Chapter 9

An Analysis of the Effects of Bad Smell-Driven Refactorings in Mobile Applications on Battery Usage

Ana Victoria Rodríguez  
ISISTAN, Argentina

Cristian Mateos  
ISISTAN, Argentina

Alejandro Zunino  
ISISTAN, Argentina

Mathias Longo  
UNICEN, Argentina

ABSTRACT

Mobile devices are the most popular kind of computational device in the world. These devices have more limited resources than personal computers and battery consumption is always under user’s eye since mobile devices rely on their battery as energy supply. On the other hand, nowadays most applications are developed using object-oriented paradigm, which has some inherent features, like object creation, that consume important amounts of energy in the context of mobile development. These features are responsible for offering maintainability and flexibility, among other software quality-related advantages. Then, this chapter aims to present an analysis to evaluate the trade-off between object-oriented design purity and battery consumption. As a result, developers can design mobile applications taking into account these two issues, giving priority to object design quality and/or energy efficiency as needed.

INTRODUCTION

Smartphones and tablets represent a very attractive market to develop applications, mostly because they are the commonest form of computational devices in the world (Rodríguez et al., 2012). Despite their ever-growing capabilities, these devices have limited resources compared to personal computers. In particular, devices rely on batteries as their energy supply and, while their computational capacity has increased rapidly, battery life-time has increased slowly (Paradiso & Starner, 2005) (Flipsen et al., 2012). Then, even when today’s devices have more capabilities, their batteries have shorter duration because more powerful hardware needs more energy to run. In this context, it is important for mobile computing...
research to reduce energy consumption on these devices, especially when a resource intensive application is running. Moreover, there are different levels to do this, in principle operating system level and application level. This chapter chooses to work at the application level taking into account previous works which are focused on certain good object-oriented programming practices for desktop applications, but the environment and battery limitation claim application building should be rethought in the context of energy-efficient application development. Particularly, this chapter focuses on well-known anti-patterns known as Bad smells. There are several works whose main goal is to demonstrate how the refactoring of these Bad smells improves the object-oriented application design and maintenance (Lanza et al., 2006) (Fowler, 1999), but after doing so class designs are much more complex, which might negatively impact on energy consumption (Perez-Castillo & Piattini, 2014) and hence battery usage.

Even though battery consumption is an aspect that affects all mobile devices, they are the most popular computational device and, as a consequence, there is a variety of these devices and their operating systems to operate these devices. Then, this work is limited to a specific kind of device, i.e., running a particular operating system, and on the other hand we target a particular high-level programming language. First, in regard to the targeted mobile operating system, we focus on Android-powered devices because of the widespread nature of the Android platform, which is present in millions of smartphones, tablets, and other devices. Specifically, Android smartphones units were first in the list of best selling operating systems in the United States in the last years, having at the same time an exponential growth. Particularly, their worldwide market share was 48% during the third quarter of 2011, 69% during the second quarter of 2012 and 79% during the second quarter of 2013, more than twice as much as iOS, the second most popular operating system (IDC, 2014).

After selecting the operating system, the second step is to analyze the programming language options to make the experiments because the approach focuses on improving battery consumption at the application level. In this way, there are two options to analyze: Java and Android native code which are the commonest languages used for Android applications. Then, Java is the language selected to use in this chapter because of its popularity and its interesting features, i.e., it has been recognized as being a useful language to develop and execute both computationally intensive code (Mateos et al., 2010) (Mateos et al., 2011) (Taboada et al., 2011) and user-centered applications (PyPL, 2013). Also, Java provides portability, security and robustness (Mateos et al., 2012). In fact, Java provides and supports object-oriented programming, multi-threading and distributed computing implemented at the language level, platform independence, automatic memory management and exception handling, which make it interesting for general purpose applications.

As a result, the main goal of this chapter is advancing towards measuring the energy impact of refactoring Java source code for removing certain Bad smells. This means that the battery consumption of different applications will be evaluated. For each application several versions are produced, an original with Bad smell and refactored versions without these smells, with different level of refactorings depending on developer preferences. After this, the impact of the refactorings on both battery consumption and class design quality will be evaluated to provide developers a guide about refactorings and their impact on battery consumption and application design. In the same area a previous work (Rodríguez et al., 2012) demonstrates that object creation implies more battery consumption, which allows to predict the behavior of some of our refactorings. For example, God class refactorings improve application design at the cost of an increased application’s energy consumption because a trivial refactoring for this anti-pattern is to split a God class in two or more classes, which implies more object creation at runtime. This means that trade-offs are needed between energy efficiency on one hand and application design quality

www.igi-global.com/e-resources/library-recommendation/?id=107

Related Content

An IoT-Based Framework for Health Monitoring Systems: A Case Study Approach
www.igi-global.com/article/an-iot-based-framework-for-health-monitoring-systems/219360?camid=4v1a

Overview of Big Data-Intensive Storage and its Technologies for Cloud and Fog Computing
www.igi-global.com/article/overview-of-big-data-intensive-storage-and-its-technologies-for-cloud-and-fog-computing/219362?camid=4v1a

Examining of QoS in Cloud Computing Technologies and IoT Services
www.igi-global.com/chapter/examining-of-qos-in-cloud-computing-technologies-and-iot-services/191831?camid=4v1a

Privacy Preserving Text Analytics: Research Challenges and Strategies in Name Analysis
www.igi-global.com/chapter/privacy-preserving-text-analytics/119352?camid=4v1a