Beta Testing of a Mobile Application: A Case Study

MATEJA KOCBEK AND MARJAN HERIČKO, University of Maribor

Beta testing is the last stage of testing, and normally involves sending the product for beta testing and real-world exposure outside the company. Beta testing is often preceded by a round of testing called alpha testing. Beta testing can be considered a form of external user acceptance testing. Software in the beta phase will generally have many more bugs in it than completed applications, as well as speed and performance issues that may still cause crashes or data loss. Beta testing is the first opportunity to get real feedback from target customers. The launch of a mobile application is especially crucial because it is the single biggest opportunity to get an application discovered in the mobile markets. In this article, the beta testing of mobile applications is presented. Our aim is to identify the optimal number of testers, who can reveal the majority of errors and mistakes during beta testing. The findings were obtained through the case study research method.

Key Words and Phrases: alpha testing, beta testing, software testing, acceptance testing

1. INTRODUCTION

The software development life-cycle has several phases, which should be conducted in the following order: analyse requirements, design, development, integration and testing, deployment and maintenance. We will focus on the testing phase.

However, the successful introduction of integrated systems into businesses greatly depends on whether we trust the system or not. Trust in the quality of the software-based system is determined by many factors, such as: completeness, consistency, maintainability, security, safety, reliability, and usability. However, during the development of a software-based system, there are many opportunities to find errors in the different phases of the software development lifecycle. Testing is commonly applied as the predominant activity to ensure high software quality, providing a wide variety of methods and techniques to detect different types of errors in software systems [Budnik 2012].

Software bugs will almost always exist in any software module: the complexity of software is generally intractable and humans only have a limited ability to manage complexity. Thus, design defects can never be completely ruled out, especially in complex systems [Pan 1999].

Testing is usually performed for the following purposes: (1) to improve quality, (2) for verification and validation, and (3) for reliability estimation. *To improve quality* refers to the conformance of the specified design requirements. Being correct, the minimum requirement of quality, means performing as required under specified circumstances. Debugging, a narrow view of software testing, is intensively performed to discover design defects by the programmer. The imperfection of human nature makes it almost impossible to make a moderately complex program correct the first time around. Another important purpose of testing is *verification and validation*. Within the process of verification and validation, testing can serve as a metric. Software reliability has an important connection with many aspects of the software, including structure and the amount of testing it has been subjected to. Testing can also serve as a statistical sampling method to gain failure data for *reliability estimation* [Pan 1999].

As previously mentioned, the process of testing software is very complex. For this reason there are many types of software testing. One of the formal types of software testing is alpha testing. Usually it is performed by potential users/customers or an independent test team at the development site. Alpha testing is conducted before taking the software to beta testing [Pradhan 2012].

In our paper, the actual beta testing of mobile applications will be presented. Due to the reliance on alpha and beta testing, these two types of testing will be presented as well. The main aim of the article is

Author's address: M. Kocbek, Institute of Informatics, Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova ulica 17, 2000 Maribor, Slovenia; email: mateja.kocbek@um.si; M. Heričko, Institute of Informatics, Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova ulica 17, 2000 Maribor, Slovenia; email: marjan.hericko@uni-mb.si.

In: Z. Budimac (ed.): Proceedings of the 2nd Workshop of Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA), Novi Sad, Serbia, 15.-17.9.2013, published at http://ceur-ws.org

to identify what the optimal number of testers is to reveal the majority of defects in a mobile application. In the second chapter, the background of the case will be briefly described. Some facts about mobile applications, alpha and beta testing, and the case study will be given. The third chapter is about the case: all the details of the beta testing of the actual project are given and research question is discussed. In the end, we present our conclusions.

2. BACKGROUND

In this chapter, the background of all content areas is given.

2.1. Mobile application

The era of mobility and mobile applications has arrived, and it demands the rapid development and delivery of high-quality solutions. Customers expect an exceptional mobile experience and competition in the mobile market gives them a wide range of choices. The consumer world has driven unprecedented changes in IT and now IT needs the practices, delivery models, and management solutions to make it work. The unprecedented increase of mobile app development has brought a new set of challenges that engineering and quality assurance (QA) teams need to overcome in order to keep up with this rapid rate of growth [HP 2012].

Mobile application is software designed to use it on smart devices. The mobile applications are available to the end users through mobile markets. Mobile applications are actual applications that are downloaded and installed on mobile devices, rather than being rendered within a browser. The application may pull content and data from the internet [Summerfield 2013]. A mobile application is software that runs on a device that can connect to a wireless carrier network and has an operating system that supports standalone software.

There are two types of mobile applications. The first are native applications. This is an application that runs on a handheld device (phone, tablet, e-reader, iPod Touch) which has a “smart” operating system which supports standalone software and can connect to the internet. Usually native applications are downloaded from app stores such as the Apple Store, Google Play, etc. A native application can only be “native” on one type of mobile operating system. This is why, for example, an iPhone application only works on iOS devices. Native applications are designed to run on a device’s operating system and machine firmware, and typically need to be adapted for different devices [MobiThinking.com 2013]. By contrast, a mobile web application is software that uses technologies such as JavaScript or HTML5 to provide interaction, navigation, or customization capabilities. These programs run within a mobile device’s web browser. This means that they are delivered wholly on the fly, as needed, via the internet; they are not separate programs that get stored on the user’s mobile device [Gahran 2011]. In a web application, or also a browser application, all or some parts of the software can be downloaded from the Web each time it is run. It can usually be accessed from any web-capable mobile device [MobiThinking.com 2013].

Mobile application development is also a special challenge. The nature of the mobile platform for which an application is developed requires a great deal of planning and consideration from the developer. Mobile devices have unique characteristics, such as: bandwidth, processor speed, screen size and resolution, memory consumption, battery life, and user input tools. These are issues that carry a certain level of importance in a desktop application as opposed to mobile applications [Mahmoud and Popowicz 2010].

Given the extreme pace and unpredictability of today’s mobile market, maintaining application quality has become a daunting task. The complexities of multi-platform development coupled with the ever-growing number of models, operating systems, screen sizes, and network technologies make it practically impossible to keep your mobile apps and services in sync with the ever-changing landscape of technology. To address these business and technological challenges, organizations need to implement a mobile testing strategy for the testing team [HP 2012].

2.2. Alpha and beta testing

Every company or organization follows its own software testing life-cycle. We focus on a specific stage of software testing called beta testing.

While the alpha testing is done on the side of the developer, beta testing is the first actual user test of software that provides enterprise software manufacturers with end-user usability and software functionality feedback. Beta testing begins in the last phase of the software development cycle and lasts until the final release of the software. Software developers select special users or user groups to validate software and provide tools to collect feedback. The purpose of beta testing is to enhance the product prior to general availability. Major software manufactures are focusing on improving the quality, acceptance and experience of enterprise software by promoting beta testing programs. It should be noted that there are no existing standards or models for beta software testing [Buskey 2005].

Beta testing is the most effective process to generate information about invalid or empty inserts and other similar scenarios. Beta testing is executed using a black box technique, with no direction or instructions from software development. Beta testing is also a process that extends the validation and testing phase during the software development life cycle. Beta testing strengthens the validation and testing phase and fosters the deployment phase by assuring the product is fully tested [Buskey 2005].

Beta testing is an essential component of the software validation and testing phase because of its immediate impact on the product being tested. It provides a platform to integrate real-world feedback into a product, prior to availability. This collectively improves the software usability and functionality, which lowers the potential cost and improves quality [Fine 2002]. The impact of beta testing on the software industry is important and global standards are required to manage this process [Buskey 2005].

2.3. Case study

Case study research is conducted in order to investigate contemporary phenomena in their natural context. That is, no laboratory environment is set up by the researcher, where factors can be controlled. Instead, the phenomena are studied in their normal context, allowing the researcher to understand how phenomena interact with the context. The selection of subjects and objects is not based on statistically representative samples. Instead, the research findings are obtained through the in-depth analysis of typical or special cases [Runeson and Höst 2008].

Case study research is conducted by iteration over a set of phases. In the design phase, the objectives are decided on and the case is defined. Data collection is first planned with respect to data collection technique and data sources. Methods for data collection include interviews, observation and the use of archival data. During the analysis phase, insights are both generated and analysed, e.g. through the coding of evidence from the findings to the original data. The report should include sufficient data and examples to allow the reader to understand the chain of evidence [Runeson and Höst 2008].

3. ABOUT THE PROJECT

3.1. Case study design

The focus of this study is the beta testing of mobile applications. Beta testing can be considered to be a form of external user acceptance testing. Software in the beta phase will generally have many more bugs in it than completed applications, as well as speed and performance issues that may cause crashes or data loss. Beta testing is the first opportunity to get real feedback from target customers, because the beta testing team is independent from the development team. Pre-beta testing should be noted too. Pre-beta testing is the phase between the alpha and beta testing phase, although there is no exact definition of this term.

The authors of this article worked on a project that is described through this case study. That is why the method for collecting data is descriptive. This was the only used method. In the study, only qualitative data is used and analysed.

Our research question is: How many testers are needed to reveal the majority of defects of a mobile application during beta testing?

There is a connection to existing literature, which uses basic mathematical principles to describe and summarize given facts. In study [Nielsen 2000], facts are given about how many users are needed and how many errors can be found during testing. The author's intention was to find the optimum number of users/testers to publish a stable mobile application, where 85% of errors are detected.

3.2. About the project

As previously mentioned, mobile applications can be developed during a relatively long process, called the software development process. From this perspective, they are just the same as regular desktop or web applications. The project from our case is a project developed at the Institute of Informatics, at the Faculty of Electrical Engineering and Computer Science, at the University of Maribor. The development process was nearly one year long. All phases of the software development process were carried out. There was a group for Android development, which was separate from the group for iOS development. Every group had three members – software developers. A very important part of the project was also the group for design, comprised of three designers. The internal group for testing had about five members, but there was also an external group of testers.

3.3. Performing beta testing

The project roughly described before had three versions. The first version was published in November 2012, the second in March 2013 and the third in July 2013. For every version of the application, comprehensive testing was necessary. Once the application was published, there was no way back. The process of internal testing was essential. The internal group for beta testing was receiving daily builds. The daily builds were reviewed in detail. All detected errors, deficiencies, inconsistencies and feature requests were logged with the system Flyspray [Flyspray 2012]. System enables to log several characteristics, like: reported version, place, severity, priority, name of the tester, etc. The system was used by the internal group for beta testing and the developers. This was the process on a daily basis. The external group for beta testing obtained the build every week. They had to report about their tests to the internal group for beta testing. They checked all existing defects and noted ones that did not arise before. At each stage before every release, the internal and external group had to report issues.

As mentioned before, the group for beta testing was essential for our article. Testers were performing automated and also manual testing on real devices and in different environments. The main goal of every group of testers was to identify as many defects as possible to ensure a standalone application. To ensure this condition, some essential resources were needed, the most important being human resources. How many people were actually needed? The source [Nielsen 2000] reveals that the best results come from testing with only 5 users/testers. The number of defects (*X*) found at testing with n users is:

$$X = N\,(1\text{-}(1\text{-}L)\ n), \qquad (1),$$

where N is the total number of detected problems in the design and L is the proportion of defects discovered while a single user is testing. The typical value of L is 31%, averaged across a large number of projects. Plotting the curve for L = 31% gives the following results: The most striking truth is that zero users give zero insights (Equation 1). As soon as data from a single test user is collected, the insights dramatically rise and almost a third of all defects are found. The difference between zero and even a little bit of data is astounding. The second user discovers some of the same things as the first user, so there is some overlap. People are definitely different, so there will also be something new that the second user does that was not observed by the first user. Thus, the second user adds some amount of new insight, but not nearly as much as the first user does. The third user will do many of the things already observed by the first or second user and even some things that have already been caught twice. Of course, the third user will generate a small amount of new data, even if it is not as much as the first and second users have. By adding more and more users, less and less defects are discovered because the same objects are repeated. After the fifth user, basically the same findings are discovered [Nielsen and Landauer 1993; Nielsen 2000].

The project was performed on the basis of described theory. The whole group for beta testing (internal and external together) consisted of 10 testers. Before testing, they received instructions regarding which functionality needed to be tested. With every functionality, black box testing was performed. Every tester performed a number of tests every day, wherein the Flyspray system was the main logging tool. Every defect detected by the tester was noted there. A tester had to be up-to-date with the situation on the Flyspray, because there was no need to duplicate defects.

In order to facilitate data analysis, we will concentrate on the last week before the last existing version of our mobile application was published. The discovered defects were classified according to whether they were a contextual or technical problem. All testers together found 39 defects (29 errors, 10 deficiencies and inconsistencies). According to the classification, 62% of all faults were technical problems while the rest were contextual. Duplicates were not included. To validate our theory, we first focused on the first 5 testers. The results are given in *Table 1*. Tester A found 17 errors. Tester B found 12 errors, in which 5 of them were new. Tester C found 3 new errors, tester D found 2 and tester E found 1 new defect. The number of new defects is declining fast. The biggest difference is between the first two testers. Tester A found 43 % of all defects found in this particular week.

Table 1: Number of defects of first five testers

	Number of defects	Number of new defects
Tester A	17	17
Tester B	12	5
Tester C	10	3
Tester D	7	2
Tester E	6	1

The rest of the testers from F to J found a minor number of defects (*Table 2*). But more than the number of defects are significant new defects. *Table 2* shows that the rest of the testers from F to J found only one new defect.

Table 2: Number of defects of second five testers

	Number of defects	Number of new defects
Tester F	5	1
Tester G	4	0
Tester H	3	0
Tester I	1	0
Tester J	1	0

Testers from A to E found altogether 97 % of all the defects found. This number represents the great majority of all defects found. In the report of the beta testing group, there were also a lot of defects, which were duplicates or not real defects. Beta users did not know all the requirements for software requested by the customer. This is the reason why some defects were not included in the analysis.

Our research confirmed numbers and facts from [Nielsen 2000] which we can easily answer the research question stated above. Five testers can reveal the majority of defects, while adding an additional tester does not add a crucial performance boost. Although a sixth tester can enhance the ratio of detected defects, this difference is not crucial. With the current project we revealed that the optimal number of testers when performing beta testing is five. With such a number of testers, work efficiency and the optimal consumption of resources can be ensured.

3.4. Related Work

Beta testing in the mobile domain is a relatively undiscovered area. Since mobile applications are different from traditional ones, they require different and specialized new techniques for testing [Kirubakaran and Karthikeyani 2013]. Beta testing is one of the phases where mobile applications are verified. Some basic facts about beta testing in general are given in [Buskey 2005]. Another type of testing of mobile applications is described in [Long 2010]. In [Muccini et al. 2012] directions for mobile testing automation are given. Similar content to that found in our article was not found.

CONCLUSION

We can conclude that a group for beta testing is essential in a stand-alone project that is waiting to be published. Another important perspective is also the need for iterations. The described process needs to be performed all over again, in order to gain effectiveness. This is especially important if the functionalities change or if the customer does not know exactly what to offer to the user. That is why the specifications for software are very important document for customers, developers and especially for testers, who ensure quality. Our case study can confirm that the optimal number of testers is roughly five. Adding more testers to a group does not ensure additional quality.

REFERENCES

BUDNIK, C., 2012. Software Testing, Software Quality and Trust in Software-Based Systems. *2012 IEEE 36th Annual Computer Software and Applications Conference*, pp.253–253.

BUSKEY, C.D., 2005. A Software Metrics Based Approach to Enterprise Software Beta Testing Design. *Pace University*, pp.1–268.

FINE, M.R., 2002. *Beta Testing for Better Software*, Wiley.

FLYSPRAY, 2012. Flyspray. Retrieved September 2, 2013 from http://flyspray.org/.

GAHRAN, A., 2011. What's a mobile app? Retrieved September 2, 2013 from http://www.contentious.com/2011/03/02/whats-a-mobile-app/.

HP, 2012. HP UFT Mobile accelerates automated mobile testing.

KIRUBAKARAN, B. AND KARTHIKEYANI, V., 2013. Mobile application testing — Challenges and solution approach through automation. *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering*, pp.79–84.

LONG, X., 2010. Adaptive random testing of mobile application. *2010 2nd International Conference on Computer Engineering and Technology*, pp.297–301.

MAHMOUD, Q.H. AND POPOWICZ, P., 2010. A Mobile Application Development Approach to Teaching Introductory Programming. *Frontiers in Education Conference (FIE), 2010 IEEE*, pp.1–6.

MOBITHINKING.COM, 2013. Mobile applications: native v Web apps – what are the pros and cons? Retrieved September 2, 2013 from http://mobithinking.com/native-or-web-app.

MUCCINI, H., FRANCESCO, A. DI AND ESPOSITO, P., 2012. Software Testing of Mobile Applications: Challenges and Future Research Directions. *Automation of Software Test (AST), 2012 7th International Workshop on*, pp.29–35.

NIELSEN, J., 2000. Why You Only Need to Test with 5 Users. Retrieved September 2, 2013 from http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

NIELSEN, J. AND LANDAUER, T.K., 1993. A mathematical model of the finding of usability problems. *Proceedings of ACM INTERCHI'93 Conference*, pp.206–213.

PAN, J., 1999. Software Testing. Retrieved September 2, 2013 from http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/.

PRADHAN, T., 2012. All Types of Software Testing. Retrieved September 2, 2013 from http://www.softwaretestingsoftware.com/all-types-of-software-testing/.

RUNESON, P. AND HÖST, M., 2008. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2), pp.131–164.

SUMMERFIELD, J., 2013. Mobile Website vs. Mobile App. Retrieved September 2, 2013 from http://www.hswsolutions.com/services/mobile-web-development/mobile-website-vs-apps/.