Modern Software Engineering Methodologies for Mobile and Cloud Environments

António Miguel Rosado da Cruz
Instituto Politécnico de Viana do Castelo, Portugal

Sara Paiva
Instituto Politécnico de Viana do Castelo, Portugal

A volume in the Advances in Systems Analysis, Software Engineering, and High Performance Computing (ASASEHPC) Book Series

Information Science REFERENCE
An Imprint of IGI Global

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA, USA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Names: Cruz, Antonio Miguel Rosado da, 1970- editor. | Paiva, Sara, 1979- editor.
Title: Modern software engineering methodologies for mobile and cloud environments / Antonio Miguel Rosado da Cruz and Sara Paiva, editors.
Description: Hershey, PA : Information Science Reference, 2016. | Includes bibliographical references and index.
Identifiers: LCCN 2015046896| ISBN 9781466699168 (hardcover) | ISBN 9781466699175 (ebook)
Subjects: LCSH: Cloud computing. | Mobile computing. | Software engineering.
Classification: LCC QA76.585 .M645 2106 | DDC 004.67/82--dc23 LC record available at http://lccn.loc.gov/2015046896

This book is published in the IGI Global book series Advances in Systems Analysis, Software Engineering, and High Performance Computing (ASASEHPC) (ISSN: 2327-3453; eISSN: 2327-3461)

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 8
Quality Attributes for Mobile Applications

João M. Fernandes
Universidade do Minho, Portugal

André L. Ferreira
Universidade do Minho, Portugal

ABSTRACT

A mobile application is a type of software application developed to run on a mobile device. The chapter discusses the main characteristics of mobile devices, since they have a great impact on mobile applications. It also presents the classification of mobile applications according to two main types: native and web-based applications. Finally, this chapter identifies the most relevant types of quality attributes for mobile applications. It shows that the relevant quality attributes for mobile applications are usually framed in the Usability, Performance, and Maintainability and Support categories.

INTRODUCTION

A mobile application (or just mobile app) is a type of software application developed to run on a mobile device. Mobile applications are a recent type of software application that emerged due to the appearance of many handheld devices, like smartphones, which are considered to be the personal computer of the future (Duffy, 2012), or tablet computers. Mobile devices are expected to be easily carried, held, and used in the hands; hence, mobile devices are characterized by being relatively small devices when compared to desktop computers. For this reason, mobile devices are naturally restricted in several dimensions that strongly affect the operating characteristics of an application that executes on this type of devices. In many cases, the challenges for the software or systems engineer when developing a mobile application are similar to the ones to develop other types of embedded applications (Wasserman, 2010). Common issues include integration with hardware, security, performance, reliability, and storage limitations.

DOI: 10.4018/978-1-4666-9916-8.ch008

BACKGROUND

Until recently, mobile devices were characterized for having small display sizes, low processing power, poor connectivity, and limited interaction methods. The range of mobile applications available for these devices was relatively small. Mobile devices have evolved considerably in recent years as result of improvements in mobile technology, mobile networking and mobile computing. Examples are increased processing power of mobile devices, novel forms of user interaction, for example the introduction of the touch feature to operate mobile devices, and new protocols of connectivity. These improvements in the hardware motivated the increase in the range and availability of mobile applications.

Mobile applications are software systems that have specific characteristics when compared with normal desktop applications. Mobile applications differ from desktop applications by having to deal with limitations on specific hardware resources, like display size, processing power and memory to name a few. Probably the most significant difference is that mobile applications run on mobile devices with energy consumption limitations. Increase battery lifetime is a significant concern for both mobile applications developers and hardware manufacturers, namely in what concerns multimedia applications running on mobile devices.

Mobile applications can now take advantage of new hardware capabilities, meaning that they can provide a wider set of functions or use, which inevitably results in more complex software solutions. Development of mobile applications is now closer than before to the development of desktop applications in terms of complexity (examples are *Android OS*, *Apple iOS* and *Windows Phone* mobile operating systems); however the hardware is still a relevant differentiator. These differences motivate the need to understand clearly how the hardware constraints impact the software design decisions. For that purpose, one must understand which quality attributes are more relevant for mobile applications, providing an inevitable link to constraints imposed by the hardware itself. Understanding the possible impact of the hardware of mobile devices in software design decision motivates the discussion of this chapter.

MOBILE DEVICES CHARACTERISTICS

Mobile devices have a set of characteristics that have a great impact on how mobile applications are designed or created (Juárez-Ramírez et al., 2012). These characteristics establish the constraints that influence most technical and non-technical decisions on software applications development for mobile devices. A list of these characteristics is presented in this section.

Small Display Size: Mobile devices have relatively small displays when compared to desktop computers or laptops. Small mobile devices, in particular small smartphones, have display sizes of 4 to 4.5 inches but these can go down to 2.45 inches. Typical smartphones have display sizes between 4.5 to 6 inches and tablets range from 5 up to 13 inches. Mobile applications must be designed to successfully provide the desired functionality in a limited screen size. These conditions contrast greatly with normal applications that are designed to use larger screens. Small screen sizes pose great constraints to usability of mobile apps.

Small Memory Size: Normally, mobile devices have less memory available than desktop computers. Mobile devices, as any computing device, have both ROM (Read-Only Memory) and RAM (Random-Access Memory). ROM, which technically is EEPROM, i.e., Electrically-Erasable Programmable ROM, and so is both readable and writeable by the end user, does not require power to maintain its (non-volatile)

data and is basically used by a mobile device to store the operating system, mobile applications, media (photos and video) and files. ROM memory sizes go up to 128 GB (Gigabytes), but typical sizes are around 8 to 16 GB. However, some mobiles devices allow the ROM size to be increased, by using non-volatile memory cards, e.g., SD (Secure Digital) memory cards. Due to memory size restrictions, most mobile applications are relatively small, when compared with desktop applications (e.g., Microsoft Word).

RAM memory is used to store active processes when the mobile device is functioning. RAM memory usage can greatly impact performance of mobile applications running on a mobile device. Mobile devices hold up around 1 or 2GB of internal RAM, while desktop computers typically have 4 to 8GB of RAM. Mobile applications are loaded into RAM to be executed and memory can rapidly become constrained if mobile applications do not make an efficient use of volatile memory. The result is a smaller number of applications loaded and ready to be executed at a given time. Additionally, there is a penalty in performance when RAM needs to be made available to load new applications when no sufficient memory is available. Thus, mobile apps are expected to make a moderate use of RAM memory, which results in faster loading times and increases the number of applications ready to execute.

Small Battery Capacity: Batteries provide the necessary power for mobile devices to work. While hardware manufactures struggle to increase the capacity of the batteries, mobile application developers struggle to make a more efficient use of them. Recent advancements in software technology revolutionized the way mobile devices are used. Multi-purpose mobile devices (smartphones or tablets) that allow access to email, browse the Internet, watch videos, and play games replaced basic mobile phones. Typical smartphones use batteries that can range from 1,570 to 3,400 mAh for smartphones and around 7,000 mAh for high-end tablets. Energy consumption impacts greatly mobile users. Mobile applications must strive for efficiency in terms of energy consumption in order to maximize battery life. This scenario can be achieved, e.g., by improving the algorithms for saving computation cycles.

Network Bandwidth Limitations: Mobile applications often rely on updated content or data to provide value to end users. Therefore, mobile devices must be capable of accessing content or data over a network. Most mobile devices provide more than one form of connectivity by using different network technologies. Wi-Fi technology (usually seen as a synonym of IEEE 802.11) enables mobile devices access to the Internet and private networks, while 3G is commonly used to access the internet. Other protocols, like Bluetooth or NFC (Near-Field Communication), allow direct connections between mobile devices to share data. The most significant difference of mobile devices when compared with desktop computers is that bandwidth provided by existing technologies is relatively smaller when compared with cable connectivity, especially when compared with optical fiber. Mobile applications must use bandwidth knowing that the throughput is limited by available technologies. Nonetheless, wireless technology is evolving rapidly and new Wi-Fi protocols and recent broadband standards, like Mobile WiMAX and Long Term Evolutions (LTE), are closing the gap between wireless and cable bandwidth technologies (Ethernet), thus allowing mobile applications a wider range of applicability, e.g., streaming video.

Multiple Accessories: Mobile devices are often built with different types of sensors, like GPS, accelerometer, gyroscope, barometer, light sensor and digital cameras. Interactions with the sensors occur frequently and imply that a mobile application needs to comply with the operating characteristics of the sensors, namely the operating range, the sensitivity, the accuracy, or the minimum-polling interval. Accessories are used to provide, complement, and enhance mobile devices functionality, being memory cards a good example. Some mobile devices allow built-in ROM memory to be extended by using a flash memory data device that provides additional storage. Mobile applications must consider the different types of accessories available for mobile devices to fully enhance user experience.

Multiple Network Protocols: As mentioned before, mobile devices provide one or more forms of connectivity. Different technologies can be used to provide network connectivity whether for voice or data. Examples are Wi-Fi and 3G or 4G technologies that allow mobile devices to access to data over a network and are broadly used to provide data access for most mobile applications. Other technologies, like Bluetooth or NFC allow short or very-short distance data transitions and are independent of a service provider to interact with other mobile devices. Typically, using Bluetooth or NFC mobile devices interact directly with other capable mobile devices for data transfer. Mobile applications can take advantage of these technologies by choosing the most convenient for their desired functionality. As an example, Bluetooth is very popular for hand-free devices, like headsets and NFC is expectable to be used to implement contactless payments.

Real-World Interaction: Mobile applications provide real-world interaction by interfacing with context information, like user location, time of day, neighboring users and devices. Mobile devices are capable of acquiring and using context information by using built-in sensors, like GPS and accelerometers, and complementing this information with data acquired from the network. Context awareness in mobile computing is an extensive research field and is emerging as a new computing paradigm, where infrastructure and services are seamlessly available anywhere, anytime and in any format (Hong et al., 2009). Mobile applications are now capable of taking advantage of context information that a few years back was not possible.

Always On: Mobile applications are expected to be promptly, quickly and easily available to end users. Opening or loading a mobile application is expected to be possible in a few seconds. This capability stresses resource utilization, namely processor and memory, resulting in high energy consumption that reduces battery life considerably. Mobile devices are evolving to be capable of being readily available to end users and meet very strict power consumption requirements. Always on devices have their hardware designed with dedicated processors for sensor processing tasks, typically control tasks and signal processing that allow the mobile device to enter 'sleep' mode when the device is not being used and 'wake-up' when a stimulus (signal) is given by the user. Mobile applications must be designed to take advantage of these hardware capabilities that are built into mobile devices to enhance functionality and extend battery life.

Individual and Personal Use: According to Salmre (2005), the major difference between mobile and desktop applications is the way in which people use them. Those who use both desktop and mobile applications tend to use the two in different ways. When using a mobile application, the activities of the user tend to be short. As Salmre puts it: "The user is either responding to being interrupted or using the application to make an immediate request of some other person or process." Thus, user interface takes a great importance for mobile applications, as their users are often seeking to quickly complete a simple task, and cannot take advantage of the full range of functionality provided by a traditional desktop application. Contrarily, users of desktop computers tend to have long sessions.

TYPES OF MOBILE APPLICATIONS

Mobile applications can be classified according to two main types: native applications and web-based applications (table 1) (Charland and Leroux, 2011). Native applications are constructed to be used on a specific platform, like iOS and Android, or mobile device. Mobile web applications consist of web pages specifically developed for taking into account the characteristics of mobile devices.

Table 1. Summary of main characteristics of the three types of mobile applications discussed.

Mobile Web	Hybrid	Native
Runs in mobile web browser Built with HTML5+JS+CSS Served by web server No app store needed Can look like native May be responsive or target specific screen sizes	Runs as locally installed app Built with HTML5+JS+CSS Installed on device In embedded web browser Fed by APIs App stores needed Built for specific devices May be responsive	Runs as locally installed app Built with Objective-C (or Java) Installed on device Fed by APIs App stores needed Built for specific devices

Native apps are designed and coded for a specific kind of device. For instance, iPhone apps are written in Objective-C and Android apps in Java. Native apps can usually be found in an app store, such as Google Play, and require the store's approval. They are downloaded from the store and installed on that mobile device. They run exclusively on that type of mobile device and interact with and take advantage of the features of the operating system and other software that is usually installed on that platform. A native application cannot be used for a different platform without major modifications. For example, an app developed for the iOS runs only on Apple devices.

A native apps tends to be faster and more responsive. Because its code is stored on the phone, there is no delay waiting for static content (such as images and text) to be downloaded from the web. While dynamic content still needs to be accessed from the Internet, it is an improvement over the web-based model in which everything needs to be downloaded each time. In some cases, a native app, like a game, when installed, can work with no Internet connectivity requirements.

Native apps offer better user interface features and perform faster, because they are developed to take advantage of the device's built-in features. They can use, for example, the camera, GPS, accelerometer, compass, list of contacts, etc. Native apps are less prone to security issues when compared to web-based applications as a result of decreased need for network access.

Mobile web apps are developed using technologies that one uses to build websites, such as HTML, CSS, and Javascript. A mobile web application is simply accessed by its URL in a web browser (that runs on the mobile device). In most cases, a web app can be seen as a website that mimics a native app. A major characteristic that distinguishes a mobile web application from a standard website is the fact that it is designed to be accessible in a small display, most of the cases with touch-screen interface capabilities. Hence, operations that in websites are operated with mouse clicks are operated using fingers in a mobile web app.

A major benefit related to mobile web applications is that they operate across different devices and operating systems, as long as a web browser is available. Web-based apps do not require users to install new software or manually search for updates. However, full access to features and resources available on mobile devices is not straightforward, since they are platform agnostic and have no direct access to resources of a mobile device as a native app. These limitations are being addressed, for example, by cloud phone platforms (Taivalsaari and Systä, 2012) that provide solutions for web-based apps to work seamlessly on mobile devices.

A third variation of mobile application can also be considered: *a hybrid application*. A hybrid mobile application takes an HTML-based mobile application and inserts it inside a native wrapper. The core of the application uses HTML, JavaScript and CSS, similar to web applications, but the outside is a native

shell. Hence, the app runs within a native app framework rather than a browser. The hybrid application are also downloaded and installed on a device. Most users cannot distinguish native from hybrid applications.

Hybrid apps are popular since they allow cross-platform development, an approach that reduces development costs. HTML code components can be reused on different mobile platforms.

QUALITY ATTRIBUTES

The successful development of a software application is tightly linked to the ability of managing software requirements. Development of a mobile application is no different; in particular, it is of primal relevance the elicitation and documentation of quality attributes. A quality attribute (also designated quality requirement or non-functional requirement) corresponds to a set of restrictions imposed to the system under development, establishing, for instance, how usable, attractive, fast or reliable it is (Fernandes & Machado, 2015). Examples of this type of quality attributes are security, modifiability, performance, or portability. Quality attributes of mobile applications are highly constrained by the hardware where the software executes. Hardware characteristics establish constraints to mobile applications and quality attributes must be carefully identified and documented to address these constrains, which assume central relevance in architectural design decisions.

Quality attributes should be contrasted with functional requirements that define the behavior of the software system. Quality attributes and functionality are orthogonal, in the sense that many software designs can achieve the same functionality with distinct levels of quality built in. Therefore, software architects focus more on quality attributes rather than on functionality (Fairbanks, 2010, p. 142). In the majority of the cases, a quality attribute is an emergent property, which implies that it is not possible to locate one place in the source code of the software system with the responsibility for ensuring the accomplishment of that attribute.

There are several frameworks to categorize quality attributes of software systems and products. The ISO/IEC 9126 standard is focused on quality, proposing quality attributes, divided in six main characteristics, for evaluating software products: functionality, reliability, usability, efficiency, maintainability, and portability.

The classification scheme suggested by Sommerville (2011, p. 88) differentiates in a first level the non-functional requirements into three categories:

- **Product Requirements** characterize aspects of the behavior of the system itself, including, for example, (i) reliability, (ii) performance, (iii) efficiency, (iv) portability, (iv) usability, (vi) testability, and (vii) readability.
- **Organizational Requirements** come from strategies and procedures established in the context of the manufacturing process of the system or the client organization, being examples (i) process standards that must be followed, (ii) implementation requirements, like the programming language to be adopted.
- **External Requirements** have origin in external factors to the system and the development process, being examples (i) interoperability requirements that define how the systems interact with other systems, (ii) legal requirements to guarantee that the system is compliant with the laws, and (iii) ethical requirements to make sure that the society in general will accept the system.

There are many other proposals for classifying quality attributes suggested by several authors, for example, Boehm et al. (1978), Roman (1985), Grady and Caswell (1987, p. 159), Jureta et al. (2006), and Meyer (2013, p. 703). All these proposals demonstrate the lack of a consensus in the software engineering community with respect to classification of quality attributes. In this chapter, we adopt the eight categories for quality attributes proposed by Robertson & Robertson (2006) for discussing quality attributes in the scope of mobile applications:

- **Look and Feel:** The visual aspect and the aesthetics of the system, namely the graphical interface.
- **Usability and Humanity:** The easiness of utilization of the system and everything that permits a friendlier user experience.
- **Performance:** Aspects of speed, real-time, storage capacity, and execution correction.
- **Operational:** Characteristics about what the system must do to work correctly in the environment where it is inserted.
- **Maintainability and Support:** Attributes that allow the system to be repaired or improved and new functionalities to be anticipated.
- **Security:** Issues related to access, confidentiality, protection, and integrity of the data.
- **Cultural and Political:** Factors related to the stakeholders culture and habits.
- **Legal:** Laws, rules, and standards that apply to the system so that it can operate.

Generically, all categories need to be considered for a given mobile application, as it is a form of a software system. However, mobile devices characteristics imply that some quality attributes become more critical to the success of a mobile application than others. We discuss these topics in the next section.

RELEVANT QUALITY ATTRIBUTES FOR MOBILE APPLICATIONS

This section identifies and discusses the most relevant types of quality attributes for mobile applications and argues that relevant quality attributes for mobile applications are usually framed in Usability, Performance, and Maintainability and Support categories.

As mentioned in the previous section, all quality attributes are relevant and the relevance of these three types does not imply that the other five types of quality attributes are not important or should be discarded when developing mobile applications. In fact, all types of quality attributes have their relative importance, highly depending on the scope or context that motivates the need and use for a mobile application. For instance, security is a transversal aspect for all types of software applications, mobile ones included. Running third-party applications within a mobile device (that stores private information related to its owner) raises privacy and security risks (Leontiadis et al., 2012). Legal requirements are also important, since any system, regardless of the technology, has obviously to respect the established laws. The choice for highlighting Usability, Performance, and Maintainability and Support categories is that these are almost pervasive in what concerns mobile application development and if discarded or considered not relevant, the perception of quality may be deeply affected.

Usability

Usability is a crucial quality for all types of software products. Usability covers many aspects, like ease of use, personalization, ease of learning, and accessibility. According to ISO, usability is related to the extent to which a given product can be used to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use. A usable system means to the user increased productivity, quality of work, and user satisfaction. Usability also implies reductions in support and training costs. An interesting book on usability, edited by Spiliotopoulos et al., (2010), provides a set of methods and approaches that enable and facilitate the development of usable web systems.

Ease of use is related to the efficiency of utilizing a given system and with the mechanisms that exist to reduce the errors made by the users (for example, an automatic spell checker). Here one should take into account the GIGO (garbage in, garbage out) principle, which indicates that software programs that receive incorrect input data, after processing them, generate undesired output data. The best way to avoid garbage in the output is preventing garbage in the input. For example, when a telephone number must be introduced, it is important to ensure that only digits are typed in and that the number of digits is correct (for instance, in Portugal all phone numbers have 9 digits) and that all possible access codes are available to be chosen (avoiding thus typing in them). If these mechanisms are made available, the chances of introducing an incorrect phone number greatly decrease.

Personalization is associated with the capacity of adapting the system to the tastes and needs of the users, including the choice of the language, currency, time zone and other configuration options (colors, backgrounds, icons). Mobile devices are usually tied to a given person, which wishes to introduce his personal data (phone number, email, contacts, passwords, bookmarks, etc) in the device to configure and personalize it.

Ease of learning is concerned with the way users are trained to use the application. Based on requirements of this type, the development team can prepare training and help procedures for the users. Associated with the ease of learning is comprehensibility that determines if the users intuitively capture the functionalities of the application and how to operate it. In many cases, mobile applications are expected to be so easy to use that no external help is required. Some applications must indeed have very high comprehension levels; if this is not the case, the users resort to other alternatives, i.e., mobile applications provided by the competitors, to satisfy their needs.

Accessibility indicates how easy it is to use the system, for people who are somehow physically or mentally disabled. The disabilities can be related with physical, visual, auditory, or cognitive aspects.

Usability is very likely to be the most crucial aspect of mobile applications, since they are used in many different contexts and environments. The user shall be able to use the mobile application in crowded and noisy environments, which imply that restrictions on the inputs and outputs may apply. In many cases, the device is supposed to be handled with just one hand, while in others two hands may be necessary. This means that mobile applications must be developed with the possibility of being effectively used with either one or two hands.

Responsive design is an increasingly important concern for mobile applications. The main challenge is building applications that ensure the same level of user experience, regardless of screen size and orientation of mobile devices. This can imply show, hide or re-arrange graphical elements like number of text columns, adapting font size, resizing or hiding images to maintain the same level of user experience as mobile applications are used in devices with different screen sizes.

Performance

Performance refers to the capacity of a system to respond to its stimulus, that is, the time necessary to deliver a response to an event or the number of events processed in a specific time unit. It is the degree to which a system can accomplish its functionalities within a given set of constraints. The performance of a system is related to the processing time of a task, response time of an operation, accuracy of the result, reliability, availability, fault-tolerance, storage capacity, and scalability.

In addition to the intended functionality, the behavior of a real-time system must respect a set of timing requirements externally defined. The correction of an answer of the system includes also the instant when it is produced. A late answer is incorrect and constitutes a non-conformity, according to the stipulated behavior for the system. For example, the time limit to fire an alarm in the case of a dangerous situation must be scrupulously met; otherwise, it may not be possible to save the persons and the goods that are being monitored.

Accuracy of the results is related to the precision of a calculation computed by the system, how they are stored and delivered. These factors are relevant for values related, for example, with hours, GPS coordinates, money (rounding of bank interests), scientific calculations, and percentages.

Availability quantifies the percentage of time during which a given system is operational and working correctly. It is a measure of the likelihood that a system will be operating when called upon. It is a very important aspect that has ramifications to other issues: confidence of the users in the products that they use, value of the information, efficiency of the processes, and productivity of the organizations. Availability is normally measured by two indicators: MTBF (Mean Time Between Failures) and MTTR (Mean Time To Repair). MTBF represents the predicted elapsed time between inherent failures of a system during operation, while MTTR is the speed with which the system is able to be available again, after the occurrence of a failure. Availability is expressed by the following formula (Stapelberg, 2009, pp. 417).

Availability = MTBF / (MTBF+MTTR)

Availability is of great importance for providing the desired functionality; only when available the system may provide value to its users. For critical systems it assumes a vital role in design as loss of functionality may imply severe losses, whether in terms of money or human life. This characteristic is not however enough, since the system must also be reliable, since a single error may represent severe damages or even the loss of human lives in more extreme situations. Reliability is the capacity of a system to deliver the correct output over time and is measured by the probability of a system producing correct results in a given period of time. Nowadays, there are various businesses that depend totally on software-based systems that support their operation (for instance, airline, bank, insurance, e-commerce, industry), and the expectation is that these systems are “always” available (i.e., that they have short periods of unavailability) and reliable. Reliability is normally measured by the mean time between failures.

Fault tolerance indicates the capacity of the system in keeping an acceptable level of operation, even in undesirable circumstances. The higher the fault tolerance, the bigger the capacity of the system to continue working, even if some of its components fails. Software engineers must be able to build applications that will acknowledge the presence of faults and to include techniques to tolerate these faults, so that the system still delivers an acceptable quality of service (Koren & Krishna, 2010, pp. 1-2).

Data storage capacity specifies the amount of data that the system should be able to process and store. While the function of storing data is a functional requirement, the respective capacity should be seen as a non-functional requirement.

Scalability is the ability of a system to continue to show high quality of service, even when subjected to a greater number of requests. Scalability can be related with the ability to serve more users simultaneously, treat a higher volume of information, or respond to more requests. In any case, it is assumed that this load increase does not imply significant changes to the system response in order for it to maintain the performance levels. Scalability of software systems often needs to be evaluated taking into account the hardware resources, because it is only possible to determine factors such as response time or number of users that can be served simultaneously, if both the hardware and the software are analyzed as a set.

Maintainability and Support

System maintenance, in general, is divided into four main types: preventive, corrective, perfective, and adaptive. The strategy for maintaining systems is limited by several factors (organization, context, technology, laws, business), which determines how the different types of maintenance efforts are developed over the time to make sure that the system remains useful and updated. Thus, in systems in which maintenance is paramount, it is convenient to consider maintenance requirements during the analysis phase.

Modifiability is an attribute strongly related to maintenance and is dependent on how easy it is to locate the system components that must be changed. In principle, it is preferable that the change has impact on a reduced number of components, since it is more costly and difficult to make modifications in the context of many components.

Portability is also an important characteristic to take into account when developing mobile applications. A mobile application is said to be portable if it is relatively easy and straightforward to move it from one mobile device to another. As noted by Silakov and Khoroshilov (2011), the cost of porting an application to different platforms is smaller as developers consider it the sooner during the development. This characteristic is of paramount importance, as today the market has many platforms, being dominated by Google Android, Apple iOS, and Microsoft Windows Phone.

In terms of support, it is important to know, for example, what kind of support and training the users are expected to receive. It may be necessary to provide technical support services or to include personal or passive assistance, e.g., using tutorials built in the software product that explain how to operate it. Today, it is also common to use videos that explain how to use a given product.

Issues, Controversies, Problems

This section discusses what we believe are the most relevant issues currently in mobile applications development. One of the challenges in mobile application development is the one of portability. As discussed in the previous sections, there are several platforms for which one can develop mobile applications. The ability to have a mobile application available for each platform is a decision that directly impacts the cost associated with development and maintenance of mobile applications.

Achieving desired portability can greatly impact the business decision of developing mobile applications. If portability is desired, one may opt to have a mobile application based on the use of web

technologies, but development of the graphical user interface may be the biggest limitation of such approach (Heitkötter et al., 2013) when compared to native language based mobile applications. Also lack of stability in web-based applications seems to be a limitation (Zibula & Majchrzak, 2013). There are now several platforms that support development for hybrid applications and which platforms best satisfy the developers' needs is also a topic (Palmieri et al., 2012).

An increasing relevant issue on mobile application development is efficiency of computation. Although battery efficiency as evolved significantly, it was outpaced by the demand imposed by the software. Mobile applications demand greatly on computing power to provide the desired functionally and this impacts negatively on the use of energy, which decreases the availability of mobile devices. In this topic, developers need to explore how to be more efficient in terms of computation. Energy efficiency might be improved without any changes to hardware by using software solutions that reduce overheads and improve processor utilization (Silven and Jyrkk, 2007).

Mobile applications maintenance and modifiability is an issue as mobile applications have typically a fast update cycle. The need to adjust to customer needs requires frequent updates and evolution in the software. This problem increases when multiple platforms need to be supported and maintainability and modifiability become even more relevant attributes, as cost is multiplied by the number of supported platforms. Design of mobile application must consider upfront the issue of maintenance and modifiability.

SOLUTIONS AND RECOMMENDATIONS

To achieve portability, current solutions focus on using cross-platform frameworks, but with this approach developers need to be aware of its limitations. Mainly, these limitations are related to the quality of the end result in terms of user interface. However, cross-platform frameworks may lower the barrier for developing mobile applications by allowing the use of Web technologies like HTML, CSS and JavaScript, along with the widespread of Web development paradigms. These technologies allow a greater number of developers to easily start developing mobile application without having the effort required to learn native, specific, platform languages, which imply a significant effort to learn (Heitkötter et al., 2013).

Native based mobile applications are probably the best option for developing rich end user interfaces. Native languages seem to provide more flexibility and ease of development when creating the user interface. Also, when designing user interfaces several guiding principles need to be considered to captivate users. Examples of relevant principles to consider are (1) users' limited attention, (2) minimization of keystrokes to completely perform a given functionality, and (3) task-orientation with a minimum set of functions.

In the scope of the mobile devices efficiency issue, developers need to design and develop their software with the mindset of saving computation power, but ensure, at the same time, the same level of functionality. The path to take may involve designing more efficient algorithms to save computation resources and thus as result extend battery life for the same level of functionality. The wide spread of multimedia mobile applications in mobile devices today is an example where improvements on software can help decrease battery consumption by improving the software. Current algorithms for processing media were not designed with the resource limitations in mind and can make a difference (Marius et al., 2010).

FUTURE RESEARCH DIRECTIONS

Our belief is that portability, usability and energy efficiency will drive research of mobile application development. Usability is probably one of the most relevant quality attributes when designing mobile applications. Mobile developers shall try to innovate or differentiate in user experience by researching and developing new user interaction models. These new models will make a significant impact if successful and will allow additional levels of functionality. An example of a possible evolution in new interaction models is augmented reality that certainly will disrupt the range of applicability of mobile applications. An example is augmented reality applied in the automotive industry by allowing a new dimension of driving assistance systems (Gabbard et al., 2014).

Improving energy efficiency on mobile devices will be based in developing approaches to extend battery life. With the increase in the range of functionality made possible by mobiles devices, developers need to be aware that a mobile application is now part of a resource-limited environment. Examples of future research in this topic may consider the Green Software Engineering project that believes software can play an important role in reducing the power used by mobile applications (Rott, 2011).

CONCLUSION

A mobile application is a type of software application, developed to run on a mobile device. The chapter discusses the main characteristics of mobile devices, since they have a great impact on mobile applications. Among those characteristics, one can find for example: small display size, small battery capacity, network bandwidth limitations, and multiple network protocols, always on, and personal use. It also presents the classification of mobile applications according to two main types: native and web-based applications. A third type, hybrid, can also be considered. Building on the information presented, the chapter identifies the most relevant types of quality attributes for mobile applications followed by a discussion on the most relevant issues to be found on mobile application development. Current solutions for the issues presented are discussed and future research directions are identified.

This chapter identifies and discusses relevant quality attributes for mobile applications. It argues that usability, portability, performance and maintainability and support categories are among the most relevant quality attributes for mobile application development. Mobile application development evolved significantly in recent years with technological improvements in mobile devices. These improvements changed the range of applicability of mobile devices and mobile software developers are now taking advantage of these new features and mobile software development will face new challenges and opportunities.

REFERENCES

Boehm, B. W. (1978). *Characteristics of software quality*. North Holland.

Charland, A., & Leroux, B. (2011). Mobile application development: Web vs. native. *Communications of the ACM*, *54*(5), 49–53. doi:10.1145/1941487.1941504

Duffy, T. (2012). *Programming with mobile applications: Android™, iOS, and Windows Phone 7*. Boston, MA: Cengage Learning.

Fairbanks, G. (2010). *Just-enough software architecture: A risk-driven approach*. Boulder, CO: Marshall & Brainerd.

Fernandes, J. M., & Machado, R. J. (2015). Requirements in engineering projects. Springer. doi: 10.1007/978-3-319-18597-2

Gabbard, J. L., Fitch, G. M., & Hyungil, K. (2014). Behind the glass: Driver challenges and opportunities for AR automotive applications. *Proceedings of the IEEE*, *102*(2), 124–136. doi:10.1109/JPROC.2013.2294642

Grady, R., & Caswell, D. (1987). *Software metrics: Establishing a company-wide program*. Upper Saddle River, NJ: Prentice Hall.

Heitkötter, H., Hanschke, S., & Majchrzak, T. A. (2013). Evaluating cross-platform development approaches for mobile applications. *8th International Conference on Web Information Systems and Technologies* (LNBIP), (vol. 140, pp. 120–138). Springer. doi:10.1007/978-3-642-36608-6_8

Hong, J., Suh, E., & Kim, S. J. (2009). Context-aware systems: A literature review and classification. *Expert Systems with Applications*, *36*(4), 8509–8522. doi:10.1016/j.eswa.2008.10.071

Juárez-Ramírez, R., Licea, G., Barriba, I., Izquierdo, V., & Angeles, A. (2012). Orchestrating mobile applications: A software engineering view. In R. Aquino-Santos & A. E. Block (Eds.), *Embedded systems and wireless technology: Theory and practical applications* (pp. 41–72). Boca Raton, FL: CRC Press. doi:10.1201/b12298-3

Jureta, I. J., Faulkner, S., & Schobbens, P. Y. (2006). A more expressive softgoal conceptualization for quality requirements analysis. Lecture Notes in Computer Science, 4215, 281–295. doi:10.1007/11901181_22

Koren, I., & Krishna, C. M. (2010). *Fault-tolerant systems*. San Francisco, CA: Morgan Kaufmann.

Leontiasis, I., Efstratiou, C., Picone, M., & Mascolo, C. (2012). Don't kill my ads!: Balancing privacy in an ad-supported mobile application market. *12th Workshop on Mobile Computing Systems & Applications*. ACM. doi:10.1145/2162081.2162084

Marcu, M., Tudor, D., & Fuicu, S. (2010). A view on power efficiency of multimedia mobile applications. In K. Elleithy (Ed.), *Advanced Techniques in Computing Sciences and Software Engineering* (pp. 407–412). Springer; doi:10.1007/978-90-481-3660-5_70

Meyer, B. (2019). *Touch of class: Learning to program well with objects and contracts*. Berlin: Springer.

Robertson, S., & Robertson, J. C. (2006). *Mastering the requirements process* (2nd ed.). Boston, MA: Addison-Wesley.

Roman, G. (1985). A taxonomy of current issues in requirements engineering. *IEEE Computer*, *18*(4), 14–23. doi:10.1109/MC.1985.1662861

Rott, J. (2011). *Developing green software*. Retrieved from https://software.intel.com/sites/default/files/developing_green_software.pdf

Salmre, I. (2005). *Writing mobile code: Essential software engineering for building mobile applications*. Boston, MA: Addison-Wesley.

Silakov, D. V. & Khoroshilov, A.V. (2011). Ensuring portability of software. *Programming and Computing Software*, *37*(1), 41–47. Springer. doi 10.1134/S0361768811010051

Silven, O. & Jyrkk, K. (2007). Observations on power-efficiency trends in mobile communication devices. *EURASIP Journal on Embedded Systems*. doi 10.1155/2007/56976

Spielberg, R. F. (2009). *Handbook of reliability, availability, maintainability and safety in engineering design*. London: Springer.

Spiliotopoulos, T., Papadopoulou, P., Martakos, D., & Kouroupetroglou, G. (2010). *Integrating usability engineering for designing the Web experience: Methodologies and principles*. Hershey, PA: IGI Global. doi:10.4018/978-1-60566-896-3

Spriestersbach, A., & Springer, T. (2004). Quality attributes in mobile web application development. Lecture Notes in Computer Science, 3009, 120–130. doi:10.1007/978-3-540-24659-6_9

Taivalsaari, A., & Systä, K. (2012). Cloudberry: An HTML5 cloud phone platform for mobile devices. *IEEE Software*, *29*(4), 40–45. doi:10.1109/MS.2012.51

Wasserman, A. I. (2010). Software engineering issues for mobile application development. *FSE/SDP Workshop on Future of Software Engineering Research*, (pp. 397–400). doi:10.1145/1882362.1882443

Zibula, A., & Majchrzak, T. (2013). Cross-platform development using HTML5, jQuery Mobile, and PhoneGap: Realizing a smart meter application. *Lecture Notes in Business Information Processing, 140*, 16–33. doi:10.1007/978-3-642-36608-6_2

KEY TERMS AND DEFINITIONS

Mobile Application: A type of software application developed to run on a mobile device. Also named mobile app.

Mobile Device: A computing device that is made for portability, and is thus both compact and lightweight. Smartphones and tablet computers are examples of mobile devices. Recent technological advances have allowed these small devices to be able to perform nearly anything that had previously been accomplished with personal computers. Also named handheld computer.

Quality Attribute: A set of restrictions imposed to the system under development, establishing, for instance, how usable, attractive, fast or reliable it is. Examples of this type of quality attributes are security, modifiability, performance, or portability. Quality attributes of a mobile application are highly constrained by the hardware where the app executes. Also designated quality requirement or non-functional requirement.