Certifying Energy Efficiency of Android Applications

Johannes Meier[1], Marie-Christin Ostendorp[1], Jan Jelschen[1], Andreas Winter[1]

Abstract

While smartphone and tablet functionality is increasing, battery runtime goes down. Both users and software developers are not always aware of the energy consumed by applications. Without this information, users cannot choose energy-efficient apps to prevent battery drain. Awareness of energy consumption would also entice app developers to implement more energy-efficient apps to remain competitive. This paper presents a certification process and corresponding Android tool support for comparing apps regarding energy efficiency. It is demonstrated using notepad apps.

1. Motivation

Application possibilities of mobile devices (e.g. smartphones, tablets) keep increasing, yet average battery runtimes stagnate at about one day [7]. Reasons for low battery runtimes are not only slow progress in battery research, but also increasing energy consumption of apps. Even though power-wasting apps are known and discussed by users, e.g. at Google's Play Store [8], there is little information about the energy efficiency of apps [1]. This concerns both users and developers: On one hand, users need selection aid for energy-efficient apps. On other hand, developers need data about their apps' energy consumption to guide energy efficiency improvements, and for comparison with competitors. This paper's objective is to create certificates (example in Fig. 1), allowing to compare applications wrt. their energy efficiency, akin to EU energy labels for electrical devices [2]. Thereby, awareness for energy-efficient apps, and in the long run, energy efficiency itself rises.

A certification process has to be supported by a toolchain, allowing certifiers to create such certificates. This paper's core idea is to measure energy consumption for typical user actions on apps. There are challenges to overcome when specifying such a certification process and its tool support:

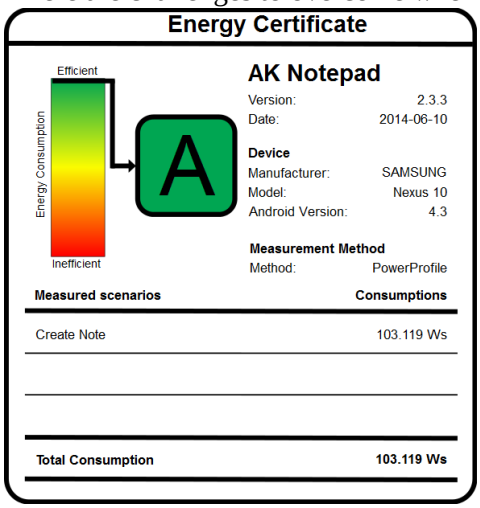

Figure 1: Example Certificate

1) Modeling user actions: Typical user actions have to be taken into account, to find comparable apps and get comparable values while measuring.

2) Certifying without computer science knowledge: Certifiers do not necessarily have programming skills, and must be enabled to easily perform certifications without it, using appropriate tooling (cf. Sec 4).

3) Measuring energy consumption: Application's energy consumption has to be measured during user actions on actual mobile devices.

4) Using devices minimally invasive: Devices for measurements shall be used minimally invasive, to prevent damage or voiding warranties, and allow reuse for further projects. Particularly, this prohibits rooting devices.

5) Automating user actions: For comparability of energy measurements of the same user actions for different apps, human inaccuracy while operating devices (e.g. different delays) has to be reduced. To this end, "test cases" are used to automate user actions. This provides repeatability, and

[1] Carl von Ossietzky University, Oldenburg, Germany, {meier,ostendorp,jelschen,winter}@se.uni-oldenburg.de

comparability of different test case runs, and ensures uninfluenced measurements. To create such test cases without computer science knowledge (Challenge 2), tool support is required.

6) Preparing results: Measurements have to be evaluated and visualized (cf. Fig. 1) differently, e.g. for users (energy efficiency as buying criteria) and developers (details for app optimization).

After discussing related work in Section 2, this paper presents a certification process coping with these challenges in Section 3. Its tool support, generating energy efficiency certificates for Android apps, is demonstrated in Section 4. For certification and toolchain validation, Section 5 applies the certification process and tooling to notepad apps. The paper concludes with Section 6.

2. Related Work

A similar approach for application certification was presented by Wilke [8], which can be seen as the first relevant work in this field. Apps with similar functionality are compared by measuring energy consumption while running typical user actions. Measurements are used to compare and certify apps, which are visualized with energy labels A through F. The challenges introduced in Section 1 will be used in the following to delimit Wilke's approach from this paper's approach.

For *modeling user actions* (1), both approaches group apps with similar functionality into categories: Apps in the same category support similar user actions called "scenarios". Scenarios of apps represent their main functionalities – e.g. "create note" is a scenario for notepad apps. To define user actions, Wilke uses finite state machines, which model app states like the current window as states, and user actions as state transitions between app states, enabling dynamic consumption calculation for different user actions. In contrast, this paper defines typical user actions as *text-based* scenarios, so certifiers need no state machine modelling skills to use the tool support (Challenge 2).

Moreover, the tool support presented in this paper is more comprehensive, because certifiers are supported in administration and controlling the whole certification process via a GUI to allow *certifying without computer science knowledge* (2). Furthermore, certifiers and users have access to different visualizations presenting results of a certification in detail via a GUI (Challenge 6).

Wilke provides several methods for *measuring energy consumption* (3) and focuses on hardware-based measurements. This paper uses only software-based measurement methods, to allow measuring energy consumption without external hardware and without breaking devices to access non-removable batteries (Challenge 4). However, Wilke's approach yields higher precision results.

While Wilke offers no tool support for the creation of test cases for *automating user actions* (5) on a device, due to his different focus, this paper introduces a *recorder* to log user actions, and a *generator* to generate test cases out of recorded data. This technique is self-implemented, because existing tools for non-rooted devices have restrictions or runtime errors. Approaches for rooted devices, e.g. RERAN [3], conflict with non-invasiveness (Challenge 4).

3. Certification Process

The steps of the certification process (as shown in Figure 2) are described in detail using an example coming from the domain of notepad applications. These steps address the challenges of Section 1, and are set in italics with their number in brackets.

At the beginning of the certification process the apps to certify will be categorized after their main functionality (steps 1 and 3). The scenarios define user actions (step 2). For automating these scenarios for each app and scenario a test case has to be developed (step 4) which is executed during the measurement of energy consumption on devices (step 5). After measuring all scenarios for each app of the category, the measuring results are evaluated (step 6). The results of measurements and evaluations are visualized in different forms for users and app developers (step 7).

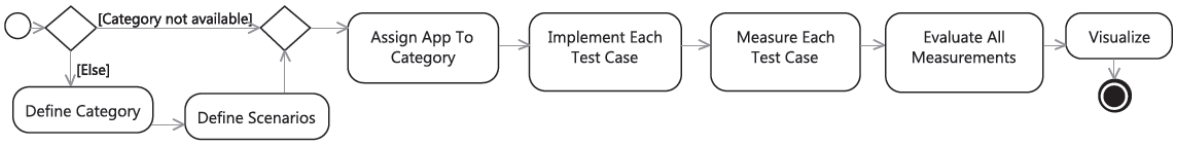

Figure 2: Certification Process

1) Define Category: For comparability of similar apps, the application to be certified (called *app*) has to be assigned to a specific *category* as first step. A category is a classification of apps providing the same set of similar user actions. In the following, typical user actions are called *scenarios* for *modeling user actions* (1). E.g., if a specific app for managing notes like "AK Notepad" should be certified, it can be assigned to a category called "notepad apps". Another possible category is, for instance, "email apps", containing apps for managing emails.

2) Define Scenarios: For each typical user action of the category, a scenario has to be specified for *modeling user actions* (1). To get comparable certifications for different apps of one category, scenarios have to be precise. Two possible scenarios of "notepad apps" would be "create note" and "delete note". "Create note" is further detailed: "create new note, add text "Hello", and save note".

3) Assign App to Category: To find proper categories, the functionality of apps to certify has to be compared with scenarios of existing categories. If there is no match, a new category with scenarios has to be created (see above). E.g., "AK Notepad" is an app to manage notes and allows creating new notes. The latter functionality fits the scenario "create note" of the category "notepad apps". Thus, "AK Notepad" belongs to "notepad apps", not "email apps", as it does not show emails.

4) Implement Each Test Case: For each scenario of the category, and each app in it, a test case has to be implemented. A *test case* is an app for *automating user actions* (5) of one scenario for exactly one app. Such test cases are necessary to replay user actions on different apps, and to reduce human inaccuracy during measurements. This ensures the comparability of apps, and reproducible results. E.g., for „AK Notepad", a test case for the scenario „create note" has to be implemented, which presses the icon to create a new note, adds text into the new note, and saves the new note by going back to the main window. For test case implementation, certifiers can record user actions and automatically generate test cases, or they can implement test cases by hand.

5) Measure Each Test Case: Each test case is executed on a device for *measuring energy consumption* (3) while the test case is running.

6) Evaluate All Measurements: After measuring all test cases, the results are evaluated for each category (using all its scenarios), and each device, which is necessary for *preparing results* (6), because this step calculates the energy efficiency of apps to certify. The evaluation is done for each device, because the same app can behave differently on different devices, and various hardware, for instance different screen sizes cause different energy consumption. Measurement data of apps relating to the same scenario is used for energy efficiency rating: The best app (lowest consumption) gets an A rating, defining bounds for the other energy classes, which are (for example) 10% higher than the boundary before. Wilke [8] proposed different boundaries for different categories.

7) Visualize: After the evaluation, results are visualized in five forms to fulfill the challenge *preparing results* (6) for different stakeholders: *Labels* present evaluation results as single colored letter, the simplest visualization of the energy efficiency classes for users and app developers. *Certificates* (cf. Fig. 1) are more detailed, containing data such as background information about the app, the device used for measuring, and the energy consumption of different scenarios for users and app developers. *Reports* show all certified apps of one category for comparison, for users who want to buy the most energy-efficient app. *App reports* represent one app, and show further details about the measurements for app developers, to compare different energy consumptions per scenario or

device component. *Measurement data visualizations* are the most detailed descriptions of all data collected during the measurement of one app. This is interesting for app developers, to view all measured data, particularly with regard to the point in time of the energy consumption. For each device, all visualizations are available. Labels are also available for mean values over all devices.

4. Tool Support

The toolchain supporting the certification process (Section 3) is shown in Figure 3. The tool architecture follows a three-layer-architecture with data management on the *server*, web views allowing the actors to interact (*PC*), and energy measurements on *Android devices*.

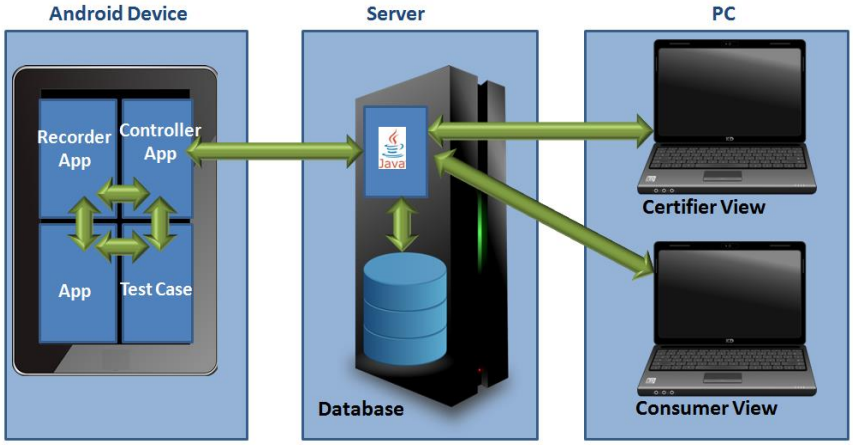

Figure 3: Architecture Overview

The steps *Define Category*, *Define Scenario*, and *Assign App to Category* of the certification process (Section 3) are technically supported by a web view called *certifier view*. The certifier view allows the certifier to manage the toolchain. Separate from the certifier view, the *consumer view* allows users and app developers to see certified apps with their visualizations (step *Visualize*). Therefore, all certification results are placed at one central place. Certifier and consumer view are communicating with the *server*, which manages data persistence with an external *database*.

Android devices used by the toolchain use four relevant apps: The *ControllerApp* manages *server* communication and measurements, *test cases* execute scenarios on *apps* to certify, and the *RecorderApp* simplifies creating *test cases*.

The **ControllerApp** runs on each Android device used for measuring. The ControllerApp takes the communication tasks, manages the installation of possibly missing apps and test cases, and execution of apps and test cases, and runs measurements. For that, the ControllerApp secures the installation of required apps, starts measurements, and runs test cases. After the test case finished, the measurement will be stopped, the collected measurement data will be sent to the server, and saved in the database (step *Measure Each Test Case*).

Apps are Android applications whose energy consumption should be certified, and have to be installed on the measuring device. In the example, "AK Notepad" is an *app*.

From the technical perspective, **Test Cases** are Android apps which control the apps to certify, automatically executing user actions of scenarios at measuring time (steps *Implement Each Test Case* and *Measure Each Test Case*). Test case implementations use the Robotium framework [6], which is a test framework for Android to control GUIs of other apps. The toolchain supports uploading test cases via the certifier view, or recording test cases using the RecorderApp.

The **RecorderApp** allows creating test cases via executing the scenarios on the app to certify, instead of writing Robotium code. The certifier executes the user actions of scenarios on the app. The RecorderApp listens for those user events, and sends the events to the server. The server generates Java code using the Robotium framework from the recorded data to replay the user actions, and

creates automatically test cases through compiling the Java code as Android app (step *Implement Each Test Case*). This allows creating test cases without programming skills (Challenge 2).

As measurement method for Android, the *Power Profile* method is implemented in the ControllerApp, reusing the work of Josefiok, Schröder, Winter [4]. The Power Profile method is based on power profiles available on devices. In a power profile, the average energy consumption per time unit of specific hardware components such as screen, CPU, Bluetooth etc. is listed. Measuring the time each component has been active during test case execution allows calculating the total energy consumption. Side effects at measuring time will be reduced by the certifier through stopping other apps, and running the test cases multiple times (step *Measure Each Test Case*).

On the server, the measured data will be evaluated to determine the energy efficiency of apps using the evaluation idea of step *Evaluate All Measurements*. In the toolchain, there are five visualizations, automatically generated using ELVIZ [5], a framework for model-driven visualization. All visualizations are provided through the *consumer view* (step *Visualize*).

5. Application

This section shows how to use the presented certification process and the supporting toolchain. As the result of the following practical example with three notepad apps, Figure 4a shows the view for consumers with certified notepad apps.

To certify notepad apps "AK Notepad", "ColorNote", and "Fast Notepad", the certifier has to create a category "Notepad Apps" for apps whose main functionality is managing notes (step *Define Category*). To concretize this functionality, scenarios have to be created representing typical user actions with notepad apps. Examples for notepad scenarios are "create note", "delete note", "read note", and "search for a note" (step *Define Scenarios* (2)). Here, the "create note" scenario is considered. After that, the three apps have to be uploaded and assigned to the notepad category (step *Assign App to Category* (3)). All these steps are managed via the certifier view (see Figure 3).

The consumer gets results and visualizations as shown in Fig. 4. The certified "notepad" apps will be shown as list with their names and labels, showing the final certification result (A, B, C) in the consumer view (Fig. 4a). The visualizations of the measurement data ("Measurement Data") and the app report ("App Report") are accessible via the links. The detailed certificates are accessible by clicking on the app's label graphic. The category report (accessible via "Category Report") includes as an example Fig. 4b, which shows different energy consumptions of the three apps.

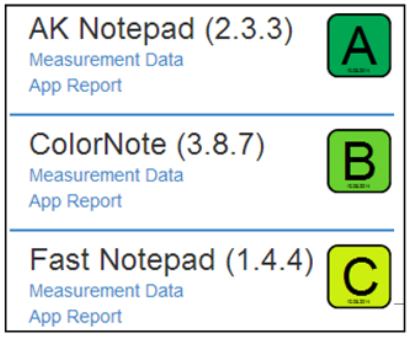

Figure 4a: Consumer view

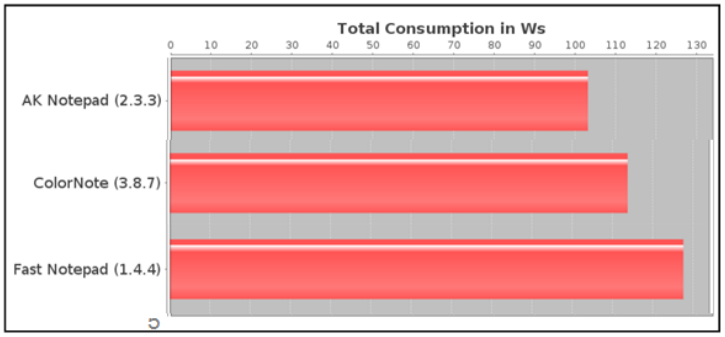

Figure 4b: Consumptions of apps in category report

With the list of certified apps, the user can choose an energy-efficient app and can use the energy efficiency as criterion at his buying decision. An app developer can use this overview for a first fast evaluation of his apps and the apps of competitors. More detailed information is accessible via the list of different scenarios (in certificate and app report) of the category showing differences having

regard to scenarios. Further information for the app developer consists of the measurement data and app report of the app. Here, the Power Profile measurement method allows the analysis of energy consumption per device component showing, as example, different CPU usage.

Realized experiments with the tool support show that the duration of a test case has a significant effect on the energy consumption, because the measured energy is as tendency proportional to the time in which a test case is running. As consequence of that, app functions which need many user actions (for example menus with much nesting levels) need more energy because the app runs longer. That is the reason why "Fast Notepad" consumes more energy than "AK Notepad".

6. Conclusion

This paper presented a way for certifying the energy efficiency of applications. A certification process which allows comparability of energy consumption of different applications has been demonstrated. Furthermore, tool support for this certification process for Android apps has been provided allowing certifiers to generate certificates – including administration, measurement of energy consumption, evaluation, and visualization – automatically, without having specific knowledge in computer science. Thereby, a selection aid for users is available with visualizations like labels, certifications, and reports. But not only users benefit from this new information: App developers now have detailed information about the energy consumption of their applications and are thereby able to improve the energy efficiency of their implementations. As future trend, the awareness for energy efficiency can increase and app developers will be able to reduce the energy consumption of their apps. Thus, in the long term apps can be more energy-efficient.

As outlook, the tool support can be used to evaluate the energy efficiency of apps for other research questions. Further experiments with much more certified apps will help to validate and fine-tune the certification process. Furthermore, the working certification process can be transferred to other domains like iOS apps and desktop software.

Acknowledgment

This work has been realized in a one year project group by six master's students. We thank Alex Bauer, Dennis Kregel, Kai Tammen and Alexandru Zay for their collaboration.

References

[1] Bunse, C., Naumann, S., Winter, A., "Entwicklung und Klassifikation energiebewusster und energieeffizienter Software", Springer, In: Wohlgemuth, Gomez, J. M., Lang, C. (eds): IT-gestütztes Ressourcen- und Energiemanagement: 5. BUIS-Tagen 2013, Oldenburg.

[2] Deutsche Energieagentur, "Das EU-Energielabel", Deutsche Energieagentur, http://www.stromeffizienz.de/private-verbraucher/eu-energielabel.html (21.04.2013).

[3] Gomez, L., Neamtiu, I., Azim, T., Millstein, T., "RERAN: Timing- and Touch-Sensitive Record and Replay for Android". International Conference on Software Engineering (ICSE 2013).

[4] Josefiok, M., Schröder, M., Winter, A., "An Energy Abstraction Layer for Mobile Computing Devices", In: Bunse, Christian; Gottschalk, Marion; Naumann, Stefan; Winter, Andreas (eds): 2nd Workshop EASED@BUIS 2013 Oldenburg.

[5] Ostendorp, M.-C., Jelschen, J., Winter, A, "ELVIZ: A Query-Based Approach to Model Visualization", GI-Edition LNI, Bonner Köllen Verlag, In: Modellierung 2014, Vienna, Austria.

[6] "Robotium. The world's leading Android test automation framework". https://code.google.com/p/robotium/ (02.06.2014).

[7] Spiegel, "Angefasst: Der Blackberry Z10 im Test", 2013, http://www.spiegel.de/netzwelt/gadgets/angefasst-der-blackberry-z10-im-test-a-880411.html (15.06.2014).

[8] Wilke, C., "Energy-Aware Development and Labelling for Mobile Applications", PhD Thesis, University of Dresden, 2014.