AT*SQA

Testing for IOT & Mobile Micro-Credential

Syllabus

AT*SQA

MICRO-CREDENTIAL

Testing for IOT and Mobile

Table of Contents

General Information

STUDY TIME – 180 MINS.

KEYWORDS

connected devices, emulators, lightweight testing, simulators

LEARNING OBJECTIVES FOR IOT (TESTING CONNECTED DEVICES)

Introduction

Connected Devices

(K2) Understand the challenges with connected device testing

Environments and Tools

(K2) Understand the differences between simulators and emulators, and which is appropriate for a given situation

(K1) Recall how test automation tools can assist in connected device testing

Quality Characteristics

(K1) Recall why understanding user expectations is particularly challenging for connected devices

(K1) Recall why requirements for quality characteristics must be identified early in the lifecycle

(K2) Explain why usability is an important quality characteristic to consider in connected device testing

(K2) Explain why performance is an important quality characteristic to consider in connected device testing

(K2) Explain why security is an important quality characteristic to consider in connected device testing

(K2) Explain why interoperability is an important quality characteristic to consider in connected device testing

(K2) Explain why accuracy is an important quality characteristic to consider in connected device testing

(K2) Explain why reliability is an important quality characteristic to consider in connected device testing

Lightweight Testing

(K2) Describe how lightweight testing can be advantageous in connected device testing

Introduction

The world of connected devices is expanding. What started with mobile phones soon became smart phones and tablets, then blossomed into the Internet of Things (IoT). As software has adapted to be quicker and smaller, testing must also adapt to be quick and lightweight. That said, good testing practices still apply and quality characteristics are still important to the users. However, it is important to remember that users have expectations that their mobile applications and IoT devices will “just work”.

Connected Devices

Devices in the connected world vary from smart phones 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 actually more is needed. There is an expectation from users not to be forced to upgrade in order to take advantage of new features and applications.

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. Just because the software runs on a refrigerator allowing the user to increase or decrease the temperature remotely does not mean that all models of refrigerators need to be tested. 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, but if all refrigerators use the same communication interface with the Internet device, then testing one may be sufficient (i.e., applying equivalence partitioning).

Environments and Tools

The proliferation of devices supported by Internet appliances has resulted in a plethora of tools and simulators/emulators that can be used for testing. This reduces the need for having a large set of devices available for testing and can greatly speed up manual testing and the development of test automation. Device labs are available for popular devices, such as smart phones, and new simulators/emulators are constantly being created. It is always good practice to verify the results from a simulator/emulator against a real device, but the majority of the testing does not require the more expensive real devices.

Simulators generally provide a standard response to various inputs and “simulate” the interaction with a real device at the software level. Emulators go a step further and “emulate” the responses of the hardware device as well as the software running on that device. For example, testing an application’s interactions with a smart phone’s gyroscope requires an emulator, whereas testing interaction with the device’s email application can be done with a simulator. Similarly, simulators in the form of service virtualization can be used to simulate a web service’s interaction with an application.

Test automation tools are quickly adapting to the IoT and mobile device market. Many of these tools will interact with simulators and, sometimes, even include their own simulators. Simulators for compatibility testing, such as cross-device or even cross- browser, are commonly supported by test automation tools. The tendency is for these automation tools to be more lightweight in features than the traditional client/server or web services tools. Cost is always a consideration and the open source market quickly adapts to the needs for new and purpose-built tools.

Even the best simulators/emulators cannot simulate/emulate everything. There are user actions, such as a complex set of gestures, that must be tested on a real device. It is important for the tester to determine which tests are best conducted with which environment. Generally a mix of simulators/emulators and real devices will yield the most accurate result for the least cost.

Quality Characteristics

While quality characteristic testing is important for all types of software, there are some unique needs for quality characteristic assessment when dealing with connected devices. One of the most difficult aspects of connected device 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 "acceptable". Defining and documenting this range early in a product's lifecycle is particularly important for connected 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.

While all quality characteristics are important, the following characteristics are particularly important with connected devices:

- Usability
- Performance
- Security
- Interoperability
- Accuracy
- Reliability

Usability

As connected devices become more integral to modern life, users expect "good" usability and learnability. Particularly for downloaded applications for smartphones, users expect software to be attractive, inviting, easy to use, easy to understand, and enjoyable. Users are becoming less patient with software that does not provide a good user experience. This can result in a product that is functionally sound being rejected by the users because it does not provide the expected feedback (e.g., prompts and sounds). Of course, being functionally sound is still the basic building block of software quality, but the ease with which a new user can and wants to work with an application often determines the market share. See the usability micro-credential syllabus for more information.

Performance

Performance criteria tend to be loosely defined and are often identified when a product does not meet expectations. While this is true for any software product, it is apparent for applications for connected devices which may have limited capability (memory, bandwidth). Ideally, the performance criteria should be defined and captured early in the requirements phases of a project. Defining these criteria, setting up the proper personas and benchmarks, and testing to ensure the criteria are met on the target device, are 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 on a smart phone to not respond in time for a driver to make a turn.

Another variable with connected 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. See Section 5 for more about performance.

Security

Connected devices are sometimes used for safety-critical applications as well as for financial transactions. Cybersecurity testing is difficult with the shortened development cycles of connected devices, but it is a critical component of a quality product. Architecting, developing, and testing for security must occur throughout the development lifecycle, as there will rarely be enough time at the end of development for thorough cybersecurity testing (and any necessary corrections/changes).

It is important to understand the expected and potential usage of an application running on a connected device. Is the GPS-based guidance application used to get to the mall or to direct ambulances to emergency scenes? Is a web-monitored intruder alarm used to guard the refrigerator from marauding teenagers or to protect a pharmacy? Seemingly simple devices can easily be used in safety-critical situations, necessitating the highest levels of security testing.

Like quality, security must be built into the product right from the start.

Connected devices have unique security risks. For example, an attacker can gain access through man-in-the-middle attacks by exploiting Bluetooth vulnerabilities of the device. Using connected devices with public Wi-Fi hotspots can lead to security vulnerabilities that would not be experienced on a controlled and protected network. Because mobile devices are easily carried with a person, they are also easily lost or stolen, which can allow personal information to fall into the hands of a criminal. Connected medical devices present a unique security risk as they have been compromised in past Wi-Fi cyberattacks to gain access to the larger network in a hospital. The list goes on and on. In general, security requirements for connected devices are always expanding, particularly as new vulnerabilities are discovered.

See the cybersecurity micro-credential syllabus for more information.

Interoperability

Connected devices offer some unique challenges for interoperability. Software is often developed to work on multiple operating systems and a range of devices. For example, a banking application may be intended for use on a wide range of smart phones and tablets. The portability of an application to different devices, the ability of an application to interoperate with other software and the compatibility of the software across different browsers and operating systems are often lumped into the general category of interoperability.

From a testing perspective, this gives a nearly infinite number of combinations to test and those combinations continue to evolve rapidly. Using combinatorial testing techniques can bring this potential set of test targets down to a manageable number. This type of testing tends to be pushed to the end of the testing cycles, when there is enough functionality to support the testing.

The architecture of the product often determines its interoperability capabilities. If the architecture is limiting, the capability of the end product will also be limited.

Accuracy

Accuracy testing targets the ability of the software to provide accurate results for a given set of inputs. Users expect software to be accurate, but there are specific considerations for connected device software primarily because the usage can be so varied. A simple humidity detector developed on a Raspberry Pi may be intended for use by plant enthusiasts to ensure proper watering levels. However, this same detector could be used to ensure an asthmatic child has the proper humidity in their room air. Because the use of a product may not be controllable, accuracy must be ensured for any possible safety-critical usage. This tends to push most of the connected device testing into the safety-critical arena, unless a product can be specifically excluded from a safety-critical use.

Testing in a safety-critical environment requires higher levels of documentation and due diligence to protect the developing organization from potential legal action if something should go awry. This is an area that is open to debate, but from a testing standpoint, more thorough and better documented testing will help mitigate deployment risk and may help meet regulatory and industry standards. This is particularly challenging in the short cycles of connected device products and is why lightweight testing approaches are critical.

Reliability

As with the higher expectations for usability and performance, users expect complete reliability from their connected devices and software. No one expects to reboot their smart phone and much less their refrigerator. Realistic or not, this is the expectation that must be met in order to capture and retain users. Testing for reliability is difficult and requires test environments that are representative and stable. It also requires time, as reliability tests usually require applications to be observed under continual use for a given length of time to determine the mean time between failure (MTBF). The time required for this testing potentially conflicts with the goal of being quick to market.

Reliability cannot be tested into an application – it must be built in. This means that reliability targets must be set early and reviewed frequently. Reliability testing in the connected device area tends to be limited to discovering whether there are significant problems. More subtle problems or problems that only appear over long usage tend to be discovered in production. Perhaps more than any other area, reliability assurance is a development activity more than a testing activity.

Lightweight Testing

A common theme has emerged in this section – software has to be built right because there is no time to fix it later. This means the requirements for quality characteristics must be defined and understood by the development and testing teams. In lightweight testing models, such as Agile, the whole team approach helps everyone to review and understand these requirements from the beginning and to revisit and validate fulfillment of the requirements throughout development. Any lightweight methodology is dependent on the engagement of the BAs, product owners, developers and testers throughout the lifecycle to ensure that the product is reviewed and tested as it is being built. Testing cannot be pushed to the end of the process if the schedule time is to be reduced and quality criteria met.

Testing documentation must be as lightweight as possible in this environment. This means that detailed test cases are probably not needed, but repeatable tests are. Testing from decision tables and checklists rather than from step-by-step test cases is a good way to test the implementation of business processes and workflows. The testing techniques can be well-applied to reduce testing documentation while increasing coverage and repeatability. Exploratory testing can help fill the gaps between the testing techniques but should not be the only form of testing, as repeatability tends to be compromised or lost. To adapt to this changing environment, testers need to document the minimum needed for repeatability and to let risk and coverage goals guide the testing.

References

ISO/IEC/IEEE Standards

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

Trademarks

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

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

Books

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

[Firtman]: Maximiliano Firtman, "Programming the Mobile Web", O'Reilly Media;

Second Edition (April 8, 2013), ISBN-10: 1449334970

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

Other References

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

https://techcrunch.com/2013/03/25/ip-oh-my-gosh-all-that-money-just-disappeared/ https://www.reuters.com/article/us-facebook-settlement/facebook-settles-lawsuit-over-

2012-ipo-for-35-million-idUSKCN1GA2JR

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

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

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

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

AT*SQA

MICRO-CREDENTIAL

Testing for
IOT and Mobile

www.atsqa.org