Performance Engineering and Testing

The Challenges on Mobile Platforms

Pavlo Bazilinskyy
University of St Andrews
School of Computer Science
pb52@st-andrews.ac.uk

Markus Brunner
University of St Andrews
School of Computer Science
mb246@st-andrews.ac.uk

ABSTRACT
This paper discusses the challenges which mobile devices raised in terms of Performance Engineering and Perfomance Testing. Possible improvements and adaptions to increase the awareness of importance of both areas are presented.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics

Keywords
Software Optimisation, Perfomance Engineering, Performance Testing, Software Metrics

1. INTRODUCTION
Performance is a universal quality of software that is affected by all aspects of the code, design and means of execution. A recent study reports that at least half of executives of IT companies have encountered performance issues in as many as 20% of their projects [1]. Acceptance of software products and how successful they are depends largely on experience that users of the system receive from using the applications and poor performance is often linked with dissatisfaction and frustration of clients. Ensuring high performance of software systems is especially critical in mobile devices.

Adoption of mobile devices is the fastest technology adoption curve in history. Mobile application market is expected to reach $25 billion by year 2015 [10]. Success of applications deployed to mobile devices depends on their responsiveness: in case of poor performance of the application levels of revenue, brand and customer loyalty decline [8]. Moreover, as reported by IBM, 66% of mobile device users are less likely to purchase goods or services from a company following poor performance of the company's mobile application [7].

Clearly, performance is crucial for software solutions targeted to mobile applications market. This paper focuses on Performance Engineering as a whole and on its practical application during the development process - Performance Testing. The focus of this work is on how Performance Engineering and Testing can be organized and optimised on mobile platforms whose advent over the course of the recent years entailed new challenges for software manufacturers.

2. BACKGROUND
2.1 Performance Engineering
Software Development for mobile platforms is one of the the fastest growing fields of Software Engineering. The connection between business recognition and application success when applied to mobile-based software is now becoming apparent [17]. User requirements for software performance are often neglected before it is too late and required changes become costly [13]. Similarly to other fields of Software Engineering, Performance Engineering is limited by intense schedules, badly designed requirements, and over-optimistic objectives [19]. However, these days it is becoming common knowledge that all types of software have specific performance requirements that need to be taken into consideration on the stage of conceptualizing.

Working on undesired changes and going over budget may be avoided by following principles of Software Performance Engineering or just Performance Engineering, which is focused on software system performance and scalability [13]. The objective of this field of Computer Science is to meet responsiveness goals through modeling software requirements and variations of design. Received models are then used to measure and evaluate the expected performance. Evaluation is done by estimating trade-offs in functions, size of hardware, accuracy of obtained results, and resource requirements. If the results of testing do not meet the requirements, the process is repeated with different models [15]. This series of actions starts on the design stage, but it also continues during the implementation stage of the project. Performance Engineering ensures that more accurate models of software and its performance estimates may be developed.

Performance Engineering applied to software development normally includes all of the following activities [19]:

Identify concerns : the qualitative evaluation of influences of performance goals.

Define and analyse requirements : by using UML or special scenario languages outline the operational profile, estimate workload, delay and throughput require-

ments, and scenarios describing what the system is expected to output.

Predict performance : by utilising scenarios, architectures, design outlines estimate expected performance by modeling the interaction of the behaviour with the resources.

Performance Testing : check performance of the system under normal and stress conditions. This paper focuses on this aspect of Performance Engineering.

Maintenance and Evolution : estimate the effect of potential changes and additions, such as added features, migration to a new platform, migrating to new web application platforms.

Total System Analysis : plan how software will behave in the deployed system.

Further, the latest report of Eurostat indicates that most countries in Europe have more mobile subscriptions than inhabitants [2]. Our generation is experiencing the evolution of the mobile platform: to succeed we need to constantly rethink how we comprehend ways in which customers and employees communicate with us, and universal access to information is in great demand now [4]. However, traditional approaches that are utilised in Software Performance Engineering may no longer be applied to the mobile platform - the rate of progress in research connected to Performance Engineering is much slower than that of evolution of the mobile platforms. Additionally, analysing the software performance of software created for mobile devices is complicated because software architects usually need to find equilibrium between the building blocks of Performance Engineering, performance and scalability, and other quality-of-service attributes such as manageability, interoperability, security, and maintainability [9]. Nevertheless, this is an important research area due to popularity of mobile devices in today's world.

2.2 Performance Testing

As mobile devices are characterised by limited performance capabilities (e.g. memory, battery life, computing power, etc.), Software Optimisation and Performance Engineering are two crucial elements in the software development process. The performance progress of mobile platforms in the recent years was imposing, resulting in even more powerful devices. However, energy is the limiting factor which sets the border for further progression. Application developers must understand the trade-offs between performance and battery life in order to optimise resource utilisation [16]. This fact emphasises the importance of Performance Testing on mobile platforms. The practice shows that a majority of application developers and software houses which are deploying the applications put only limited focus on this aspect. Owners of an early version of Apple's iPhone will probably already have faced the challenge of installing and using a new application for their 2-4 year old phone. This very short life cycle derogates the users' experience and takes the wind out of the sails of the mobile platforms' advance.

Weyuker et. al claimed that reasons for the lack of Performance Testing in order to produce resource-efficient software can often be attributed to "the lack of performance estimates, the failure to have proposed plans for data collection, or the lack of a performance budget" [18]. Furthermore his experience showed that a lack of sufficient planning for performance issues is another precursor for bad software. Even though these findings were determined already 13 years ago, we can project them onto nowadays' mobile app development. One of the reasons for a lack of Performance Testing on mobile platforms is the fact that applications can be developed by third-parties which do not have to fulfill certain quality requirements and standards.

Amongst the countless magnificent app-concept that were developed by third-party developers (e.g. groups of students, programmers, freelancers, start-ups, etc.), there are very few project teams which have taken Performance Testing into serious consideration. Applications which permanently strain the resource pool of a mobile device have a bad influence on the device's energy consumption and are subsequently shortening the battery life. A solution could be to restrict developers with imposing stricter developing guidelines or quality standards on them. However, this approach would lead to a decline of diversity in terms of widely available and easy accessible SDKs (Software Development Kits). Considering that many apps were developed in home offices or dorm rooms, this would lead to a loss of creativity and dynamics in the developer community.

3. RELATED WORK

Literature on Performance Engineering in mobile devices is limited. We think it may be explained by the fact that this area of research is still seen as a cutting-edge field of software engineering and measuring performance of projects developed for mobile platforms is currently a developing area of research. There are virtually no literature sources that address the issue of applying performance engineering to mobile development projects. However, ideas on how to approach the issue could for example be received from Harman an Smith's work, who review the key ingredients required for successful Software Optimisation and describe the benefits that are likely to acrue from growing work in the field, while considering problems and challenges for future work. [5]

In regards of Performance Testing, research has been conducted on the generation of frameworks and performance metrics. However, there is very few literature which deals with delivering these concepts into practice.

4. METHOD

Similarly to the situation with published research papers that touch the issue of maintaining Performance Engineering in mobile devices, a pool of methods that are available for addressing the issue is also limited. After finishing our literature review and realising that the scientific community is not fully aware of the problems of Performance Engineering and Testing with mobile software we decided to focus our writing on outlining existing problems. Additionally, our main concern about methods used to implement performance analysis of the process of engineering mobile applications is that they are mostly created using agile development techniques. Performance Engineering is overly complicated when they are used. Results obtained from analysis of performance may be improved by setting multiple objectives. We propose using

Search Based Software Engineering (SBSE) for investigating what objectives may be defined for a project that deals with mobile devices.

5. IMPROVING THE PERFORMANCE ON MOBILE DEVICES

5.1 Performance Engineering

The work of Smith and Williams points out that Software Performance Testing should be started early in the software development process: as early as at the stage of conceptualising the system [13]. This approach works best with systems that have safety as the main attribute. However, due to limited access to resources mobile software companies tend to focus on the commercial side of their projects and neglect Performance Engineering until problems with their published on the market applications start to arise. One may argue that Performance Engineering should be integrated into mobile software development process, rather than it being a mere addition to the process [14].

Additionally, mobile development projects tend to have short time dedicated to the actual development phase. Agile development is commonly used with mobile development to produce multiple releases that incorporate small changes. Problems with performance in projects often deployed to multiple mobile platforms are difficult to notice. Therefore, Performance Engineering may be an important asset to projects created in this way. The problem of measuring performance of the final mobile software product received through following rapid development is defined by Barber [3]. One may find it hard to add Performance Engineering into the agenda outlined for the project that is created using agile development. This issue may be addressed by defining precise, quantitative performance objectives as explained by Smith and Williams [14]. They help to explicitly outline what is expected from the system and once the milestone is reached, quantitatively determine whether the software meets that objective. Developers may also wish to define more than one objective. During the modeling process yielded by the model results should be compared to what is expected from a particular objective. It helps to realise if the project faces a risk of not meeting the objective. If that is a case, appropriate actions should be taken. As soon as results of performance tests can be obtained, one is able to see whether or not created software meets the objective.

The objectives for the project may be obtained by using SBSE. Its main objectives could be defined as: 1. Choose the representation of the problem. 2. Define the fitness function [5]. The fitness function determines the best solution by using the search algorithm of choice that can differentiate between solutions and quantify achieved progress [6]. SBSE is a relatively young field of science and its revolutionary capabilities may be used to optimise Performance Engineering so that it could be adapted to development of mobile devices.

5.2 Performance Testing

The motivation for Performance Testing is often not existent amongst developers, since the users' experience in regard to performance plays just a minor role in early stages of the software development process. The outcomes of a missing Performance Testing component stay latent as long as the produced applications are deployed to a set of homogeneous devices as there is no difference in performance between them. We have seen this phenomenon during the desktop era, when computers became more and more capable in terms of data processing and memory. The application and hardware lifecycles were aligned to each other - as hardware got old, applications got old and were replaced by newer versions.

The contemporary computing environment is in contrast characterised by a variety of different platforms and technologies which differ in terms of performance, mobility, energy efficiency, form factor, etc. Just a few years ago people used a single desktop computer or laptop in their homes and replaced this device when they thought that the time had come to move on to a better product. This situation has radically changed - these days people use a mix of laptops, mobile phones, desktop computers and tablets for different tasks. These devices feature different characteristics as they are designed to serve different purposes.

These diversified characteristics result in semi-latent performance issues which are highly influencing the users' experience. Semi-latent therefore, because many applications do not consider specific technological capabilities of the target platforms. A data-intensive video application for a mobile phone might run smoothly on the latest hardware but could cause problems on an older version of it. The cause of this problem is enrooted in the evolution of the programming languages and their frameworks which were developed towards a higher level of abstraction in order to be able to ignore the underlying hardware. This development might have made the lives of the developers easier but has its downsides in regard to scalability and resource-efficiency. Schmidt claimed that today's programming concepts provide abstractions of the solution space (i.e. the hardware itself) rather than abstractions of the problem space in terms of application domains (e.g. health care, insurance, biology, etc) [12].

There are two stages which have to be fulfilled in order to integrate Performance Testing into the development process. Firstly, software requirements (i.e. metrics) have to be clearly defined. These metrics are the ground on which Performance Testing can be build upon. The "body of research and practice in developing metrics for traditional software" is rich; however, there has been little research on how these metrics can be related to mobile applications [11]. Once these metrics have been established, it comes to the implementation stage, where the performance of applications has to be evaluated and tested against the defined metrics.

There is a substantial problem which can be detected when it comes to Performance Testing. Firstly, developers test their software on modeled pre-deployment environments, which are just another form of abstraction of the production environment. Moreover, performance is usually tested after the development process rather than being continuously assessed during it. Improvements on that can be achieved by tightening the coupling between Software Development and Performance Testing. According to Thompson, Model-driven Development (MDE) is a promising solution to this problem [16]. The concept of MDE is to develop and test

software in early stages of the design process to identify key characteristics, such as power consumption, memory requirements, etc.

The challenges of Performance Testing are researched and theoretical concepts have been developed. Although there is still much space for improvements and more research, since these concepts have to be delivered to the mobile application developers. Integrated Development Environments (IDEs) still lack of support for Performance Testing, but with further advance of mobile devices, developers will not be able to avoid the adoption of Performance Testing, at least if they want to succeed in the competition. Not only the integration of Performance Testing frameworks and tools in IDEs has to be improved, but the importance of Performance Engineering and Testing has to be propagated in the whole field of Software Engineering.

6. CONCLUSION

We briefly touched a problem of Performance Engineering and Performance Testing in projects that deal with applications intended for mobile devices.

Software used in mobile devices is normally created using agile development techniques and it is difficult to use conventional means of assessing and improving performance due to the nature of frequent updates submitted by the team working on the project. We suggest improving performance of mobile applications by outlining multiple precise, quantitative performance objectives. Results received during the modeling process are then compared to what a particular objective requires. This process can indicate that the project is not likely to meet the objective. If that is true, appropriate actions can be taken. Objectives defined for projects may be created using Search Based Software Engineering techniques.

We assessed that Performance Testing is not consistently integrated in the software development cycle yet. To emphasise the importance of this concept, further research has to be conducted with the aim to deliver sound and profound justifications why performance metrics are an important issue to consider. In a worst-case scenario, the end user would have to pay the price for a lacking consideration of Performance Testing in form of bad-performing or even non-working applications.

7. REFERENCES

[1] Applied performance management survey. Technical report, Compuware, 2006.

[2] Eurostat. statistical office of european union, 2013.

[3] S. Barber. Tester pi: Performance investigator, 2006.

[4] M. Brandwin. Mobile application performance engineering: A lifecycle approach to achieving confidence in application performance. 2011.

[5] M. Harman. The current state and future of search based software engineering. In *2007 Future of Software Engineering*, pages 342–357. IEEE Computer Society, 2007.

[6] M. Harman, P. McMinn, J. T. de Souza, and S. Yoo. *Search based software engineering: Techniques, taxonomy, tutorial*, pages 1–59. Empirical Software Engineering and Verification. Springer, 2012.

[7] IBM. 10 Million UK Consumers Using Mobile Commerce but 83% Have Experienced Problems. Technical report, Tealeaf, 2011.

[8] E. Lucas. Real devices and real networks: Ensuring mobile performance, 8/14 2012.

[9] Microsoft Corporation. Fundamentals of engineering for performance, 2013.

[10] S. Perez. Mobile app market: $25 billion by 2015, 1/18 2011.

[11] C. Ryan and P. Rossi. Software, Performance and Resource Utilisation Metrics for Context-Aware Mobile Applications. *11th IEEE International Software Metrics Symposium (METRICS'05)*, (Metrics):12–12, 2005.

[12] D. Schmidt. Model-driven engineering. *Computer-IEEE Computer Society*, 39(2):25–31, 2006.

[13] C. U. Smith and L. G. Williams. *Performance solutions: a practical guide to creating responsive, scalable software*. Addison Wesley Publishing Company Incorporated, 2001. 2001022849.

[14] C. U. Smith and L. G. Williams. Best practices for software performance engineering. In *CMG-CONFERENCE-*, volume 1, pages 83–92. Computer Measurement Group; 1997, 2003.

[15] C. Stary. Performance parameters and context of use. In *Performance Engineering, State of the Art and Current Trends*, pages 119–130, London, UK, UK, 2001. Springer-Verlag.

[16] C. Thompson, J. White, B. Dougherty, and D. C. Schmidt. Optimizing Mobile Application Performance with Model-Driven Engineering. pages 36–46, 2009.

[17] G. van der Heiden and P. Redshaw. Banking industry lessons learned in outsourcing testing services. Technical report, Gartner, 2012.

[18] E. Weyuker and F. Vokolos. Experience with performance testing of software systems: issues, an approach, and case study. *IEEE Transactions on Software Engineering*, 26(12):1147–1156, 2000.

[19] M. Woodside, G. Franks, and D. C. Petriu. The future of software performance engineering. In *2007 Future of Software Engineering*, FOSE '07, pages 171–187, Washington, DC, USA, 2007. IEEE Computer Society.