Mobile Energy Efficiency Services

E2.1 . Report on research results and conceptualisation of the analysis system, acquiring and processing data related to energy efficiency

Authors: R&D Teams from UNIVERSIDADE BEIRA INTERIOR, UNIVERSIDADE DE COIMBRA, and Caixa Mágica Software

Version: 1.0 (September 2022)

PORTUGAL 2020 — UNIÃO EUROPEIA — Fundos Europeus Estruturais e de Investimento

INDEX

1 Introduction

This section contains a brief introduction to the Ranking Mobile Applications by Energy Efficiency task inside the **Greenstamp** project, with its motivation and objectives. At the end of the section, one may find the description and objectives of the task being studied in this document. This general introduction aims to give the context of the project to new team members who participate in only one task.

The rest of the document contains the necessary sections for the research and development of the task being addressed.

1.1 Motivation

With the abundance of smartphones worldwide using all kinds of mobile apps and using a sizable amount of energy, we need to start thinking about Energy Efficiency in this context [16, 27]. Mobile applications are becoming more complex and demanding more computational power than ever. Although we see development in software and hardware, batteries are not evolving at the same pace. If apps are not energy efficient, they contribute to increased energy consumption, thereby adversely impacting the global environment. These apps will also consume users' smartphone batteries faster than usual.

Previous studies have reported that many pages in app stores have comments referencing apps' energy efficiency. However, no mobile application store provides information about the energy efficiency of an app.

Researchers have pursued various approaches to address energy efficiency in mobile app development. One such method involves the creation of guidelines and code refactors aimed at enhancing battery consumption.

1.2 Objectives

This project aims to create an independent and scalable method that calculates fair labels about the Energy Efficiency of an Android application. This work is in the context of the GreenStamp project, and it's one of the main points of focus of the project.

The main goal is to answer the question:

> "How to correctly and fairly label Android applications in an independent and scalable way?"

This means labelling all Android applications by their Energy Efficiency and ranking them accordingly. The idea is to follow the EU Energy labels used for many household appliances such as microwaves, freezers, fridges and washing machines. When the app is analyzed and labelled, this information can be used in an Android application distributor, such as Aptoide.

1.3 Main contributions

To achieve the strategic objective and impacts mentioned, this project aims to research and develop highly innovative techniques and technologies, unparalleled in the market. They are translated into the following technical-scientific objectives:

- Integration of an APK decompiler (Jadx) and several analysis tools (EARMO, Kadabra, Android Lint, ADoctor, Paprika, Relda2);
- Classification thresholds calculation per analysis tool and Label thresholds calculation;
- Development of a Python program that runs everything together and automatically labels the analysed application;
- Threshold calculation and testing done with more than 700 apps.

2 Description of Tasks

T1.1: Integration of an APK decompiler (Jadx) and several analysis tools (EARMO, Kadabra, Android Lint, ADoctor, Paprika, Relda2)

Most of the analysis tools and the APK decompiler (Jadx) were developed in Java, so the integration was similar between all these tools. Only ADoctor and Paprika needed some customisation to support the needs of the project. All the tools were tested using Java 20, and all results were produced using that version.

Relda2 was developed in Python2, and although the main code developed in this analysis project of GreenStamp was made using Python3, a port from Relda2's code to Python3 was prone to errors, so we decided to keep Relda2 in Python2 and run it with that version.

T1.2: Classification thresholds calculation per analysis tool and Label thresholds calculation

To correctly calculate thresholds to help us decide which label to attribute to each application, we calculated separate thresholds for each category and each analysis tool. This is because apps in different categories and with other purposes will need less or more energy. For example, a game will be more computationally expensive than a note-taking app. Thus, we need to classify them differently.

We split the results file into separate datasets by category to achieve this. Then, for each of these, calculated the thresholds of 5 different classification levels for each tool depending on the number of detections. These levels are also characterized by a number of stars, being five stars the best level and one star the worst one.

Then, we applied the same method to calculate the final label thresholds using the sum of the number of stars of each app (not using categories in this stage to get a global ranking).

To calculate the different thresholds, we followed a method based on making a density function based on weights [1].

T1.3: Development of a Python program that runs everything together and automatically labels the analyzed application

A Python program was made to automatically decompile an Android application, analyse it using various analysis tools and calculate its label using thresholds while taking into account the app's categories. This program was made with future changes in mind, so if it is needed to add more and different analysis tools, it would not be hard.

All the code can be found on Github:

https://github.com/JayRx/Labelling-Android-Apps-by-Energy-Efficiency

The following list shows and explains the mandatory and optional program arguments:

- ApkPath -- The path to the APK file of the app to be analyzed. (Mandatory)

- -categories / -c -- A list with the names of the categories of the app. (Mandatory)
- -analyzers / -a -- A list with the names of the analyzers to run. Default: Earmo Kadabra AndroidManifestAnalyzer Lint ADoctor Paprika Relda2. (Optional)
- -force / -f -- A flag that, if present, will force the execution of all or some analyzers discarding previously saved values. (Optional)
- -fdroid -- The package name of an app from F-Droid. The program will try to download the app from F-Droid if a package name is given. (Optional)
- -aptoide -- The package name of an app from Aptoide. The program will try downloading the app from Aptoide if a package name is given. (Optional)

After the analysis and label calculation, the program saves the results and the app's info in a JSON file named "report.json".

3 References

[1] Tiago L. Alves, Christiaan Ypma, and Joost Visser. Deriving metric thresholds from benchmark data. In 2010 IEEE International Conference on Software Maintenance, pages 1–10, 2010.