires causing failures and verifying that the software correctly detects the failure and either handles it or recovers gracefully. It is important that IoT and mobile device applications are able to reconnect when connections are lost and continue processing without losing any transactional data. Mobile devices, by definition, move around. They go to places with poor network connectivity. They go in tunnels and underground. They go on airplanes. At a minimum, mobile devices go everywhere people go and this creates a large set of potential reliability issues.

IoT and mobile device reliability testing must be concerned with such things as temperature tolerance, impact, submersion, extreme heat and cold, and so forth. The applications running on the device must be able to respond to the effects of these conditions on the device, including device failure.

Copyright ASTQB,
All Rights Reserved

3.3.4.2. RELIABILITY TESTING APPROACHES

IoT and mobile device applications must be tested for the same reliability issues as any other application. This includes insufficient memory or other resource constraints, hardware failure, and network or communications failures.

In addition to the traditional reliability tests, testing must also include the ability of the application to handle low battery levels, shutdown conditions, complete power failures and similar power related issues. Because mobile devices usually run on battery power, power failure poses a risk with a higher likelihood than would be expected for a traditional application. Similarly, issues with network connection, disconnection and reconnection (including switching from cellular to Wi-Fi) must all be tested because of the high likelihood of occurrence. Mobile devices tend to be shut down less frequently than traditional computers which results in a longer period of operation. This can allow problems such as memory leaks to become more apparent.

Reliability can also be measured by determining how long a mobile application can operate continuously without failure (or without recharging the battery). This can be performed by creating and performing simple automated tests and measuring the mean time between failures (MTBF). When failures occur, they can be captured in reliability failure scenarios. These scenarios can be provided to the development team to help them design stronger mobile applications that will prevent or handle the defined failures. It will also allow the developers to create recovery procedures if the failures do occur.

4. Environments and Tools - 285 mins.

Keywords

emulator, native device, simulator

Learning Objectives for Environments and Tools

4.1 Tools

CD-4.1.a (K1) Recall the expected capabilities for mobile application testing tools

CD-4.1.b (K2) Explain the use of generic tools in testing IoT and mobile device applications

4.2 Environments and Protocols

CD-4.2.a (K1) Recall the sources of data for an IoT or mobile device application

4.3 Specific Application-Based Environment Considerations

CD-4.3.a (K2) Explain the differences between browser-based and native device applications

4.4 Real Devices, Simulators, Emulators and the Cloud

CD-4.4.a (K2) Explain why testing is not conducted entirely on real devices

CD-4.4.b (K3) For a given IoT and mobile testing project, determine how and when to use simulators/emulators during testing

CD-4.4.c (K1) Recall how to verify the reliability of a simulator/emulator

Copyright ASTQB,
All Rights Reserved

CD-4.4.d (K3) For a given IoT and mobile testing project, determine how and when to use cloud-based testing

4.5 Performance Test Tools and Support

CD-4.5.a (K2) Explain how the cloud can be used to support performance testing

CD-4.5.b (K2) Explain the types of data a performance tool needs to be able to create and track

4.6 Test Automation

None

Common Learning Objectives

The following learning objective relates to content covered in more than one section of this chapter.

CD-4.x.a (K3) For a given IoT and mobile testing project, select the appropriate tools and environments for testing

---

4.1. Tools

The IoT and mobile device and application market is expanding rapidly. Fortunately, the testing tools are keeping up with the explosion. Tools for testing IoT and similar IoT and mobile devices are primarily focused on networking, performance and security, as opposed to functional testing. The tester is now confronted with trying to select the best tool from a large set of changing tools with varying capabilities and reliability. This section will better prepare the tester for understanding the tool options and provide information to help select the best tool for the specific application and environment.

4.1.1. APPLICATION TO IOT AND MOBILE DEVICES

There are plenty of test tools on the market, but it is important to discern between tools that provide general testing support, those that support IoT and mobile devices in general, and those that are specifically aligned to testing mobile applications. In general, tools that are focused on mobile testing should be able to do the following:

- Adapt to different environments and protocols
- Simulate a native device
- Support testing across iOS, Android and other operating systems
- Simulate multiple users simultaneously
- Support mixed and varying locations of devices
- Support or simulate networks of varying speeds and quality
- Simulate or provide connections that can be disconnected and reconnected

When seeking a tool, the tester must understand the capabilities of the application, the environments to be supported, and the testing requirements in order to select the tool most suited to their needs. Because the market is changing so quickly, it is important to use good tool selection processes and ensure the vendor has created and will support a quality product. Pilot tests are needed to ensure the tool will work in the specific environment. Even if the purchase price of a tool is low (or free) the organization will still invest a considerable amount of money to implement and use the tool. Proper evaluation processes need to be followed for any tool procurement, even a low cost one.

Tools should be evaluated for ease of use. As the IoT testing tools increase in the market, the usability will become better. Sometimes it makes sense to wait for a later version of a tool if it will have a vastly

Copyright ASTQB,
All Rights Reserved

improved interface. As with any tool, it is important to remember the skill sets of the tool users and ensure the tool will provide the necessary functionality in a way that is accessible to the tester.

4.1.2. GENERIC TOOLS

Generic tools are still useful when testing IoT and mobile device applications. For example, test management tools, defect management tools, and requirements management tools are all still needed. Build tools, continuous integration/deployment and unit testing tools are also needed to support the development process. Since many IoT and mobile device applications also have a backend component, tools used in the testing of the software running on application servers, web servers, and database servers are still needed too. In general, the background or supporting software will be tested in the same way for IoT and mobile device applications as it would be for client or web applications.

4.1.3. COMMERCIAL OR OPEN SOURCE TOOLS

Commercial tools are those made by a company, for profit. While these may be considered by some to be more reliable, they tend to lag behind the market needs.

Open source tools are created by interested individuals or communities who have created a tool for a specific need. Open source tools tend to be focused on solving a particular problem whereas commercial tools are designed to address a wide range of capabilities. In the IoT and mobile device application testing world, open source tools are readily available with a varying focus and capability set. Commercial tools are also available but may lag behind in adding coverage for new capabilities and technologies. Before selecting either type of tool, due

Copyright ASTQB,
All Rights Reserved

diligence is required to ensure it is the most appropriate tool for the job. It is important to consider tool support, maintenance, applicability and ability to grow with the industry, as well as cost and usability.

4.2. Environments and Protocols

4.2.1. ENVIRONMENT CONSIDERATIONS

An IoT or mobile device application is expected to work in numerous environments. Each of those environments can have its own particular features. At a minimum, the following areas need to be considered when test environments are being selected and established.

4.2.1.1. CONNECTIVITY

Devices are able to connect via Wi-Fi, cellular networks and may talk to each other via Bluetooth technology and various other means. When testing the connectivity, it is important to test for disruptions in the connection, reconnection capabilities, the ability of the application to continue when data loss has occurred during the transmission, and lost connections. It must also be verified that the device can switch between connections without losing data or impacting the user. For example, a device may switch from Wi-Fi to cellular, or may switch from a slower 3G network to a faster one.

When the IoT and mobile devices are mobile (as opposed to IoT appliances), they should be selecting the best connection available. Testing the various connectivity options requires significant network resources and configuration capabilities.

4.2.1.2. MEMORY

Device memory varies widely. Tablets, smartphones and other devices come with a range of memory options. New devices generally start on the market with a lower memory capacity and increase capacity as newer models are introduced. Devices on the market may also be able to add more memory. It is usually a safe assumption that if an application fits within the memory on an existing device, it will still fit on future versions of that device that have greater capacity. New devices support new features and peripherals which may be competing for memory usage, but that is no different from the memory competition that will always occur.

Testing for efficient memory usage (efficiency testing) and ability to handle low memory situations (fault tolerance) is important for IoT and mobile device applications running in a shared environment with competing processes. Memory management must be monitored to ensure no memory leakage is occurring due to allocated memory not being released, and that no memory corruption is occurring.

4.2.1.3. PERFORMANCE

The performance of the test environment is an important consideration for creating valid test results. The performance of the test environment should mimic the performance of the production environment, including communication interruptions, reconnections and network traffic. Because communication is at the core of performance for IoT and mobile device applications, the test environment must provide a realistic communication interface, including an ability to introduce and control the problems that are likely to be encountered in production such as weak connections, timeout errors, and so forth.

Copyright ASTQB,
All Rights Reserved

4.2.1.4. DEVICE CAPABILITIES AND FEATURES

IoT and mobile devices vary considerably in their capabilities and features. While application testing might not include testing all the capabilities of the device, it is important to understand how the features of the device may interact or affect the application. For example, if the application requires the use of an accelerometer and gyroscope to determine the orientation of the device to display the application interface correctly, testing must include devices with various versions of these features (any of which may have different interfaces), as well as devices without the features or with malfunctioning features. The more features of the device used by the application, the larger the testing pool of devices needed.

Features and capabilities to be considered when determining the proper test environment for the application include:

- Screen size for display
- Screen lighting
- Geolocation
- Telephony
- Accelerometer
- Gyroscope
- Magnetometer

In addition, the capability of a device to install the application over-the-air (OTA) must be considered when determining how to download releases and updates to the device. The OTA capability may limit the size of the downloads and the installation routines must have strong recovery from interruptions, partial downloads and missed updates. An update must not adversely affect the functioning of the device or result in data loss.

Copyright ASTQB,
All Rights Reserved

As was mentioned in the techniques section, combinatorial testing techniques (such as pairwise testing) can help reduce the potential number of test environments by determining representative combinations of capabilities and features that do not interact. Decision tables can be used to determine necessary combinations of capabilities and features that do interact.

4.2.1.5. DATA HANDLING

Another environmental consideration is the data that will be used by the IoT and mobile device application. In general, data used by an IoT or mobile device application can come from one of five sources:

- The backend system (e.g., database servers)
- The device itself (e.g., GPS location, internal clock)
- The user (via some type of input e.g., text, selections, UI interaction)
- External sensors (e.g., motion and temperature detectors)
- Another IoT and mobile device (e.g., a PC or mobile device connected via USB)

Testing for IoT and mobile device applications must consider receiving data, sending data, and storing data on the device. Data must be handled with the appropriate level of security and reliability. When data is stored on the device, special care must be made to ensure the data is safe and protected from unauthorized use.

4.2.1.5. DEVICE LOCATION

When testing IoT and mobile device applications, the tester has to consider access to the physical or simulated device. Since IoT and mobile device application testing often requires testing across a number of devices, access to these devices must be determined during the test planning phase. There are a number of ways to access multiple devices and, in some cases, multiple testers on multiple devices. The following list includes some of the more common approaches to procuring a large number of devices for testing:

- Actual physical devices co-located with the testers (e.g., a test lab with IoT appliances or a test range which would be used for automotive testing)
- Open Device Labs [OpenDeviceLab]
- Crowdsourcing (e.g., utest.com)
- Remote device labs (e.g., from the vendor or an external organization)
- Virtual environments (e.g., simulators, emulators and cloud-based)

Acquiring a large set of devices and keeping that set current with new models and versions can be cost-prohibitive. As a result, using a remote test environment furnished by tool vendors or virtual environments is often a more feasible approach.

4.2.2. PROTOCOLS

Different devices may use different communication protocols. Testing tools, particularly load testing tools, may not support all protocols. The tester must understand what protocol is used by the devices to be tested to ensure there are no incompatibilities with the tools that will be used during testing. If simulators will be used, the simulator must be able to simulate the use of the device's protocol.

4.3. Specific Application-Based Environment Considerations

4.3.1. BROWSER-BASED APPLICATIONS

Browser-based applications are designed to run on a set of supported browsers that run on the device. Some of these may give the user the option to switch to the mobile version of the application or stay with the standard web application. One advantage to creating browser-based applications is the built in portability. Any device that can run a browser should be able to run the application. This should, in theory, limit the testing requirements to testing with the various supported browsers on any type of device. This is sometimes helpful in gaining portability with mobile applications and devices. There are, however, specific considerations for testing when a browserbased application is also considered to be a mobile application. For example, different browsers and versions vary widely in how they handle certain components, java script, and cascading style sheets (CSS). Some plug-ins may not be available on a mobile device (e.g., Flash, in which case a special Flash-enabled browser is required).

4.3.1.1. CONSIDERATIONS FOR USABILITY AND PERFORMANCE

Because these applications may not have been optimized for an IoT or mobile device, there may be issues with usability and performance. In particular, font sizes, navigation and screen layout may not be user-friendly when viewed on a smartphone and even worse on a watch. While many of these applications have an option to switch to the “mobile” version, that option may not be apparent to the user who is confronted with tiny fonts and only a portion of the main window.

Copyright ASTQB,
All Rights Reserved

Performance may also be an issue for an application that was not designed for the IoT and mobile device environment. As was discussed in the performance testing section, the tester will still need to test for performance but should also watch for connectivity issues that might impact reliability and performance.

4.3.1.2. BROWSER VERSION SUPPORT

As with any web application, the application must be tested with the supported browser versions. When the application is also intended to be deployed in the IoT and mobile device world, additional browser versions may be required and, more importantly, the frequency of use of some browsers may differ from the traditional platforms. For example, the PC environment may see a dominance of one type of browser where mobile devices may use a different one. When this happens, testing prioritization must shift proportionally to the more frequently used browsers.

4.3.2. NATIVE DEVICE APPLICATIONS

An application that is built for a native device is generally using specific features and capabilities of that device and its operating system to deliver functionality. When this happens, the same application is generally not portable to another environment without modification. There are a number of reasons for developing native device applications, such as the desire to utilize the hardware capabilities of a certain device or the need for an IoT device manufacturer to develop native apps for its proprietary hardware. In addition to being able to take full advantage of the device capabilities, the developer is also able to tune the user interface to a more specific market. This approach is also used for devices that do not have the capability to run a browser or those that must work without Internet connectivity.

Copyright ASTQB,
All Rights Reserved

4.3.2.1. GOOD SIMULATOR OR THE REAL DEVICE

With native device applications, the simulator must be designed specifically for that device. If simulators are not available, then the testing must occur on the real device. This, of course, may become costly if the device is difficult to obtain or performance testing is required for a large number of the devices. The tester should work with the developer to understand what aspects of the application require the specific device and what could possibly be tested with a generic simulator.

4.3.2.2. TOOL SUPPORT

Depending on the market popularity of a native device, tool support may not be available or may not be available soon enough for the testing of new applications. Tool support will follow the market. In general, tools will be developed first for the major market players, such as iOS and Android. New devices may not have the tool support needed to procure simulators, conduct performance tests or to support test automation.

4.3.3. HYBRID APPLICATIONS

A hybrid application may use specific features and capabilities of a device via plugins. Because device compatibility depends on the framework used to develop the application, testing across different devices must still be done as the framework may not be defect-free. Generally, all the testing needed for native applications and for browser-based applications is needed for a hybrid application.

Copyright ASTQB,
All Rights Reserved

4.4. Real Devices, Simulators, Emulators and the Cloud

Testers, working with developers, must determine the most appropriate test platform. This can be real devices, simulators of devices, emulators, or a mix of the three.

4.4.1. REAL DEVICES

By definition, real devices are the most realistic, providing the most accurate test environment. Real devices will provide the production environment and will allow the tester to observe any device specific issues that might be missed with a simulator.

Devices have hundreds of differences between each other. They also have their own defects [Firtman], such as inconsistent sensitivity to user input (e.g., not detecting pressure at the edge of the screen) or physical issues (e.g., buttons that are not reliable or consistent). Simulators will always lag behind the real devices because it takes time to build and test quality simulators.

Real devices, although the best test platforms, may be difficult and expensive to obtain. It may also be difficult to create test automation and performance testing that will work with real devices, however, there are a growing number of solutions to allow test automation of multiple real devices driven by a single driver computer. Generally, it is best to test with a mix of platforms, using the real devices as a sample set for usability, performance sampling, and general functional tests. Real devices are often used for comparison against the simulated devices to ensure the simulators are giving “real” responses.

Copyright ASTQB,
All Rights Reserved

Usability testing should always be conducted on the real device in order to give a proper and full assessment of the user experience. Device differences can affect usability and even though the application is the same, it may have a different usability level on different devices.

Depending on the application and the target platform, it may be possible to test with devices that are similar in capability to the target. This is sometimes done when new operating system versions are available or new features are available on new devices, but are not leveraged by the application.

4.4.2. SIMULATORS

A simulator is a program that simulates some aspects of a device. It does not emulate the hardware itself and may not simulate all the device responses and activities. It does not work on the same operating system as the device.

Simulators are sometimes supplied by device manufacturers to help developers test applications. Since a device will be more popular if it has many applications, it is in the best interest of the manufacturer to supply a good simulator. However, simulators are not always good or reliable and may not be a good representation of the real device.

4.4.2.1. BUY OR BUILD

If a reliable simulator is not available, the development organization may choose to write their own. While this will enable them to create exactly what they need to develop and test the application, the simulator itself must be tested. Building a simulator is a development project in itself and requires analysis, design, architecture, development, testing and, of course, documentation. Furthermore, a simulator must stay current with changes made to the real device or testing with the simulator becomes invalid.

4.4.2.2. VERIFY SIMULATOR RELIABILITY

Before relying on the simulator, it is important to ensure that the simulator is giving correct responses to inputs. This is done by providing the same inputs to both the real device and the simulator and verifying if the results are the same. If not, the simulator is not reliable. This is true for functionality as well as performance. A slow simulator that is functionally correct (or a fast one) can still be used for functional testing, but further testing will be needed on a real device to ensure that the response time differences do not affect the application.

4.4.2.3. USING SIMULATORS FOR PERFORMANCE AND LOAD TESTING

Simulators are commonly used for load generation and performance evaluation. Because simulators are software rather than hardware, large numbers can be created and run without additional costs of procurement. As with any performance testing, the tester should be sure the activities per simulator are equal to the expected activities per real device in order to create accurate load and performance reports.

4.4.3. EMULATORS

Emulators are used to provide the functionality of the device itself, including software, hardware and operating systems. This is necessary for certain applications that may use various device components, such as cameras or special screen controls. Emulators are usually written by device manufacturers although some are available from other sources. It is difficult to test an emulator for proper functioning without knowing the internals of the real device. If an emulator is to be used, the tester should ensure that it is from a reliable source and that it has been thoroughly tested. Spot checking emulator responses against real device responses is a good idea and will help verify that the version of the emulator being used corresponds to the target device.

4.4.4. CLOUD

There are a number of cloud alternatives for IoT and mobile device application testing.

These include the following:

- **Cloud hosted appliances –** Appliances or devices exist in the cloud and can be accessed via manual or automated test. This allows access to many different types of devices. These devices can be used for functional testing, as well as performance and usability testing.

- **Cloud hosted agents –** Software can run in the cloud that simulates users from all over the world. This allows a site to verify what happens to its backend when many users of mobile devices from many types of networks use the application. This is sometimes performed with device simulators in the cloud and sometimes performed with real devices in the cloud.

- **Cloud network simulators –** When performing testing in the cloud, network simulators can be used to simulate various network configurations, speeds and error conditions. This allows a realistic test environment to be created with varying network types.

- **Cloud protocol simulators –** Since devices may communicate via different protocols, the protocol simulators are used to simulate those protocols. This allows an organization to test their application with varying protocols or to conduct performance testing across multiple protocols.

Any cloud solution for testing must consider the reliability of the cloud environment, the realism of the environment (which often is determined by the configurability) and the accessibility. In addition, there are some security concerns with using cloud environments for testing, particularly for new and innovative applications and devices

4.5 Performance Test Tools and Support

Just as IoT and mobile devices have exploded into the market, so have the tools to support their testing, particularly for performance. Cloud solutions allow access to a large number of devices, networks and protocols, allowing load to be developed on a system from a variety of cloud-based devices or device simulators. However, a good tool by itself is not enough to conduct effective performance testing. Performance testing must be targeted to the operational profiles that matter and the loads that are expected to be experienced by the system, as discussed in Section 3.3.1.

Copyright ASTQB,
All Rights Reserved

When selecting a performance testing tool, the tester should consider the tool’s ability to create and track the following:

- Information flowing between the device and the servers, such as
- APIs
- Transaction data
- GPS information
- Images
- Volume and frequency
- Connection/disconnection patterns
- Activity bursts
- Usage patterns
- Variances for time of day/week
- Uses for business devices versus personal devices
- Peak usage times (e.g., daily, seasonal)

The ability of the tool to control and monitor this information is critical to the success of the performance testing effort. Testing with the wrong tool can result in wasted time or, worse, incorrect results.

There are many good commercial tools available for performance testing. Some traditional tools are adding support for mobile devices. New, targeted mobile performance testing tools are entering the market frequently. When making a tool decision, it is important to assess the capabilities needed now and in the future. It is good practice to plan for success and assume the application’s user base will grow dramatically, much faster than with traditional software.

Copyright ASTQB,
All Rights Reserved

4.6. Test Automation

Test automation is not optional in IoT and mobile device application testing; it is a requirement. As more tools are joining the market, test automation is feasible and maintainable only if it is well designed and planned. IoT and mobile device application test automation projects must consider the following:

- Base the tests on realistic usage patterns
- Understand and test the interactions between the user and the device
- Understand and test the interactions between the device and the servers
- Isolate the data from the test automation script by using data-driven or keyword-driven approaches
- Ensure the ability to control the real device (or simulator)
- Develop for maintainability
- Version control test cases so that older versions of test automation are available to check a maintenance release

Test automation is vital for IoT and mobile device application testing and requires good planning, solid design and careful implementation. Maintainability must be designed into the test automation code and good practices (such as isolating the data) will help ensure maintainability. Because IoT and mobile device applications change so rapidly, maintainability is an even higher priority in IoT and mobile device application test automation than it is for traditional applications. Test automation can be as valuable as the software it is testing. Version control, good coding practices, and quality requirements apply to the test automation code. Ideally, test automation is built as the application is built, enabling automated testing of the continuously integrated and deployed application.

Test automation with simulators (or emulators) is the more common approach and requires simulators to be built to support automation by providing the proper interfaces to allow the automation to communicate with the simulator. Since a simulator may not have buttons to push or a screen to touch, it needs to provide a capability for the test automation to send commands that would accomplish the same result as a user interacting with a real device. There are tools on the market that drive automated tests to actual devices, but cost may be a prohibitive factor for smaller development organizations and using simulators may be the more practical solution.

4.6.1. TOOL SUPPORT

As with performance testing tools, test automation tools for IoT and mobile device applications are also rapidly entering the market. Careful tool evaluation is required, but it makes sense to look to new tools with more finely tuned abilities for IoT and mobile device applications rather than some of the traditional test automation tools that adapt more slowly to the new markets. Because IoT and mobile devices and their applications will continue to evolve rapidly, any test automation tool must be able to adapt as well.

Test automation is a significant investment and buying the wrong tool can be a costly mistake.

4.6.1.1. PICK THE ENVIRONMENT

Tightly coupled with proper tool selection is proper environment selection. Test automation must be targeted to an environment. That environment may include:

- Real devices
- Simulated/emulated devices
- Cloud hosted devices or user agents
- Combination of any of the above

Focusing the test automation on the target environment will help the tool selection process and will ensure that the tool will be able to support testing as the testing environments expand. Even if a cloud solution is not appropriate for the organization today, it may become necessary at a later date. Since test automation should be designed to last for several years, these types of environment changes must be considered early on.

4.6.1.2. SUPPORT FOR COORDINATION

It may be necessary for a tool to be able to support transactions being sent to and received from multiple devices and simulators. The tool may need the ability to correlate this information, particularly for server tests, to understand what is happening on the system at the time the test automation is being executed. This coordination may include:

- Number of transactions
- Timing of transactions
- Types of transactions
- Summarized reporting

Copyright ASTQB,
All Rights Reserved

Tools that are not capable of managing this coordination will cause the test team to spend considerable manual effort to set up tests and analyze results.

4.6.2. SKILLS NEEDED

As with any test automation project, programming and scripting skills are needed to develop high quality, maintainable test scripts. It can be misleading to think that an IoT or mobile device application will have a short life in production before being upgraded and therefore the test automation code will only have a short life as well.

Some IoT and mobile devices may remain in service for many years. Good test automation can grow with the product and can provide good regression testing for the old features when new features are introduced. In order for this to happen, the test automation architecture must be robust – built to last and grow.

In the IoT and mobile device application world, the capability sets will continue to grow.

Automation will not be stable – it will always be expanding to accommodate new features of the device and the applications. Tools may lag behind device development. It is a reasonable expectation that programming or scripting will be required to bridge the gaps between tool capabilities and device capabilities. Before coding to fill a gap, the tester should check if a reliable tool is available. Tools are developed and deployed very quickly, therefore frequent investigation is warranted to avoid unnecessary effort.

Copyright ASTQB,
All Rights Reserved

5. The Internet of Things – 90 mins.

Keywords

connected smart car, dual path connectivity, edge computing, fog computing, wearable device, Industrial Internet of Things (IIoT)

Learning Objectives for The Internet of Things

5.1 Introduction

None

5.2 The Nature of The Internet of Things

LO-5.2.a (K2) Explain the unique characteristics of the IoT

LO-5.2.b (K1) Recall applications of the IoT

5.3 Risks and Benefits of the IoT

LO-5.3.a (K2) Describe the risks of the IoT

LO-5.3.b (K2) Describe the benefits of the IoT

Copyright ASTQB,
All Rights Reserved

5.4 IoT Implementations

LO-5.4.a (K1) Recall examples of smart appliances

LO-5.4.b (K1) Recall examples of wearables

LO-5.4.c (K2) Describe the difference between autonomous systems and connected systems

LO-5.4.d (K1) Recall IoT applications

LO-5.4.e (K2) Explain why information may be maintained locally

5.5 IoT Protocols and Environments

LO-5.5.a (K2) Understand the various protocols that are used in the IoT

LO-5.5.b (K2) Explain the complexity of IoT testing environments

LO-5.5.c (K2) Explain the uses of edge and fog computing

5.6 Unique Testing Approaches and Techniques Needed for the IoT

LO-5.6.a (K4) For a given IoT implementation, define a testing approach that meets IoT quality objectives

5.7 The Role of Test Tools in the IoT

LO-5.7.a (K2) Summarize the types of test tools that can be used in testing IoT devices and implementations

5.1. Introduction

Most of this syllabus has discussed IoT and mobile devices, but there are further considerations regarding the devices that are part of the Internet of Things (IoT). These devices and their applications have special considerations for testing, in addition to the testing areas already discussed. The IoT devices,

Copyright ASTQB,
All Rights Reserved

which are also IoT and mobile devices, have a wide range of applications and this influences the level and degree of testing required.

This section provides background information on the IoT and discusses the unique aspects of the IoT devices and applications, and how those can affect the scope and depth of testing. As always, the level of testing required should be correlated with the amount of risk that must be mitigated or, at least, understood.

5.2. The Nature of The Internet of Things

As the number of “smart” devices increased dramatically with the rise of high-speed Internet connectivity, so did the need to connect and control the devices. The term that was coined in 1999 to describe this massive collection of Internet-IoT and mobile devices was the “Internet of Things” or “IoT”. The idea behind the IoT is that each device has an unique identifier (UID) and can be associated with many types of things, such as other devices, appliances, animals or even people.

The Gartner Group estimates that by 2020 over 20 billion IoT devices will be in use worldwide. While this exponential increase in IoT devices presents a variety of opportunities and benefits, it also presents great challenges in terms of security, performance, interoperability and correctness.

It is important to understand that IoT devices can take many forms, such as sensors, medical devices, industrial controls, home automation and security devices (e.g., cameras and lighting), automotive sensors and applications, as well as integration with other IoT and mobile devices, such as mobile applications to control some IoT devices.

Copyright ASTQB,
All Rights Reserved

An IoT device is an IoT or mobile device but not necessarily a mobile device. For example, a home security camera with Internet connection is an IoT device but remains in one location. However, an IoT device might also be mobile in the case of an implanted medical device that is constantly sending information to a monitoring station.

5.2.1. UNIQUE CHARACTERISTICS OF THE IOT

The IoT has several unique and distinguishing characteristics:

- The ability to be in virtually any thing or person (such as a medical device in a human)
- No need for any human-to-human or human-to-computer interaction
- Lightweight data transmission protocols (such as MQTT)
- Part of a larger digital ecosystem that shares information
- Devices are under constant use, in most cases
- Devices require a high level of interoperability, often achieved with APIs and microservices
- Devices can be discovered worldwide, hence security is important

These characteristics present unique challenges for testing.

5.2.2. CHANGING THE FACE OF TECHNOLOGY AND BUSINESS

While the concept of IoT is not new, people are only beginning to see the applications and impact of the IoT as everyday items are starting to incorporate them. This includes home automation, medical devices, bio tracking, automobiles, entertainment devices, digital assistants and kitchen appliances.

This rapidly growing deployment of the IoT began when manufacturers started to incorporate IoT functionality into TVs, appliances, and other

Copyright ASTQB,
All Rights Reserved

devices. Now the interest in the IoT is shifting from a technology focus to a business and consumer focus. People are intrigued by new gadgets and innovative technology. This “cool factor” of the IoT has started a race to the marketplace for companies to be first with this technology in their market niche.

Developers and testers must deliver on management and marketing promises. This pressure to deliver has forced developers and testers to learn new technologies, new approaches and new ways to work together.

5.3. Risks and Benefits of the IoT

The IoT presents many benefits and opportunities, but risks must also be considered. Some of the key risks include the following:

- **Security –** With an IP address, IoT devices can (and have been) hijacked by hackers. Many cybersecurity professionals agree that the security risk of IoT is great due to vulnerabilities in the devices and lax user configuration, such as not changing default passwords to the main device. In addition, the huge number of IoT devices provides an opportunity for hackers to create an enormous botnet for a Distributed Denial of Service (DDoS) attack.

- **Connection –** Without Internet connectivity, there is no IoT. In some areas of the world, Internet connection is poor or non-existent, making the functioning of these devices limited or impossible.

- **Interoperability –** Integration is how devices work together. Interoperability is the ability of the software and devices to use the information exchanged correctly. With more devices sharing more information, the risk of misinformation being sent or received increases.

- **Correctness –** IoT devices must function correctly or else great loss can occur, especially in safety-critical applications, such as medical devices and automobiles.

- **Reliability –** A unique concern with IoT devices is that in many cases, they are always "on" and sending and receiving data. This means that power sources must be reliable and any connectivity or power-related issues must be reported accurately and quickly.

- **Performance –** IoT devices must function at high levels of efficiency because response times are often required to be within very high levels. An example of this is when IoT devices control home security functions.

- **Usability –** While user interaction is not required for the actual transmission of data, user interaction is often required for device setup. Poor usability in configuration setup can lead to failures due to device configuration errors.

- **Trust –** In consumer products and other application areas, such as healthcare and travel, the IoT has the capability of capturing and transmitting sensitive personal information. People have legitimate reason to be concerned with what will be done with the data obtained through IoT devices and their use.

For example, in a well-documented court case, audio files captured by an Amazon Echo device were used as evidence in a murder case [CNN].

Copyright ASTQB,
All Rights Reserved

Despite these risks, there are a number of clear benefits with the IoT. These include the following:

- The benefits resulting from automation of routine tasks, such as locking the doors of the house at night
- Increased insight gained from the monitoring of business processes and functions
- Improved customer experience by targeting services and products, such as being able to remotely record favorite TV shows
- Savings of time and money by reducing human interactions with devices
- Enhancing and increasing employee productivity
- Gaining customers and generating more revenue

In order to achieve the benefits, the risks must be understood and mitigated.

5.4. IoT Implementations

While it is common to speak of the “Internet of Things” as a single type of application, in reality there are many ways in which the IoT can be implemented and the possibilities are continually increasing. When considering how to test a “Thing”, one must first understand the application and its context of usage.

5.4.1. APPLIANCES

While “smart” appliances are still a small minority of appliances in use in households today, there is a clear trend toward appliances having IoT functionality. There has been much made of the example where a smart

Copyright ASTQB,
All Rights Reserved

refrigerator can detect that it is low on milk and can automatically order more from the grocery store. In reality the more likely use case is for the smart refrigerator to detect a temperature that is too cold or too warm and alert the user, or self-adjust and advise the user. Another use case is the ability of a consumer to remotely control an appliance, as well as check the status of an appliance that must be continually working.

5.4.2. WEARABLES

The term “wearables” applies to any IoT and mobile device that is worn by a human. Wearable devices can be incorporated into clothing or worn on the body as implants or accessories.

Examples include watches, glasses, fitness trackers, virtual reality (VR) headsets, audio headsets, heart monitors and cameras (such as police bodycams). In many cases, wearables are paired with a mobile device, such as a smartphone, to deliver information to the user. However, wearable devices also often display information directly to the user and can use their own capability to upload information directly to the Internet.

5.4.3. AUTOMOTIVE

Today, the average newer model automobile contains in excess of one million lines of code. A high percentage of that code is for the monitoring and control of the automobile in a non-connected mode. Some of the code is used for autonomous functions, such as traction control, self-parking and self-braking. Self-driving cars are at the extreme end of the autonomous spectrum.

However, some of the functionality contained in modern cars is delivered through IoT and mobile devices and are considered IoT. Some call this the "connected smart car". Examples of the IoT in connected smart cars include GPS navigation, vehicle location tracking, speed control, driving analytics (used by insurance companies, employers and parents), and emergency call functions. While the IoT offers new and helpful features for individual car owners, an even greater opportunity is seen by the owners of fleet vehicles to monitor and control hundreds of vehicles driven by many drivers.

A unique consideration for automotive IoT is the lack of complete network coverage, such as when driving through sparsely populated areas. Where connectivity is not supported by cellular networks, full functionality requires that a satellite connection be available.

5.4.4. INDUSTRIAL

One of the greatest levels of IoT adoption is in the industrial and manufacturing sectors. In fact, IoT has become such a major trend, the term "IIoT" (Industrial Internet of Things) is being used to describe the specific context and implementation of the IoT in the industrial sector. The Industrial Internet of Things is commonly defined as "machines, computers and people enabling intelligent industrial operations using advanced data analytics for transformational business outcomes". This is also known as "Industry 4.0".

This distinction was needed to address the difference seen in industrial applications as compared to the enterprise IoT and the consumer IoT. The IIoT is applied across a variety of industries, such as manufacturing, transportation, oil and gas, logistics, aviation, energy/utilities, mining and metals, and other industrial sectors. The main focus in many IIoT implementations is on operational efficiency, along with cost optimization. However, these goals should result in other more significant goals related to higher profitability and innovation.

The IIoT is not the same as traditional industrial automation, which has been used for many years and does not necessarily require Internet connectivity. However, the IIoT can be used as a driver and monitor for automation.

Taken to the largest context across industries, a partial list of IIoT applications include:

- Connected logistics
- Smart factory applications and smart warehousing
- Predictive and remote maintenance
- Freight, goods and transportation monitoring
- Smart environment solutions
- Smart city applications
- Smart metering and smart grid
- Smart farming and livestock monitoring
- Industrial security systems
- Industrial heating, ventilation, and air conditioning
- Energy consumption optimization
- Manufacturing equipment monitoring
- Asset tracking and smart logistics
- Ozone, gas and temperature monitoring in industrial environments
- Safety and health (conditions) monitoring of workers
- Asset performance management
- Remote service, field service, remote maintenance and control

This is only a partial list and applications are growing, both in breadth and depth.

5.4.5. HOME AND INDUSTRIAL SECURITY

Home and industrial security are major users of IoT technology. In the past, security cameras were completely based on physical recording equipment with low resolution video. Now, security companies and cable providers are promoting home security systems as part of their offerings. In some cases, the set-up may be a simple doorbell camera that alerts a user via smartphone when someone approaches the home.

Facial recognition is also a feature of IoT security found in high-end systems, such as those used for highly secure facilities. However, as video surveillance becomes more prevalent, facial recognition is also becoming a feature in home and small business systems, along with the methods for reviewing large amounts of video footage. For example, Google now has technology and an API for searching video for people (“Jack Jones”), things (“blue hat”), activities (“walking”), text (“ABC”) and other parameters. Many of these features are now available on Google Photos and the Google Lens app.

As the information being stored becomes more secure (such as a finger print that will allow access to a secure facility), a trend is developing to keep data on local devices to mitigate cybersecurity risks. This also applies to local integration, where devices network locally instead of in the cloud.

Copyright ASTQB,
All Rights Reserved

5.4.6. DIGITAL ASSISTANTS

One of the most acquired IoT devices for home use has been the digital assistant (e.g., Amazon Echo, Google Home, Apple Home Pod). These devices can perform a multitude of tasks ranging from web queries and ordering products to controlling other home devices such as lighting, climate control, home security and home entertainment.

The great challenge in using and supporting digital assistants is frequent changes to the device software or APIs that can cause other IoT and mobile devices to fail at unexpected times.

5.5. IoT Protocols and Environments

5.5.1. PROTOCOLS

The key thing to understand in testing IoT devices in any context is the protocols used in networking. These protocols, such as MQTT (Message Queuing Telemetry Transport, also called Mosquito), MQTT-SN (MQTT for sensor networks), CoAP (Constrained Application Protocol), REST (Representational State Transfer), and the common web protocols of HTTP/HTTPS, can be used across any network type, such as Wi-Fi, 4G, or 5G.

There are dozens of protocols used in IoT for infrastructure, communication, identification, data transfer, discovery, and device management. To describe all IoT protocols is beyond the scope of this syllabus. To see a complete list of IoT protocols and learn more about them, refer to [Postscapes].

Copyright ASTQB,
All Rights Reserved

While some IoT devices have only one means of connection (such as Wi-Fi or cellular), it is becoming commonplace to have “dual path” connectivity. Dual communication is the use of redundant modes of communication in case one of the connections fail. For example, an IoT device that connects from a premises to a central monitoring station or cloud-based service could have dual paths consisting of IP and cellular communications. However, another way to implement dual path connectivity could be the use of two different mobile carriers. Basically, the decision depends on which types of connections are available and what redundancy offers the best solution.

5.5.2. ENVIRONMENTS

IoT devices are designed to work with as many other devices as possible. However, it is common for home IoT consumers to acquire IoT devices within a digital “ecosystem”. For example, Apple users may prefer to use iTunes with an Apple Home Pod. Amazon users may prefer to use Amazon Echo with Amazon Music.

IoT devices can be oriented to a wide variety of environments, such as home automation, appliances, manufacturing, healthcare, automotive, wearables and utilities. To effectively test the IoT device, a real-world replication of the operational environment must be created or obtained at some point. Some IoT testing can be done safely in the actual production environment, or “in the wild”. For example, in testing wearables, alpha and beta testers might wear a fitness tracker during their daily activities for a period of time. However, care must be exercised when testing in a production environment if safety is at risk. For example, testing self-driving cars in actual traffic can put both testers and other drivers at risk of bodily injury or death.

Copyright ASTQB,
All Rights Reserved

Edge computing is a “mesh network of micro data centers that process or store critical data locally and push all received data to a central data center or cloud storage repository, in a footprint of less than 100 square feet,” according to research firm IDC.

The goal is to process IoT data closer to where it is created instead of sending it across long routes to data centers or clouds. Edge computing is often done by the IoT devices transferring the data to a local device, which can perform computations, store data and have network connectivity in a small format. Once data is collected and processed by the local device (the edge), all the data, or a portion of it, can be sent to a storage repository in a corporate data center, co-location facility or IaaS cloud for central processing.

“Fog computing”, or “fogging”, is the term used to describe the architecture that employs the edge devices to provide local processing. This is in contrast to cloud computing that uses remote servers on the Internet (i.e., in the cloud) to store, manage, and process data away from the local device that gathered the data. The OpenFog Consortium is an association of major tech companies aimed at standardizing and promoting fog computing.

In testing the IoT in an edge computing environment, three aspects must be tested – the local processing environment and functionality (computations, data transformation and storage), networking to the cloud as data is sent for processing, and the cloud processing environment and functionality.

5.6. Unique Testing Approaches and Techniques Needed for the IoT

Testing the IoT requires a unique approach due to the wide variety of applications, environments and technologies involved. These approaches include:

- **"Headless testing" –** since the IoT device may not have an accessible UI for functional testing
- **API testing –** since connectivity between the IoT device and the receiving devices (edge devices or cloud) utilize APIs for sending and receiving data
- **Performance testing –** because response time is often critical for IoT devices
- **Security testing –** due to the high risk of compromised communication
- **Usability testing –** especially in consumer devices
- **Structural testing –** at the unit test level to find and prevent code-based defects that could cause reliability, functional, security and performance failures
- **Interoperability testing –** because the main purpose of the IoT is high integration and interoperability
- **Testing of networking protocols and the data sent across the network**

Copyright ASTQB,
All Rights Reserved

Similar to the testing of mobile applications on real devices, one approach for testing the IoT involves crowdsourcing.

5.7. The Role of Test Tools in the IoT

There is a role for test tools in the IoT, however, many of the tools tend to be open source and oriented primarily for testing network transmissions of data between IoT devices and edge devices or cloud applications.

There is also the need to consider the role of tools in testing IoT applications. For the purpose of IoT application testing, the same tools used for testing mobile applications can be used here, since the focus is on testing the application and not the device. IoT application testing should occur before IoT device testing to avoid finding defects later in the development lifecycle.

Examples of test tool usage in IoT are:

- **Wireshark –** to examine network packets as they are transmitted across the network
- **Tcpdump –** to capture and examine packet contents
- **Gatling or Locust –** for performance testing
- **SoapUI –** to test APIs

Copyright ASTQB,
All Rights Reserved

Two websites which are also helpful in IoT testing are:

- **Shodan.io (https://www.shodan.io/) –** A site for finding which devices are connected to the Internet. It can also be used to keep track of all the computers which are directly accessible from the Internet. Security testers and hackers also use Shodan to find IoT devices which are using default passwords.

- **Thingful.net (https://www.thingful.net) -** A search engine for the IoT. It allows secure interoperability between millions of objects via the Internet. Thingful is also used to control how data is used and empowers data owners to take more decisive and valuable decisions.

There are commercial tools available for API testing, performance testing, service virtualization and other types of testing. As with any test tool, new vendors and offerings are always entering the marketplace. Any tool search should be based on a set of pre-defined evaluation criteria, which are based on your needs. A further discussion on tools can be found in Section 4.

6. Future-Proofing – 135 mins.

Keywords

none

Learning Objectives for Future-Proofing

6.1 Expect Rapid Growth

CD-6.1.a (K1) Recall ways in which the IoT and mobile device application and device market will expand

CD-6.1.b (K1) Recall areas in which user expectations will increase

6.2 Build for Change

CD-6.2.a (K2) Summarize the considerations for building a flexible testing

Framework

CD-6.2.b (K4) Analyze a given IoT and mobile testing project and determine the appropriate activities to reduce maintenance costs, while enabling wide product adoption

6.3 Plan for the Future

CD-6.3.a (K2) Explain how lifecycle models are likely to change and how this will affect testing

6.4 Anticipating the Future

CD-6.4.a (K1) Recall the ways in which testers will need to adapt

Copyright ASTQB,
All Rights Reserved

6.1. Expect Rapid Growth

If the past is any indication, the connected application and device market will continue to expand. This means there will be more types of devices and variations of existing devices. The devices will have more capabilities. There will be more applications and those applications will continue to become more feature rich to take advantage of device capabilities and increased memory.

But those are not the only areas for growth. The number of users, the variety of users and the expected usage of these devices will also grow. As the devices become more and more a part of personal and business life, user expectations will continue to increase in the following areas:

- High reliability
- Excellent usability
- High performance
- Consistent experience
- Portability
- Fast turnaround for fixes and new features

As a result of these ever-increasing expectations, software testers will need to become more adept at defining and executing the necessary tests, while also facilitating a rapid time to market. This will require leveraging the right tools, picking the appropriate environments and using new approaches to deal with the large number of devices and user types.

Copyright ASTQB,
All Rights Reserved

6.2. Build for Change

As profit margins decrease due to competition in the market, pressure will be put on the development and testing teams to produce high quality, maintainable products quickly. Testing will need to engage early with development in order to plan out the testing, procure the tools and environments, and upskill as needed.

6.2.1. ARCHITECT THE TESTING

It will not be enough for the developers to plan for change. The test approach must also be architected for change. Some of the considerations for this flexible framework include:

- Implementing or utilizing test environments that can be quickly assembled and disassembled
- Utilizing the most appropriate tools for the tasks
- Implementing maintainable test automation (e.g., keyword-driven)
- Designing the load testing approach at the beginning of a project
- Conducting load testing throughout the software development lifecycle
- Maintaining a reasonable set of representative devices
- Accommodating a risk-based testing approach
- Providing a framework to support crowd-sourcing
- Providing a strong ROI that supports the shorter lifecycle

Tools must be selected for flexibility and ability to adapt to the changing market. Tool vendors must be highly responsive to the market changes. Relying on past reputation will not be sufficient in a market that is moving quickly and has many tool options.

Copyright ASTQB,
All Rights Reserved

While re-usability is always a goal, in the short lifecycles of the IoT and mobile device applications, achieving time to market with a quality product may supersede the value of having a fully reusable testing solution. Re-usability will likely save money in the long run, but too much time spent creating a fully re-usable testing solution may result in an unacceptable delay in product release. Investments must be justified by the expected lifetime of the product. Today's best smartphone is likely to be tomorrow's paperweight, so the investment in the testing must be justified.

To complicate matters, users with IoT devices (such as connected appliances and automobiles) may retain those appliances for many years, which will present challenges for users, vendors and testers in sustaining long-term approaches to testing and maintenance.

6.2.2. ENABLE EFFICIENT MAINTENANCE

While maintainability may not be the ultimate goal, efficient maintainability is a requirement. The test environments, tools, and testware must be able to be efficiently maintained or replaced. It may be that an environment is needed for an initial release of an application but will not be needed for the next release. This means the investment in the environment must be justified by the risk mitigation that can be achieved from testing in that environment. Unlike traditional software, where longer term planning is justified, IoT and mobile device application software may have a much shorter lifecycle. The maintenance of the test environment and the need for reuse may also be similarly limited. The exception to this is in IoT devices that may remain in service for long periods of time, with their applications being more stable with less frequent updates.

Copyright ASTQB,
All Rights Reserved

6.2.3. SELECT TOOLS FOR FLEXIBILITY

If tool vendors cannot be responsive to the market, purchasing their tools and creating significant test assets with those tools would be foolish. The tools must be as flexible as the products will be and any investment in test automation or even test management must conform to the expected lifecycle of the product. Inexpensive tools that can be used to create less resilient test automation may be justified over heavier weight tools that will create a product that will provide long term maintainability, particularly if the product is not planned to be used long term.

When designing testware, it may make sense to design it to work with a simulator, rather than a particular device. This may offer more flexibility in the long run as devices evolve and add more features. Developing to a simulator will also allow test execution prior to the availability of the real device. Care must also be taken when using open source tools for IoT and mobile device automation as changes may occur in the tools, which may require extensive changes throughout a test automation library. It is also important to remember that some IoT and mobile device applications are safetycritical.

In this case, the tools used to test the applications, such as simulators and test automation tools, may require a level of certification from a regulatory group before their use is approved.

6.2.4. SELECT PARTNERS CAREFULLY

When third party relationships are formed for testing, these relationships should be built on the assumption that there will be many releases. Long term relationships will allow more flexibility in meeting market needs without incurring downtime due to contract negotiations, requirements debates and so forth. If a third party relationship will be formed, it is important to ensure that the partner will be able to keep up with the changes in the industry. Expect a demonstration of flexibility and ability to adapt to changing markets. A partner that is locked into a particular tool set or a particular methodology may not provide the agility needed to cope with the rapidly changing market.

These relationships may include outsource testing, device labs, cloud environment vendors and so forth. Because building a full IoT and mobile device application test capability may be cost prohibitive, partnerships may be the most viable way to meet variable needs.

6.3. Plan for the Future

6.3.1. LIFECYCLE MODELS

It is likely that new lifecycle models may be introduced as a result of the application and device requirements for testing. Agile and iterative lifecycles already dominate the industry, but new leaner methodologies may become popular. The tester must remember that any lifecycle model will require some adaptation and the tester must understand the moment and level of involvement that will be expected in the various models.

Timescales for the development and testing of applications have never been so short and the market demand has never been so high. Of course, the opportunities are plentiful. However, an organization's reputation can be badly damaged by poor quality products and ill-considered feature sets. The tester must expect to work with developers to ensure the best possible product is being released within the given timeframe.

6.3.2. ALTERNATIVE TESTING

While new lifecycle models, such as DevOps, have emerged and others may be on the horizon, there will still be a need for efficient testing. This means taking a lightweight approach to deliver a proper ROI. Automation and performance testing will be required and the tools must be appropriate to the needs, schedules and budgets. Security will always be a consideration and security testing tools will likely adapt to fill the mobile market as well.

Testing in the cloud is likely to become a commonplace practice, where device simulators and user simulators are used to create and test a realistic variety of transactions and system load. Crowd-sourcing, testing-in-the-wild, and other forms of outsourcing are likely to continue to grow and present test management challenges.

As the tester community expands to non-testers, automated error reporting and screen capture will become more commonplace to help developers track and diagnose issues that occur in production. This information should also be fed back through the testing process to ensure improvements are made to close any testing gaps.

6.4. Anticipating the Future

Planning testing practices, processes, and tools for use two to three years in advance is difficult in the IoT and mobile device application space. New applications and devices emerge daily and competition will continue to drive the market and lifecycle.

Testers need to be ready and willing to adopt new technologies, investigate new tools and learn more efficient and leaner testing methodologies. Some products and practices will be unsuccessful but good research should allow testers to select the right approaches and the best tools.

7. References

7.1. ASTQB and AT*SQA Documents

- AT*SQA Essentials of Software Testing Syllabus
- AT*SQA DevOps Syllabus
- ASTQB Performance Testing Syllabus
- ASTQB Security Testing Syllabus
- ASTQB Test Automation Syllabus
- Standard glossary of terms used in Software Testing

7.2. Trademarks

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

- ASTQB® is a registered trademark of the American Software Testing Qualifications Board (ASTQB)
- Teststorming™ is a trademark of Rice Consulting Services, Inc. and is used by permission, with permission granted to freely use Teststorming in the creation of courses and other derivative works with attribution to Rice Consulting Services, Inc.
- AT*SQA® is a register trademark of Association for Testing and Software Quality Assurance Global Certification Body, Inc.

7.3. Books

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

[Whittaker]: James Whittaker, Jason Arbon, Jeff Carollo, “How Google Tests Software”, Addison-Wesley Professional; 1 edition (April 2, 2012), ISBN-10:0321803027

7.4 Other References

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

[CNN] “Suspect OKs Amazon to hand over Echo recordings in murder case” - https://www.cnn.com/2017/03/07/tech/amazon-echo-alexa-bentonville-arkansasmurder-case/index.html

[InfoQ] http://www.infoq.com/news/2014/10/world-quality-report

[Nielsen] Jakob Nielsen, “Why you only need to test with 5 testers”, www.useit.com

[OpenDeviceLab] www.opendevicelab.com

[OWASP]
https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks

[Paskal] www.gregpaskal.com
[Postscapes] https://www.postscapes.com/internet-of-things-protocols/

[Rice] “Teststorming”. http://www.riceconsulting.com/home/index.php/General-Testing-Articles/teststorming-a-collaborative-approach-to-software-test-design.html

[Webtrends] http://www.webtrends.com/blog/2010/04/top-metrics-for-mobile-appsmeasure-what-matters/

Copyright ASTQB,
All Rights Reserved