FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Planting Trees in the Android Forest: Energy Labeling for Mobile Applications

André Moutinho

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: João Paulo Fernandes

Second Supervisor: Bruno Cabral

October 31, 2022

Planting Trees in the Android Forest: Energy Labeling for Mobile Applications

André Moutinho

Mestrado Integrado em Engenharia Informática e Computação

October 31, 2022

Abstract

A utilização de dispositivos móveis, tais como smartphones e tablets, aumentou significativamente desde a sua invenção e ainda mais nos últimos anos, tornando-os uma parte da nossa vida quotidiana. Existem milhões de aplicações disponíveis para download espalhadas por diferentes app stores. Este enorme crescimento da utilização de dispositivos móveis surge um novo problema: a bateria de um dispositivo móvel tem uma capacidade limitada, não permitindo uma utilização ilimitada.

As aplicações móveis são continuamente melhoradas através da utilização de funcionalidades modernas, originando aplicações cada vez mais poderosas. Este facto leva a uma drenagem mais rápida da bateria de um dispositivo móvel. No entanto, a capacidade das baterias não acompanhou o ritmo de desenvolvimento das aplicações móveis, o que leva a uma preocupação crescente com a eficiência energética.

A eficiência energética é um dos aspetos mais importantes de uma aplicação móvel para os utilizadores de smartphones. Muitas vezes, os utilizadores desinstalam uma aplicação simplesmente devido à sua ineficiência no que toca a manter uma boa utilização do hardware. No entanto, há pouca ou nenhuma informação sobre a eficiência energética de uma aplicação específica. Os utilizadores são forçados a testar manualmente uma aplicação a fim de avaliar se esta satisfaz as suas necessidades pessoais.

Esta dissertação propõe a concepção e desenvolvimento de um modelo de classificação energético que permita às aplicações Android terem a sua eficiência energética rotulada, o que ajudará a análise do nível de consumo de energia de uma aplicação. Esta classificação permitirá aos utilizadores tomar uma decisão mais consciente quando optarem por descarregar uma aplicação, ao mesmo tempo que motivará os criadores a implementar aplicações com uma maior eficiência energética.

São automaticamente executadas ferramentas para analisar várias aplicações Android para identificar problemas relacionados com a energia. O número de problemas encontrados numa aplicação é então comparado com o número de problemas encontrados nas outras aplicações. Esta comparação resulta numa classificação energética, sendo que as aplicações com menos problemas têm a maior pontuação energética. O modelo de classificação energética proposto analisou um conjunto de dados de aplicações Android para ter a sua eficiência energética rotulada. No final, quatro aplicações que tiveram a sua eficiência energética rotulada por um trabalho semelhante foram também analisadas pelo modelo proposto nesta dissertação para servir de comparação.

The usage of mobile devices, such as smartphones and tablets, has significantly increased since their invention and even more over recent years, making them a part of our everyday life. There are millions of applications available for download spread throughout different app stores. This tremendous growth of mobile device usage emerges a new problem: a mobile device's battery has a limited capacity, not allowing unlimited usage.

Mobile applications are continuously improved by making use of modern features, originating increasingly more powerful applications. This fact leads to a faster drain of the battery of a mobile device. However, battery capacities have not kept up with the pace of development of mobile applications, which leads to a growing concern for energy efficiency.

Energy efficiency is one of the most important aspects of a mobile application for smartphone users. Often, users uninstall an application simply because of its inefficiency to keep a good usage of hardware. However, there is little to no information concerning the energy efficiency of a specific application. Users are forced to manually test an application in order to assess whether it meets their personal requirements.

This dissertation proposes to design and develop a framework that allows Android applications to have their energy efficiency labeled, which will assist the analysis of the level of energy consumption of an application. This labeling will allow users to make a more conscious decision when choosing to download an application while also motivating the developers to implement energy-efficient applications.

Tools are automatically executed to analyze various Android applications to identify energy-related problems. The number of problems found in an application is then compared to the number of problems found in the other applications. This comparison results in an energy classification, with the applications with the least problems having the highest energy score. The proposed energy classification model analyzed a dataset of Android applications to have their energy efficiency labeled. In the end, four applications that had their energy efficiency labeled by a similar work were also analyzed by the model proposed in this dissertation to serve as a comparison.

Keywords: Energy Saving, Android Applications, Energy Efficiency, Mobile Computing

Acknowledgements

I would like to thank my girlfriend, Mafalda, my sister, Ana, and my parents, Cândida and José, for supporting and helping me through this tough journey that culminates in my Masters degree. Without you this would have not been possible. I would also like to thank my supervisor, João Paulo Fernandes, and my co-supervisor, Bruno Cabral, for always having the necessary patience with me to overcome some rough patches.

André Moutinho

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Energy efficiency surges much interest and has many papers and books written about the topic. Furthermore, it plays a crucial role in the plans of most developed countries, whether it is for commercial and industrial competitiveness or for environmental reasons, such as reducing the emission of CO2 [17]. Achieving an improvement in energy efficiency includes reducing the amount of energy used and carbon emissions by designing things like buildings and technologies that are more environmentally friendly [20]. Data centers are an example of a building that needs careful attention regarding energy efficiency. Moreover, the current higher computational power imposes a more significant cooling challenge, which can be solved by regulating the ambient temperature [16]. Another significant example is energy efficiency related to mobile applications, which is the one tackled in this dissertation.

This dissertation proposes a way of labeling Android applications according to their energy efficiency. The approach followed in this dissertation involves automatically executing tools that analyze Android applications and identify energy-related problems (for now, bad energy patterns). The energy label is given by comparing the results given by the tools for each of the applications. This approach was used to ensure that the process is completely automatic, allowing for little to no intervention from the person analyzing the applications. The energy classification model as a whole was used to analyze 75 Android apps from the games category, taken from F-Droid[1]. In the end, 26.7% of the apps were classified with a 5.0 (in a classification from 1 to 5, with 5 being the highest), 5.3% with a classification of 4.5, 49.3% with a classification between 3.0 and 4.0, while 18.7% had an energy classification below 3.0.

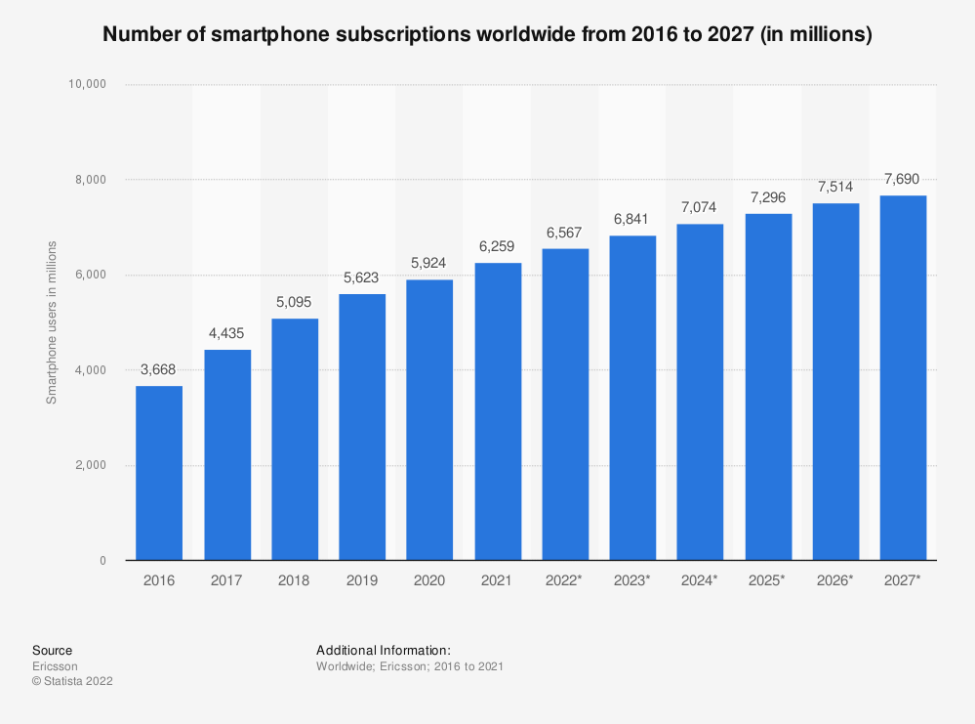

Figure 1.1: Number of smartphone subscriptions worldwide from 2016 to 2027

1.1 Motivation

Over the recent years, mobile devices have been increasing their role in society, being today crucial to the everyday life of a substantial amount of people. [2]Figure 1.1 illustrates the number of smartphone users worldwide, demonstrating that the usage of smartphones is a part of most people's everyday life. Furthermore, the usage time of these devices is of the topmost importance to the users. However, most of the recent applications use powerful functionalities that demand high computational power and high-speed data transmission, which severely impacts the energy consumption of the device's battery [18]. Additionally, there have not been any significant breakthroughs regarding battery capacity, which amplifies the concerns for energy efficiency in mobile devices. Many types of research have been conducted to optimize the energy consumption of mobile hardware, middleware, and applications, with the main focus on improving the user experience [22].

Millions of mobile applications exist across multiple app stores, allowing users to choose among applications with similar functionalities. App stores like the Google Play Store[3] or App Store[4] already help users choose between different applications by providing an overall rating

[1] https://f-droid.org/
[2] image taken from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
[3] https://play.google.com/store
[4] https://www.apple.com/pt/app-store/

based on users' feedback. Furthermore, operating systems like iOS or Android provide information about each application's proportion of battery used. However, there is no explicit information regarding the energy efficiency of a specific application before installing it. Past studies have shown that many applications contain user comments regarding their energy efficiency, which means that this particular information would be helpful to smartphone users [22].

In summary, this dissertation's motivation resides in the fact that smartphones are a part of most people's everyday life, so keeping smartphones running for extended periods is in the interest of its users. However, when downloading an application, it does not display the energy efficiency of that same app, which is an essential piece of information for users to have their smartphones available for extended periods.

1.2 Objectives

This dissertation aims to research and design a framework capable of gathering different inputs relative to the energy efficiency of Android applications and produce an energy label for each one, similar to what happens on home appliances. The main characteristics of the designed framework are:

- The apps are analyzed by tools that are capable of identifying different energy-related problems, allowing for a more accurate label;
- The labeling is generic, allowing the possibility to rate the energy efficiency of any Android application without prior knowledge regarding its functionalities.

In summary, the primary goal of this dissertation is to allow mobile device users to make a more informed and conscious decision through the availability of power-related information about each specific application before installing them, as well as help developers produce apps that are more energy-efficient.

1.3 Document Structure

This report is divided into six chapters. Chapter 1 introduces the problem of energy efficiency, particularly in mobile applications. In this chapter, the motivation and goals of this dissertation are also described. Chapter 2 introduces the concepts of energy efficiency and energy labeling, which are fundamental to this research project. Chapter 3 analyzes other solutions proposed to achieve an energy labeling for mobile applications and compares each example with the solution proposed in this dissertation. Chapter 4 describes the researched tools that fit the criteria necessary for integration in the energy classification model. Chapter 5 describes how the energy classification model was designed and its results. Chapter 6 contains the conclusions of the dissertation and the possible future work.

Chapter 2

Background

The concepts of energy efficiency and labeling are essential in the context of this dissertation. Hence they are presented as part of the reader's understandability of this dissertation.

2.1 Energy Efficiency

Energy efficiency is a generic term and does not have a universal quantitive measurement. The concept of energy efficiency is generally related to consuming less energy to produce the same expected output. Usually, the shifts in energy efficiency are measured through different indicators divided into four main groups: thermodynamic, physical-thermodynamic, economic-thermodynamic, and economic indicators [17]. Generally, customers consider the energy efficiency of a product like a refrigerator before buying it. Most of the time, customers even pay more for a more energy-efficient product to reduce the cost of the electricity bill [11]. In the context of energy efficiency for mobile applications, users also consider a particular application's energy consumption level. As mentioned earlier, the increase in the usage of mobile devices emerged the problem of having to charge the device regularly to ensure its operability, making the energy consumption of applications a significant concern.

2.2 Energy Labeling

Energy labeling is a concept that has been used in several areas, such as houses and home appliances. An energy label is a sticker that can be either mandatory or optional (depending on the field) and is attached to a product or its packaging, demonstrating the energy efficiency or consumption of the product. The primary purpose of an energy label is to allow consumers to choose between different products within a specific category, taking into consideration the energy consumption and making manufacturers produce energy-efficient appliances [12].

For an energy label to be reliable, it needs to be supported by suitable test procedures. A good test procedure must, by definition [13], comprise the following characteristics:

- reflect actual usage conditions;
- yield repeatable, accurate results;
- accurately reflect the relative performance of different design options for a given appliance;
- cover a wide range of models within that category of appliance;
- be inexpensive to perform;
- be easy to modify to accommodate new technologies or features;
- produce results that can be easily compared with results from other test procedures.

However, not all of the mentioned characteristics can be completely fulfilled due to their contradictory, which means that a good test procedure satisfies the maximum amount of characteristics, only excluding the ones that contradict the present characteristics [13]. This definition of a good test procedure can also be considered when applying energy labeling to mobile applications.

Test procedures vary between different types of appliances like refrigerators and freezers, clothes washers, boilers and furnaces, among others. One of the first appliances to have test procedures developed was refrigerating appliances due to their amount of energy consumed and ease to test. Procedures also vary between countries but generally involve placing the refrigerators in specific environments [13]. For example, European standards determine that the energy consumption of refrigerating appliances should be measured at 25º for low noise refrigerating appliances and at 16º and 32º for the rest [1].

[1] http://data.europa.eu/eli/reg_del/2019/2016/2021-05-01, accessed on February 18, 2022

Chapter 3

Related Work

As far as the literature review goes, some research projects tackle the problem of energy labeling for mobile applications. In the following sections, the related work that has been developed will be explained and critically reviewed, comparing them to the one proposed in this dissertation.

3.1 Energy-Aware Development and Labeling for Mobile Applications

The first example is from a doctorate thesis dated 2013 [21]. The labeling system proposed by the thesis was based on the design of benchmarks for specific usage domains, which were then bound and executed for particular mobile applications. Before the development of such benchmarks, a set of requirements were identified:

- Identification of use cases for usage domains;
- Definition of benchmarks for usage domains;
- Binding of the benchmark's test cases to individual applications;
- Profiling the energy consumption of mobile applications;
- Defining weights of individual use cases for the average energy consumption;
- Approximation of average energy consumption;
- Derivation of energy labels from applications' energy consumption;
- Semi-automation of the labeling process.

Moreover, the thesis proposed a system that comprises five steps: **service modeling**, where the use cases previously defined for each usage domain are transformed into a model that represents

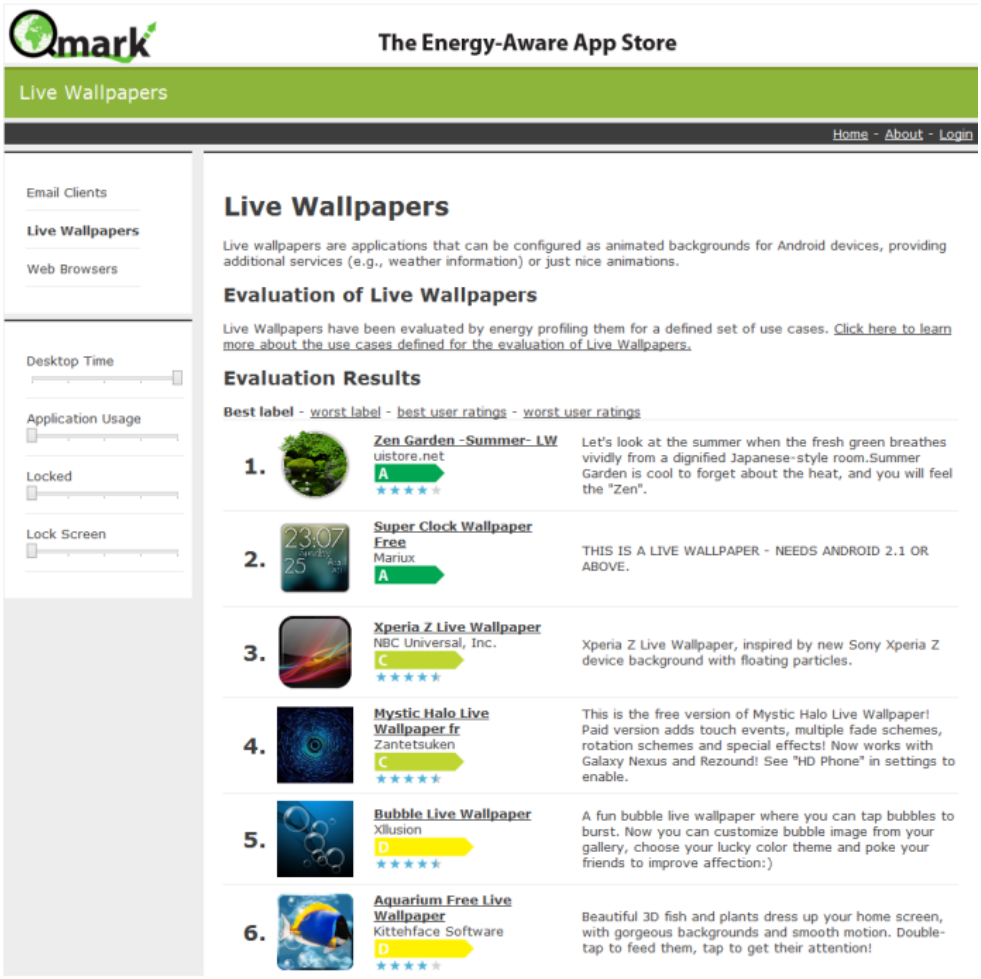

Figure 3.1: Interface of the proposed solution in the first example of the related work

the general usage of an application of the specific usage domain, as well as deriving abstract test cases from these models; **service model binding**, where the transitions of the model which represent user interactions are bound to sequences of application-specific UI interactions, transforming abstract test cases into ones ready for execution; **benchmarking**, where the concrete test cases are executed and profiled, majorly concretized by using the JouleUnit profiling framework; **usage modeling**, where the usage behavior of individual users is modeled to achieve a rating that better represents specific usage behaviors of users; **energy consumption estimation**, where the average energy consumption of mobile applications is measured.

Figure 3.1 demonstrates the application's interface, where the live wallpaper applications are listed in descending order of energy efficiency. The interface allows the users to adjust the weight that the different use cases have on their daily usage, allowing for a more precise energy labeling. However, the solution proposed by the doctorate thesis has the limitation of not being scalable because of its very specific test cases. The framework proposed by this dissertation intends to be generic, allowing the energy labeling of any application available in an app store.

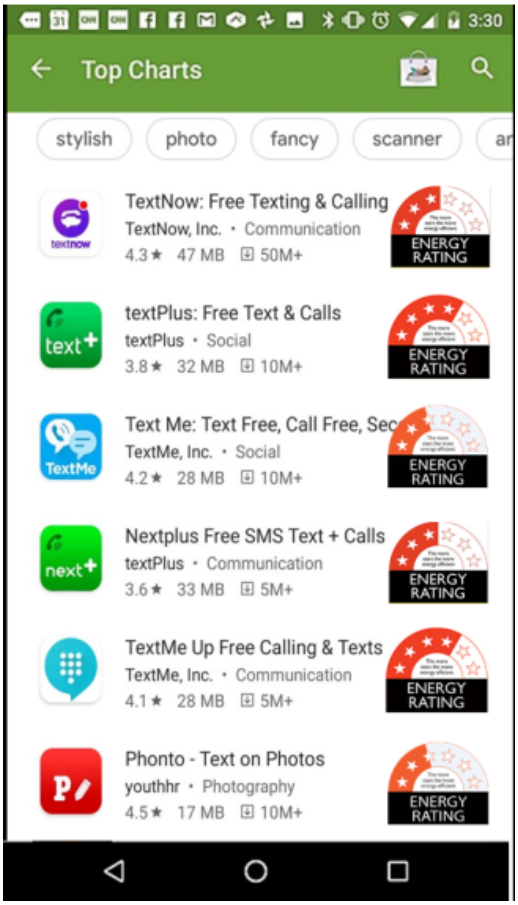

Figure 3.2: Interface of the proposed solution in the second example of the related work

3.2 Google Play Apps ERM: (Energy Rating Model) Multi-Criteria Evaluation Model to Generate Tentative Energy Ratings for Google Play Store Apps

The second example is also from a doctorate thesis dated 2020 [2]. In this case, two criteria were used to achieve a star rating system for mobile applications, particularly in the Google Play app store. The first criterion is related to the permissions of the applications. The different smartphone resources (like GPS or Wi-Fi) that applications can use were primarily measured in terms of energy consumption and given an energy rating expressed in a start system. Subsequently, each of the different application permissions was listed and related to the smartphone resources it needs in order to be functional. After these steps, all the existing application permissions were given a rating that reflected their energy efficiency based on their needed phone resources. Finally, the mobile applications were rated based on identifying the permissions needed for the application to be fully operational. The rating is based on a scale from one to six stars and divided between the most familiar network connections, which are Wi-Fi and Cellular.

The other criterion is related to the automatic refactoring of mobile applications. Two refactoring tools were used in this part of the system. One of which is called Energy-Aware Refactoring Approach for Mobile Apps (EARMO). This tool follows an anti-pattern correction approach that accounts for energy efficiency when analyzing mobile applications' anti-patterns [15]. The other tool is Leafactor, which can automatically improve the energy consumption of an Android application by refactoring the code to follow a set of patterns known to be energy efficient [7]. The energy rating is achieved by comparing the energy consumption of an application with its energy consumption after being automatically refactored by the two mentioned tools. Figure 3.2 illustrates the proposed interface for accessing the information on applications' energy efficiency found on Google Play.

The example described still imposes the limitation of analyzing the apps' energy consumption under test, which needs manual intervention. The solution proposed in this dissertation differs from the one described since it is an entirely automated way of labeling apps according to their energy efficiency.

3.3 An energy efficiency grading system for mobile applications based on usage patterns

The third example is an article published in The Journal of Supercomputing in 2018 [3]. In this article, the system proposed bases the energy efficiency grades on usage patterns. Two other scenarios to analyze the power consumption are presented and compared to the one used. The first one is where the power-related information of an application is derived from arbitrary scenarios. This scenario has the advantage of only needing to define usage behaviors of an application, which are based on its functionalities. However, the tests are generic and do not demonstrate different usage patterns of applications. The second scenario is where the power-related information is derived from customized scenarios. The advantage of this scenario is that the output is precise and reflects a particular usage pattern because it runs tests over a recorded usage scenario. On the other hand, the scenarios created are overfitted and cannot be adapted to other applications. The scenario proposed by the article is presented as a middle ground, where the output is not as precise as the one in the second scenario described but still reflects different usage patterns of the applications.

The energy labeling only compares applications within the same category since it would not be reasonable to compare an application that needs higher power-consuming resources to an application that does not. The app stores already have different categories assigned to applications, but those market-level categories were further divided into user-level categories. Furthermore, the applications were divided into different user-level categories, and each category's usage patterns were identified. In this article, only music players were analyzed. The proposed interface representing the energy labels of the applications is illustrated in Figure 3.3. The image depicts the example of a music player, which has the usage patterns of listening to music online, offline, or a combination of both.

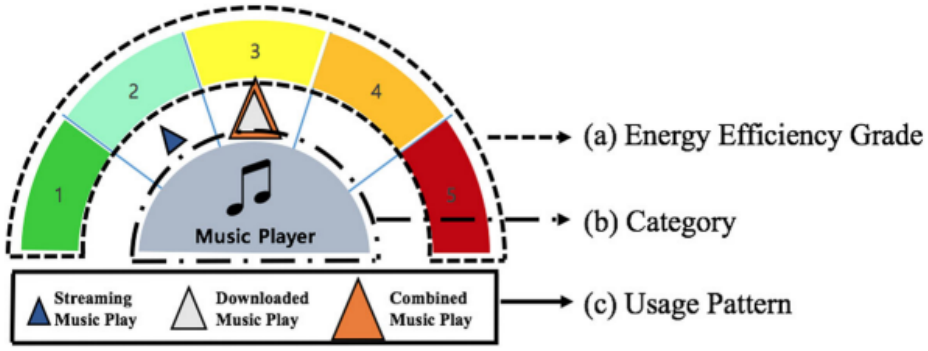

Figure 3.3: Interface of the proposed solution in the third example of the related work

Although the solution presented is attractive, it still imposes some problems. The usage pattern scenario is created empirically, and the application usage scenario cannot be automatically derived. Similar to the system proposed in section 3.1, it is not scalable to the point of analyzing all the different applications in an app store, which the framework proposed in this dissertation intends to do.

3.4 Certifying Energy Efficiency of Android Applications

The last example is an article from the EnviroInfo 2014 Conference [14]. This article proposed a system that compares applications according to their energy efficiency, akin to EU energy labels for electronic devices. Some challenges were identified when specifying the certification process, such as modeling user actions, certifying without computer science knowledge, measuring energy consumption, using devices minimally invasive, automating user actions, and preparing results. Furthermore, the certification process was divided into seven steps, which address the mentioned challenges:

- **Define category**: categories like notepad apps or email apps need to be defined to compare the energy efficiency of applications within the same category;
- **Define scenarios**:scenarios need to be specified to model the possible user actions. For example, scenarios for notepad apps would be "create note" or "delete note";
- **Assign app to category**: for applications to be comparable, they need to have a category assigned from the ones defined in the first step;
- **Implement each test case**: a particular test case needs to be implemented for each scenario of the different applications. The test cases can be manually implemented or generated automatically from recorded user actions;
- **Measure each test case**: the energy consumption must be measured in each of the test cases;

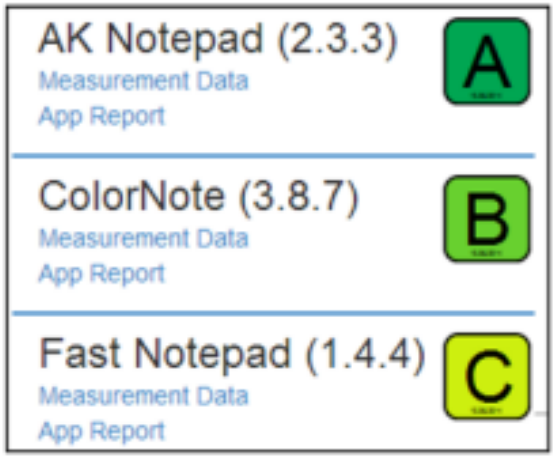

Figure 3.4: Interface of the proposed solution in the fourth example of the related work

- **Evaluate all measurements**: after running all test cases and measuring the energy consumption, the applications are given a rating according to their energy efficiency. This evaluation is done for each device because applications behave differently on different devices, and the screen size affects the power consumption. The grading is based on the energy consumption of the most energy-efficient app, which gets an A rating;
- **Visualize**: the system provides various types of information. *Labels* are represented as a single-colored letter, indicating the energy efficiency of the application, compared to others of the same category. *Certificates* present more detailed information, such as background information of the app and the device used for measuring. *Reports* demonstrate a comparison between all applications within a particular category. *App reports* show further details about the app's measurements, comparing the different energy consumptions in each scenario. *Measurement data visualization* demonstrates a detailed description of all data collected during the measurement phase.

Figure 3.4 illustrates the interface displayed to the user with the energy label for each application. The solution proposed by the article outputs much information relating to the energy efficiency of an application. However, the designed test cases tend to be overfitted, which causes the solution not to be easily scaled. The solution proposed by this dissertation is intended to be highly scalable, working with any application available in an app store and displaying the minimum amount of information for a user to make a more conscious decision when downloading an application.

Chapter 4

Tools

As mentioned before, this dissertation aims to produce an energy label for Android applications in an entirely automatic way that does not require knowledge of the functionalities of the individual applications. For that effect, the tools selected to analyze the applications must follow a set of rules so that the system as a whole is coherent. Each tool must follow the following rules:

- The analysis made is static since dynamic analyses require prior knowledge of the application's functionalities;
- It must analyze energy-related problems to be able to produce an energy label;
- It must be able to be run from the command line to achieve an automation of the process.

This section describes the different tools that were researched in order to achieve this goal. These tools are meant to be integrated in the energy classification model, being a crucial part of the energy labeling of mobile applications.

4.1 Android Lint

Android Studio provides a tool called lint which is capable of identifying and correcting problems with the code of an application without the need to execute it or write test cases, thus being independent of functionality. As seen in figure 4.1, this tool takes as input the application's source files and a "lint.xml" file, which is a configuration file that can be used to specify lint checks that the developer wants to exclude or to customize severity levels, and as output the potential issues related to *correctness*, *security*, *performance*, *usability*, *accessibility*, and *internationalization*. The lint checks can be run manually through Android Studio or the command line using the Gradle wrapper [1]. The latter is a rather important aspect since it allows for the automatic execution of the tool over several Android applications.

[1] https://developer.android.com/studio/write/lint, accessed on August 27, 2022

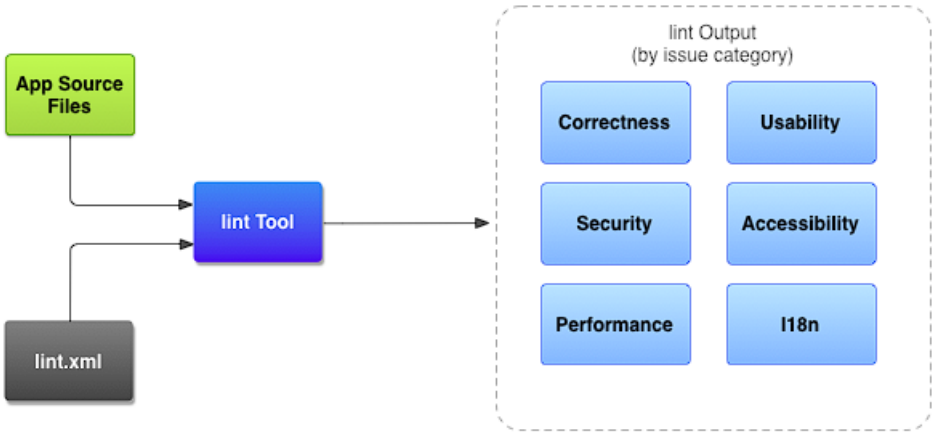

Figure 4.1: Workflow of the lint tool

However, Android lint does not identify issues specifically related to an application's energy efficiency. Le Goaër [8] presents an extension to the Android Studio tool, making it capable of finding green bugs, which are the defects present in the native code written by the developer that directly impact the battery life of a mobile device. This extension creates the category *greenness*, which can identify eleven different energy-related issues:

- **Everlasting Service**: Within the context of the Android framework, when a service is created, it must be manually stopped later on. This green bug identifies whether a service has been left to run indefinitely;
- **Dark UI**: the theme applied to an application directly impacts energy consumption since AMOLED and OLED screens benefit from the display of darker colors [10, 1]. In Android, Holo is set to Dark theme by default, and changing this value to Light theme results in this energy bug;
- **Battery-Efficient Location**: Location is a popular feature used among several mobile applications. It is a feature commonly known to have a higher impact on a device's battery life, thus making it crucial to be efficient. The developer should import the location package from Google Play Service rather than the package of the SDK;
- **Sensor Leak**: Most Android devices have built-in sensors, which are expensive in terms of battery usage, and should not be unnecessarily processing data while the app is in an idle state;
- **Sensor Coalesce**: Related to the events present in the hardware FIFO (queue), checks whether a particular variable has a positive value, which has an impact on the number of interrupts received by the Application Processor, also impacting the power consumption;
- **Bluetooth Low-Energy**: Many developers are unaware of this alternative to the commonly known Bluetooth, having a significantly lower power consumption;

- **Internet In The Loop**: Opening and closing internet connections continuously should be avoided since these are the network operations that consume the most battery [9];
- **Durable Wake Lock**: A wake lock is a mechanism that indicates whether the application needs the device to stay turned on. Generally, the wake lock is caught and then released; failing to release it impacts the duration of the battery drastically;
- **Uncompressed Data Transmission**: Compressing a file in order to transmit it over a network infrastructure is more efficient than transmitting it without the compression [4];
- **Rigid Alarm**: Using inexact alarms is preferable rather than using exact ones since they minimize the impact on the battery life;
- **Service at Boot-time**: Services can be programmed to start when a device is started. This is generally discouraged since services are operations that usually run for a long time, increasing the battery drain.

With this extension, **Android Lint** can output issues related to different categories: correctness, security, performance, usability, accessibility, internationalization, and greenness. In the context of this dissertation, only the greenness category matters.

4.2 EcoAndroid

EcoAndroid [19] is an Android Studio plugin that helps the development of a more energy-efficient Java-based mobile application by suggesting automated refactorings. This plugin also works on JetBrains' IntelliJ IDEA since Android Studio is an IDE built on it. EcoAndroid is capable of identifying five different energy patterns, which will be described later: *Dynamic Retry Delay*, *Push Over Poll*, *Reduce Size*, *Cache*, and *Avoid Extraneous Graphics and Animations*. At the time, no existing refactoring tools implemented checks for these energy patterns, which is why they were chosen. This tool provides informational and non-informational warnings. The first case is when a refactoring cannot be applied automatically or it affects too much code. In order to still help the developer, **EcoAndroid** leaves TODO comments so that the developer can apply these refactorings manually. The non-informational warnings are related to the refactorings that can be performed automatically, thus allowing for the tool to work without the need for manual intervention. Figure 4.2 represents the detection and refactoring process of the EcoAndroid tool, while figure 4.3 demonstrates all the energy patterns it can recognize in green, along with the informational (grey) and non-informational (white) cases for each respective pattern.

Below is a description of the five energy patterns that can be inspected by this tool:

- **Dynamic Retry Delay**: This pattern is related to the interval between different attempts to access a particular resource. When an attempt to access a resource does not succeed, the wait time before the next attempt should be increased since the resource is most likely down, thus decreasing the amount of energy wasted on useless attempts;

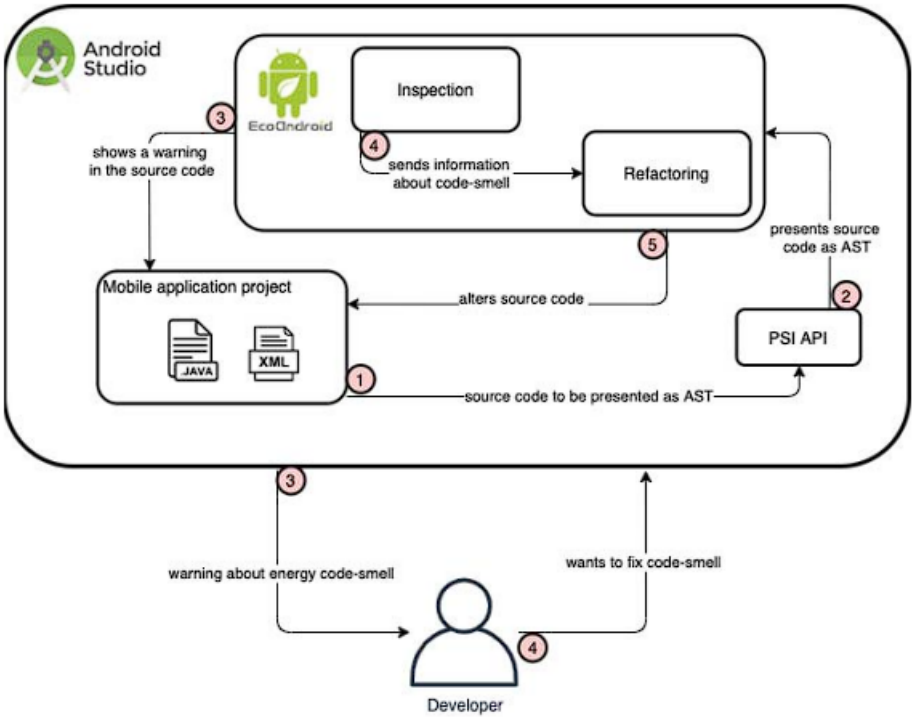

Figure 4.2: Detection and refactoring process of the EcoAndroid tool

- **Push Over Poll**: A push notification is when a client establishes a connection to a server and waits for messages. When the server changes its state, this data is sent to the client immediately. Polling is when the client periodically sends requests to the server in order to obtain information about its state, immediately getting a response with or without a message. This pattern suggests the use of push notifications instead of polling since the latter is less energy-efficient, especially in a system where there is not a significant number of notifications;
- **Reduce Size**: This pattern is based on the fact that a data transfer's size directly affects a mobile device's energy consumption. Due to this fact, the tool checks whether a transformation or compression can be made whenever a data transfer occurs;
- **Cache**: The goal of this pattern is to reduce the amount of code executed by storing data that is being used frequently, thus making applications more energy-efficient;
- **Avoid Extraneous Graphics and Animations**: Graphics and animations are known to have a high impact on energy consumption. The goal of this energy pattern is to avoid using these resources as much as possible, even though it is a challenging task to know when the use of these resources is strictly necessary.

The tool can also run in batch mode, storing the output in a file. This fact allows the tool to be eligible for the work proposed in this dissertation.

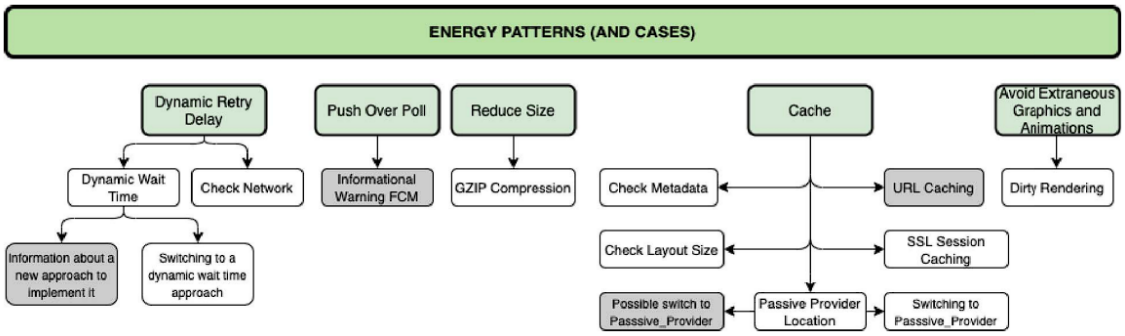

Figure 4.3: Energy patterns supported by EcoAndroid

4.3 Kadabra

Kadabra is a Java-to-Java compilation tool for code instrumentation and transformations using the LARA language. LARA [5] is a domain-specific, aspect-oriented programming language that maps applications to heterogeneous high-performance embedded systems. **Kadabra** is a tool that allows for static analysis to identify code patterns that have been proven to affect the energy consumption of an Android application. These patterns are referred to as "Energy Greedy Android Patterns", also known as **EGAPs**. This tool currently supports the analysis and identification of three EGAPs [6]:

- **HashMap Usage**: In the context of an Android application, the usage of HashMap is discouraged and should be replaced with its allegedly more energy-efficient replacement ArrayMap. This pattern simply recognizes the uses of HashMap, which ArrayMap should replace;
- **Excessive method Calls**: A method call usually involves manipulating the call stack, resulting in a performance drop when this happens an excessive amount of times. The goal of this pattern is to recognize the examples where a method call could be avoided, thus reducing the energy consumption of the Android application;
- **Member Ignoring Method**: This pattern helps recognize examples where a non-static method inside a class should instead be static. A method that does not access any class-specific information (i.e., fields and other non-static methods) or is not an overriding method should be defined as static. Static and non-static methods are stored in different memory blocks, meaning that even if many class objects are created, a static method is only stored once. Applying this pattern to an Android application helps reduce energy consumption.

Kadabra can take as input either an application's source code or apk, and runs from the command line, allowing for its automation.

Chapter 5

Classification Model

This chapter describes how the classification model was achieved in order to give an energy-efficiency label for Android applications. The code developed is available on Github[1]. The first step of the process is to gather the outputs given by using the tools mentioned in chapter 4 to analyze a set of Android applications. These outputs are essentially the number of energy-related problems found in each application. The classification consists of each tool analyzing and classifying each app by its standards, then aggregating these classifications to achieve a final one. Of the three tools mentioned, only Kadabra and Android Lint are in the classification model since EcoAndroid did not provide satisfactory results. Although EcoAndroid was successfully automated, the outputs did not provide with any information regarding energy problems, which is essential for this energy classification model.

In this chapter it is also shown results regarding 75 Android apps from the games category, taken from F-Droid[2]. The dataset should be more extensive, but some of the apps in the F-Droid repository had some issues. Only the ones that both Kadabra and Android Lint could analyze were included in the dataset used to demonstrate the results of the classification system as a whole. It is also important to note that the classification system divides the apps into categories since it makes sense to compare the apps only between themselves if they belong to the same category.

5.1 Classification

Figure 5.1 depicts a diagram of how the energy classification model generally works.

The first step of the process is to have the mentioned tools analyze all the apps. With a batch script, all the apps are automatically analyzed using the available tools, storing the outputs in a folder easily accessible. After that comes the classification process, which aims to transform the number of issues found in each application into a score that anyone quickly interprets. The classification is given by comparing all the analyzed apps between themselves. The apps with the least number of energy-related problems found have the highest energy score and vice-versa.

[1] https://github.com/AndreMoutas/Energy_Classification_Model
[2] https://f-droid.org/

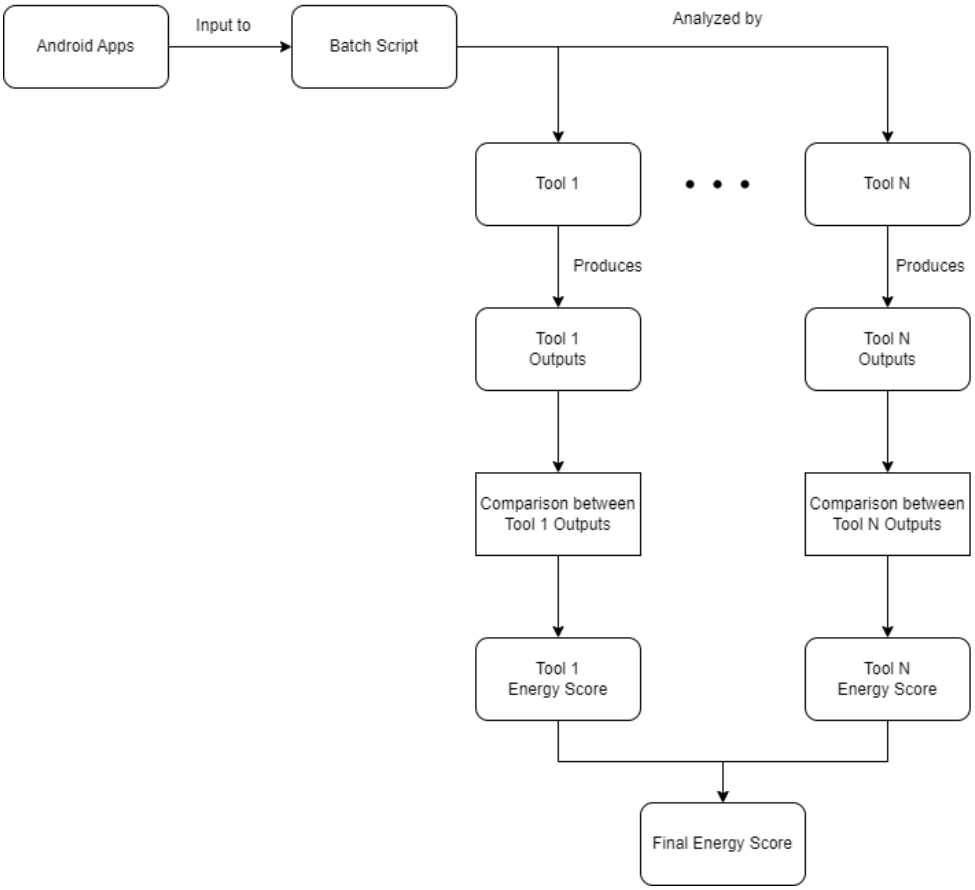

Figure 5.1: Diagram of the model

Since the tools analyze different aspects of the applications, each app has a score given by each of the tools, taking the mean value of these scores as the final classification of the app.

The comparison process considers the apps' weights instead of the raw number of problems found. The weights are the number of problems found in a particular app by a tool divided by the total number of problems found by that same tool within a certain dataset of apps. After the weight for each app is calculated, it is compared against the existing thresholds to produce a classification. The thresholds are 30%, 50%, 70%, and 90%. This means that the applications in the top 30% of apps that have the least number of energy problems found (i.e., the most energy-efficient) are given a score of 5, 30-50% are given a score of 4, 50-70% a score of 3, 70-90% a score of 2, and 90-100% a score of 1.

The value of 30% for the first threshold was chosen due to the high number of apps that did not have any energy problems found by the tools (33.2% for Kadabra and 29.3% for Android Lint). It is not reasonable to have a lower value for the threshold since all the apps with the same number of problems found have the same energy score. The following three thresholds (50%, 70%, and 90%) ensure that classifications 2, 3, and 4 each represent 20% of the apps, when analyzed by a certain tool. The 10% of apps with the highest number of problems found mostly represent the

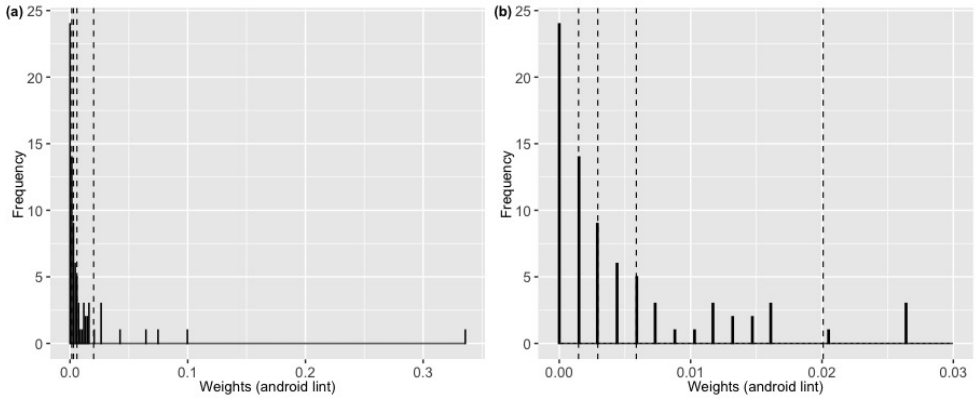

Figure 5.2: Histogram representing the distribution of the weights of all analyzed applications by Android Lint on a) and only the values between 0.0 and 0.03 on b)

outliers, which should have an energy score of 1.

The comparison only makes sense if the apps have characteristics in common since the standards can differ from category to category. For example, an application in the games category with 80 energy-related issues might have a different energy score than an application in the browser category with 80 energy-related problems. The apps only need to be compared within their category since those are the options between which the user will choose.

5.2 Results

In this dissertation, only applications within the games and theming (apps related to changing the theme of the phone) category were used for analysis and classification. Firstly, it is shown the results related to the apps in the games category.

Kadabra was used to analyze 250 Android apps, while Android Lint analyzed 82 apps, all taken from F-Droid[3]. Android Lint has a significantly smaller dataset since there were issues related to the tool analysis of some of the apps under test.

Figure 5.2-a) depicts the distribution of the weights of the 82 applications analyzed by Android Lint, marking where the thresholds appear in the case of this dataset for this tool. Figure 5.2-b) is a zoomed-in version, so the distribution and thresholds can be better seen since most values are between 0.0 and 0.03.

Similarly to figure 5.2, Figure 5.3-a) represent the distribution of weights of the 250 applications analyzed by Kadabra, along with the thresholds. Figure 5.3-b) is a zoomed-in version for the values between 0.0 and 0.015, which are the majority.

Among the 82 apps analyzed by Android Lint and the 250 apps analyzed by Kadabra, 75 apps were analyzed by both tools to achieve a final energy score. Table 5.1 shows the classification given by each tool for the 75 apps, alongside the final score, which is the mean value between the

[3] https://f-droid.org/

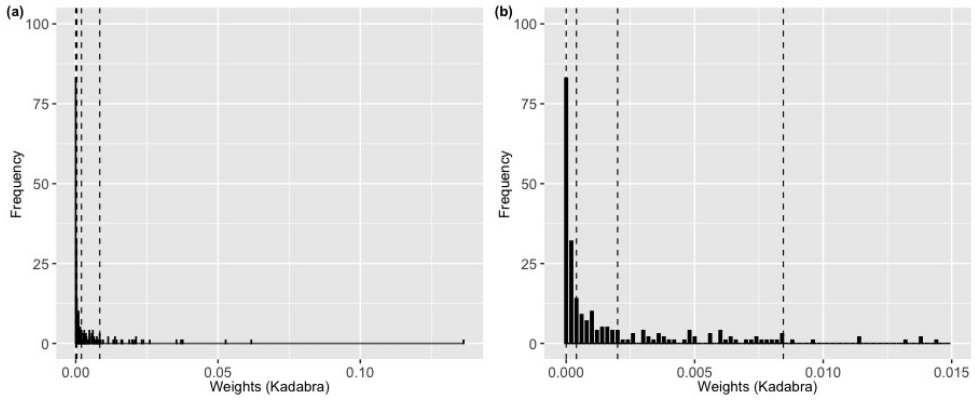

Figure 5.3: Histogram representing the distribution of the weights of all analyzed applications by Kadabra on a) and only the values between 0.0 and 0.015 on b)

two classifications. As can be seen from the table, there are a lot of mismatched values between the scores given by Android Lint and Kadabra, which is expected since these tools analyze different aspects of the applications, complementing each other.

Table 5.1: Table containing the energy classifications for 75 apps belonging to the games category analyzed by both tools

App	Android Lint	Kadabra	Final Score
2048-android	5	5	5
2050	5	5	5
AWhip	3	5	4
Adnihilation	2	5	3.5
Balance-the-Ball	5	5	5
Bummerlzaehler	1	1	1
Chalachithram	3	5	4
DSAAssistant	5	3	4
Fall	3	5	4
FlangAndroid	5	5	5
Kaesekaestchen	4	5	4.5
Kechi	2	5	3.5
Klock	5	5	5
Klooni1010	5	2	3.5
Lemuroid	5	5	5
LibreTrivia	5	3	4
LifeCounter	2	5	3.5
MHGenDatabase	5	3	4

Mines3D	4	2	3
OPMT	5	5	5
OSRSHelper	1	3	2
RSAndroidApp	1	2	1.5
RaumBaller	5	2	3.5
RoboZZle-Droid	1	3	2
ScoreSheets	4	5	4.5
Simple-Solitaire	1	1	1
StoryGame_Android	4	5	4.5
TabletopTools	3	5	4
TriviaYou	5	5	5
Trollslate	3	5	4
Unciv	3	5	4
Vector-Pinball	4	2	3
aaaaa	5	5	5
android	5	5	5
android-anuto	1	1	1
antimine-android	5	5	5
beat-feet	5	5	5
blockpuzzle	2	1	1.5
cdlc_player	4	2	3
crossword	2	5	3.5
dagger	5	2	3.5
dart-checker	5	5	5
dekadico	5	5	5
dodge-android	3	3	3
drgassistant	5	3	4
droidfish	5	1	3
gobandroid	2	2	2
gridle	2	3	2.5
hJOPandroidDriver	4	3	3.5
hayago	4	3	3.5
juvavum-android	2	5	3.5
knightsofalentejo	2	2	2
krassesSpiel	3	5	4
letters	5	5	5
makesomenoise	5	5	5
mastermindy-android	4	5	4.5

mtg-familiar	5	1	3
mtmods4android	3	3	3
nonocross	5	5	5
nounours-android	5	3	4
offline-puzzle-solver	5	3	4
passegares	2	2	2
pipepanic-android	5	5	5
pixelwheels	5	1	3
privacy-friendly-2048	1	5	3
privacy-friendly-solitaire	2	2	2
privacy-friendly-sudoku	2	2	2
randomix	2	3	2.5
rechnen-android	5	2	3.5
retrowars	5	5	5
rolldash	5	3	4
shattered-pixel-dungeon	5	1	3
tripeaks-gdx	5	5	5
vitosha-blackjack	3	3	3
wordlesolver	2	5	3.5

Furthermore, it is also interesting to compare how these 75 apps would classify under a more extensive set of applications. For that effect, the 75 apps were given an energy score when contained in a dataset of 82 apps for Android Lint and within a dataset of 250 apps for Kadabra.

Table 5.2 compares the classification of the 75 apps when within a dataset containing only those 75 apps and a dataset containing 7 more apps, totaling 82, only for the tool Android Lint. As seen from the table, none of the scores are different, which means that the new applications fall under the same pattern as the previous ones. Minimal changes are expected since only 7 apps are being added to the dataset.

Table 5.2: Table comparing the classifications of the original 75 apps when under a dataset of those 75 apps and a dataset containing 82 apps, for the Android Lint tool

App	75 apps dataset	82 apps dataset
2048-android	5	5
2050	5	5
AWhip	3	3
Adnihilation	2	2

Balance-the-Ball	5	5
Bummerlzaehler	1	1
Chalachithram	3	3
DSAAssistant	5	5
Fall	3	3
FlangAndroid	5	5
Kaesekaestchen	4	4
Kechi	2	2
Klock	5	5
Klooni1010	5	5
Lemuroid	5	5
LibreTrivia	5	5
LifeCounter	2	2
MHGenDatabase	5	5
Mines3D	4	4
OPMT	5	5
OSRSHelper	1	1
RSAndroidApp	1	1
RaumBaller	5	5
RoboZZle-Droid	1	1
ScoreSheets	4	4
Simple-Solitaire	1	1
StoryGame_Android	4	4
TabletopTools	3	3
TriviaYou	5	5
Trollslate	3	3
Unciv	3	3
Vector-Pinball	4	4
aaaaa	5	5
android	5	5
android-anuto	1	1
antimine-android	5	5
beat-feet	5	5
blockpuzzle	2	2
cdlc_player	4	4
crossword	2	2
dagger	5	5
dart-checker	5	5

dekadico	5	5
dodge-android	3	3
drgassistant	5	5
droidfish	5	5
gobandroid	2	2
gridle	2	2
hJOPandroidDriver	4	4
hayago	4	4
juvavum-android	2	2
knightsofalentejo	2	2
krassesSpiel	3	3
letters	5	5
makesomenoise	5	5
mastermindy-android	4	4
mtg-familiar	5	5
mtmods4android	3	3
nonocross	5	5
nounours-android	5	5
offline-puzzle-solver	5	5
passegares	2	2
pipepanic-android	5	5
pixelwheels	5	5
privacy-friendly-2048	1	1
privacy-friendly-solitaire	2	2
privacy-friendly-sudoku	2	2
randomix	2	2
rechnen-android	5	5
retrowars	5	5
rolldash	5	5
shattered-pixel-dungeon	5	5
tripeaks-gdx	5	5
vitosha-blackjack	3	3
wordlesolver	2	2

Table 5.3 compares the classification of the 75 apps when within a dataset containing only those 75 apps and a dataset containing 175 more apps, totaling 250, only for the tool Kadabra. As seen from the table, there is a considerable amount of changes in the scores, with 32% (24 out of

75) of the apps having a different energy score when under a dataset of 250 apps. This means the new 175 apps did not follow the same pattern as the previous ones, which is expected since the dataset grew to more than three times its original size, significantly impacting the distribution of the values. The significant changes in the energy scores demonstrate the need for a larger dataset to stabilize the distribution of values. Once there are enough applications, the energy-efficiency labels will be more accurate, and the analysis of new apps will have a minimal impact on the distribution.

Table 5.3: Table comparing the classifications of the original 75 apps when under a dataset of those 75 apps and a dataset containing 250 apps, for the Kadabra tool

App	75 apps dataset	250 apps dataset
2048-android	5	5
2050	5	4
AWhip	5	4
Adnihilation	5	5
Balance-the-Ball	5	5
Bummerlzaehler	1	1
Chalachithram	5	4
DSAAssistant	3	4
Fall	5	4
FlangAndroid	5	5
Kaesekaestchen	5	4
Kechi	5	5
Klock	5	4
Klooni1010	2	2
Lemuroid	5	5
LibreTrivia	3	3
LifeCounter	5	4
MHGenDatabase	3	3
Mines3D	2	2
OPMT	5	4
OSRSHelper	3	3
RSAndroidApp	2	2
RaumBaller	2	1
RoboZZle-Droid	3	3
ScoreSheets	5	5
Simple-Solitaire	1	1
StoryGame_Android	5	4

TabletopTools	5	5
TriviaYou	5	5
Trollslate	5	5
Unciv	5	4
Vector-Pinball	2	2
aaaaa	5	5
android	5	5
android-anuto	1	1
antimine-android	5	5
beat-feet	5	4
blockpuzzle	1	1
cdlc_player	2	2
crossword	5	4
dagger	2	2
dart-checker	5	5
dekadico	5	4
dodge-android	3	3
drgassistant	3	3
droidfish	1	1
gobandroid	2	2
gridle	3	4
hJOPandroidDriver	3	4
hayago	3	4
juvavum-android	5	4
knightsofalentejo	2	2
krassesSpiel	5	5
letters	5	5
makesomenoise	5	4
mastermindy-android	5	4
mtg-familiar	1	1
mtmods4android	3	3
nonocross	5	5
nounours-android	3	3
offline-puzzle-solver	3	4
passegares	2	2
pipepanic-android	5	5
pixelwheels	1	1
privacy-friendly-2048	5	4

privacy-friendly-solitaire	2	2
privacy-friendly-sudoku	2	2
randomix	3	4
rechnen-android	2	2
retrowars	5	5
rolldash	3	3
shattered-pixel-dungeon	1	1
tripeaks-gdx	5	5
vitosha-blackjack	3	3
wordlesolver	5	5

For the purpose of comparing the model proposed in this dissertation with another energy classification model, four apps that were given an energy score in [21] were also classified according to this dissertation's model. However, these four apps are live wallpaper apps, meaning they cannot be inserted into the previously mentioned dataset. Consequentially, a dataset of 41 apps (including the mentioned 4) belonging to the *theming* category was created and classified. Table 5.4 shows the energy scores given by each tool to the 41 apps, as well as the final energy classification.

Table 5.4: Table showing the scores given to the apps from the category theming

App	Android Lint	Kadabra	Final Score
10-bitClockWidget	5	5	5
Android-MonthCalendarWidget	4	5	4.5
Aquarium_Free_Live_Wallpaper	2	1	1.5
BeautyClockLiveWallpaper	2	5	3.5
Bubble_Live_Wallpaper	2	1	1.5
Color-Clock	2	5	3.5
DashClock_K-9	3	3	3
DashNotifier	1	3	2
EZ-Wifi-Notification	3	5	4
EmailPopup	4	3	3.5
FlashlightWidget	4	5	4.5
GLWallpaperService	4	2	3
HallMonitor	1	2	1.5
JellyBeanClock	5	5	5
MonthCalendarWidget2FOSS	5	3	4

MovingPolygons	4	3	3.5
MultiPictureLiveWallpaper	1	2	1.5
Mystic_Halo	3	5	4
Napply	4	4	4
Orbital-Live-Wallpaper	5	4	4.5
SidePanel	1	1	1
ToDo-List-Widget	2	2	2
Zen_Garden	2	1	1.5
abstract-art	5	2	3.5
android-clock-livewallpaper	4	5	4.5
android-yinyang	5	5	5
android_packages_apps_ParanoidWallpapers	3	5	4
android_packages_apps_XenonWallpapers	3	5	4
batterywidget	5	5	5
code	4	3	3.5
com.elsewhat.android.currentwallpaper	4	5	4.5
currentwidget	3	3	3
dashclock-sunrise	5	5	5
dashclockbattery	5	4	4.5
greyscale-cmte	5	5	5
jinbei3d	2	2	2
lukelauncher	5	2	3.5
null-black-wallpaper	5	5	5
scrollablecontacts	3	2	2.5
threedlite	5	3	4
wifiwidget	4	5	4.5

Table 5.5 compares the energy classifications given by this dissertation's energy classification model with the one developed by [21] concerning the previously mentioned four apps. The latter classifies apps considering different use cases of the apps, allowing for each use case to be given different weights. For this comparison, the energy labels result from the analysis of the apps, with each use case being given equal weight.

Table 5.5: Table showing the comparison of the energy score of 4 apps when being labeled by this dissertation's model and another energy model

App	My model (1 to 5)	Comparison model (G to A)

Aquarium_Free_Live_Wallpaper	1.5	B
Bubble_Live_Wallpaper	1.5	B
Mystic_Halo	4	B
Zen_Garden	1.5	A

As seen from the table, one app has a similar result, while the other three have a different result. This comparison shows that the the proof of concept presented is not yet sufficient. This discrepancy in results could be due to the small dataset, as well as due to the presence of only two tools that analyze Android apps. As seen in table 5.3, having a more extensive dataset could lead to a change in the energy scores of these apps. However, it is shown that having a more extensive dataset does not affect the scores by a significant amount. For that reason, the difference in the scores given by each model is too significant for the enlargement of the dataset to be the only solution. Adding new tools is the most crucial solution since it could have a more significant impact on the energy classifications while also making them more precise.

5.3 Discussion

This dissertation presented a model for labeling Android applications according to their energy efficiency. The characteristic that mainly differentiates the method proposed in this dissertation from the ones proposed in [21],[2],[3],[14] is that it allows apps to be labeled automatically without any manual intervention or prior knowledge about their functionalities. This is an important aspect of the classification system as it allows for the apps to be given as input to the system, and the energy score will be given as output without any manual interference. An automatic classification system is particularly advantageous when there are a significant amount of apps to be labeled.

The classifications are given by comparing the apps of a particular category. This means that when a new app is analyzed, it has to be compared to all the other apps that already have an energy label, which may change an app's energy score. This creates an instability in the score since an app with a score of 3 today might have a 4 tomorrow. However, the apps need to have their energy ratings up to date. For instance, if the new apps being developed increasingly contain more energy issues, the apps that already have an energy label should have their labels altered since they are now more energy-efficient than the newly analyzed apps.

Android Lint can identify 11 different energy-related problems present in Android apps, while Kadabra can identify 3. The proposed model uses the total number of energy issues for the classification process, meaning that each problem has equal importance. A possible way to improve the model is for each problem to have a different weight in the classification. This would require the problems to have a quantifiable impact on an application's energy consumption. To that end, it would probably be necessary to conduct an empirical study about each energy-related problem's impact.

Chapter 6

Conclusions

Energy labels have been gaining popularity over recent years. Concerns about the environmental impact of energy consumption have been increasing. Energy labels already exist in certain areas, like home appliances. However, the area of mobile applications does not still have associated energy labels.

This dissertation presented a proof of concept of an energy model capable of labeling Android applications according to their energy efficiency. This energy model consists of comparing the number of issues detected by different analysis tools over a set of Android apps in order to achieve an energy score. This resulted in a dynamic classification of applications, capable of changing the energy ratings of the apps as new ones are being analyzed and introduced. The dissertation also presents examples of other works that involved the energy labeling of mobile applications and how the work described in this document differs from theirs.

Unfortunately, not many tools satisfy the conditions required to be a part of the proposed model since the green software research field has still not yet developed much. For future work, invented and newly discovered tools that satisfy the conditions will be added to the energy model to achieve a more reliable classification.

6.1 Acknowledgements

This work was financed by FEDER (Fundo Europeu de Desenvolvimento Regional), from the European Union through CENTRO 2020 (Programa Operacional Regional do Centro), under project CENTRO-01-0247-FEDER-047256 – GreenStamp: Mobile Energy Efficiency Services.

References

[1] Tedis Agolli, Lori Pollock, and James Clause. Investigating decreasing energy usage in mobile apps via indistinguishable color changes. In *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pages 30–34. IEEE, 2017.

[2] Abdullah Mahmoud Almasri. Google play apps erm:(energy rating model) multi-criteria evaluation model to generate tentative energy ratings for google play store apps. 2021.

[3] Dusan Baek, Jae-Hyeon Park, and Jung-Won Lee. An energy efficiency grading system for mobile applications based on usage patterns. *The Journal of Supercomputing*, 74(12):6502–6515, 2018.

[4] Daniel Burgstahler, Ulrich Lampe, Nils Richerzhagen, and Ralf Steinmetz. Push vs. pull: an energy perspective (short paper). In *2013 IEEE 6th International Conference on Service-Oriented Computing and Applications*, pages 190–193. IEEE, 2013.

[5] João MP Cardoso, Tiago Carvalho, José GF Coutinho, Wayne Luk, Ricardo Nobre, Pedro Diniz, and Zlatko Petrov. Lara: an aspect-oriented programming language for embedded systems. In *Proceedings of the 11th annual international conference on Aspect-oriented Software Development*, pages 179–190, 2012.

[6] Marco Couto, João Saraiva, and João Paulo Fernandes. Energy refactorings for android in the large and in the wild. In *2020 IEEE 27th international conference on software analysis, evolution and reengineering (SANER)*, pages 217–228. IEEE, 2020.

[7] Luis Cruz, Rui Abreu, and Jean-Noël Rouvignac. Leafactor: Improving energy efficiency of android apps via automatic refactoring. In *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pages 205–206. IEEE, 2017.

[8] Olivier Le Goaër. Enforcing green code with android lint. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering Workshops*, pages 85–90, 2020.

[9] Ding Li, Shuai Hao, Jiaping Gui, and William GJ Halfond. An empirical study of the energy consumption of android applications. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 121–130. IEEE, 2014.

[10] Mario Linares-Vásquez, Carlos Bernal-Cárdenas, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, and Denys Poshyvanyk. Gemma: multi-objective optimization of energy consumption of guis in android apps. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 11–14. IEEE, 2017.

[11] Amory Lovins. Energy efficiency. *Energy Economics*, page 234, 2017.

[12] Teuku Meurah Indra Mahlia and Rahman Saidur. A review on test procedure, energy efficiency standards and energy labels for room air conditioners and refrigerator–freezers. *Renewable and Sustainable Energy Reviews*, 14(7):1888–1900, 2010.

[13] Alan K Meier and James E Hill. Energy test procedures for appliances. *Energy and buildings*, 26(1):23–33, 1997.

[14] Johannes Meier, Marie-Christin Ostendorp, Jan Jelschen, and Andreas Winter. Certifying energy efficiency of android applications. In *EnviroInfo*, pages 765–770, 2014.

[15] Rodrigo Morales, Rubén Saborido, Foutse Khomh, Francisco Chicano, and Giuliano Antoniol. Earmo: An energy-aware refactoring approach for mobile apps. *IEEE Transactions on Software Engineering*, 44(12):1176–1206, 2017.

[16] Michael K Patterson. The effect of data center temperature on energy efficiency. In *2008 11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, pages 1167–1174. IEEE, 2008.

[17] Murray G Patterson. What is energy efficiency?: Concepts, indicators and methodological issues. *Energy policy*, 24(5):377–390, 1996.

[18] Pijush Kanti Dutta Pramanik, Nilanjan Sinhababu, Bulbul Mukherjee, Sanjeevikumar Padmanaban, Aranyak Maity, Bijoy Kumar Upadhyaya, Jens Bo Holm-Nielsen, and Prasenjit Choudhury. Power consumption analysis, measurement, management, and issues: A state-of-the-art review of smartphone battery and energy usage. *IEEE Access*, 7:182113–182172, 2019.

[19] Ana Ribeiro, João F Ferreira, and Alexandra Mendes. Ecoandroid: An android studio plugin for developing energy-efficient java mobile applications. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, pages 62–69. IEEE, 2021.

[20] Elizabeth Shove. What is wrong with energy efficiency? *Building Research & Information*, 46(7):779–789, 2018.

[21] Claas Wilke. *Energy-aware development and labeling for mobile applications*. PhD thesis, Dresden, Technische Universität Dresden, Diss., 2014, 2014.

[22] Claas Wilke, Sebastian Richly, Sebastian Götz, Christian Piechnick, and Uwe Aßmann. Energy consumption and efficiency in mobile applications: A user feedback study. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 134–141. IEEE, 2013.