Model-based Energy Consumption Prediction for Mobile Applications

Felix Willnecker[1], Andreas Brunnert[1], Helmut Krcmar[2]

Abstract

Investigating the energy consumption of mobile applications (apps) is becoming a growing software engineering challenge due to the limited battery lifetime of mobile devices. Energy consumption is defined as the power demand integrated over time. Profiling the power demand of an app is a time consuming activity and the results are only valid for the target hardware used during the measurements. The energy consumption is influenced by the resource demands of an app, the hardware on which the app is running, and its workload. This work adapts resource profiles for enterprise applications to predict the energy consumption of mobile apps without the need to own a physical device. Resource profiles are models that represent all aspects influencing the energy consumption of an app. They can be used to predict the energy consumption for different hardware devices and evaluate the overall efficiency of an app. Moreover, the workload can be changed so that the impact of different usage patterns can be investigated. These capabilities lay the foundation for a platform-independent way of quantifying the energy consumption of mobile apps.

1. Introduction

The capacity of batteries dictates the uptime of smartphones and tablets. The battery usage limits the daily device availability and decreases the long term battery quality. Reducing the energy consumption of these devices decreases the number of loading cycles, improves user satisfaction, reduces total operational cost, and eases the carbon footprint of mobile technology [1, 2]. Recent developments in battery technology are mostly defined by larger or brighter displays, faster central processing units (CPU), or an increasing number of sensors [3]. Optimizations on the operating system (OS) level cannot compensate for that increasing power demand [4]. An underestimated optimization potential can be found in applications (apps) [5-7]. Capra, Formenti et al. [8] showed that, under the same workload, different Enterprise Applications (EA) with similar functionality have significantly different levels of energy consumption [8]. Comparable effects can be discovered in apps on mobile devices [9].

Reducing the power demand of apps can be achieved by exchanging algorithms or resources and limiting the usage of sensors or network traffic [1, 3]. The most common methodology to enable developers to detect energy bugs is energy profiling [10]. Profiling technologies differ per device and per platform, in formats and accessibility [3]. Profiling results can diverge per target device, even if it is running on similar platforms [6]. A common requirement of these profiling methods is that they require the hardware: the device under test (DUT) [2]. Existing research focuses on profiling and lacks models and simulations. In this work we propose an abstraction of the hardware characteristics in terms of resource profiles [11]. These profiles as input for a simulation engine can predict energy consumption and battery life for apps without the need for a DUT.

The proposed approach allows developers to compare multiple devices, which are running the same app by exchanging only parts of a resource profile. This leads to a more automated evaluation and

[1] fortiss GmbH An-Institut Technische Universität München, Guerickestr. 25, 80805 München, Germany [willnecker, brunnert]@fortiss.org

[2] Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany, krcmar@in.tum.de

less labor intensive analysis compared to using real DUTs. The effects of using an app for hours can be simulated and those simulations are typically faster than real time. Furthermore, this approach allows the comparison of multiple versions of the app according to its energy consumption in different usage scenarios (e.g. power user, medium interaction). As this approach is completely software-based it can be distributed and scaled for remote evaluations.

2. Predicting Energy Consumption

2.1. Resource Profiles for Mobile Applications

Predicting energy consumption requires three different inputs: the workload, resource-demanding aspects of an app, and the hardware environment. Each of those inputs can be depicted independently from each other. The hardware and the workload define the environment in which the app is running. The energy consumption is predicted by a simulation engine as shown in Figure 1 - Simulation Process.

The hardware environment represents the power demands of different power consumers of a device. These power demands may vary depending on their state and usage. For example, the power demand of a CPU is dependent on whether it is running on a full clock signal or in power saving mode. It is furthermore dependent on the utilization of a hardware resource. The hardware environment captures these dependencies. It contains CPU, Wi-Fi, cellular transmitters, the display and sensors like the Global Positioning System (GPS) sensor. The power demand necessary to run the OS and keep the device alive is also represented in the hardware environment. Battery life prediction relies on the capacity of the used battery; the battery capacity is also specified in the hardware environment.

The prediction of the energy consumption of an app requires the specification of its resource-demanding aspects. These specifications define which hardware resources (e.g. CPU, sensors, and display) are used by an app and how much demand is placed on them for different control flows. The power demands of these components can then be calculated using the simulated utilization of the hardware. The battery life prediction is then calculated from the energy consumption and the battery capacity.

Different usage scenarios can result in different energy consumption calculations. The third and last component of the resource profiles describes the user's behavior. This part is called the workload. The workload describes how a user utilizes the publicly available interfaces of the app. In a typical app, this public interface is the user interface; therefore the workload describes how the user interacts with the app.

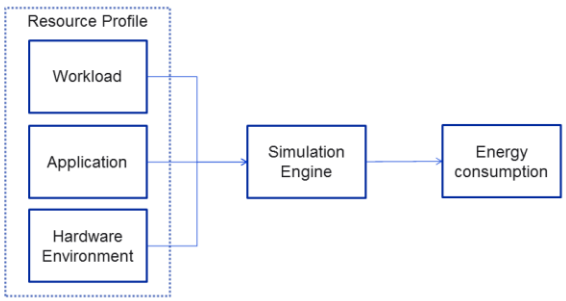

Figure 1 - Simulation Process

2.2. Use Cases

Resource profiles allow developers to predict energy consumption without the actual device. Each hardware environment contains the necessary information to simulate the energy consumption of a

device. The battery lifetime of these devices can vary due to the fact that some components are more efficient in one device, some devices have larger displays that consume more power, and the capacity of the batteries itself varies among different devices. In short, the power demand of a component depends on the platform it is running on [6]. The same app can therefore have different effects on the battery life when deployed on multiple devices. Resource profiles can be used to predict those effects by simulating the same workload and app inputs with different hardware environments. The approach can also be used to compare multiple versions of an app, to detect changes in the power behavior of the app and its components.

Prediction results simplify comparing of the energy consumption of apps. Apps with similar functionality can be compared according to their energy consumption by exchanging the app input and keeping the workload and hardware environment stable. The simulation results show the differences according to the energy efficiency of those apps. These differences allow developers to rate which of the simulated apps uses the least battery capacity. Comparing two apps with similar functionality can lead to a recommendation for using one app over another.

Investigating different workloads nowadays requires different profiling experiments. Comparing the results requires developers to re-run exactly those experiments with different hardware or different versions of an app. Using resource profiles for this purpose allows developers to change the workload input before executing a simulation. Comparing simulation results with different workloads leads to different energy consumption predictions.

2.3. Realization

To realize the resource profile concept outlined in section 2.1 and the corresponding use cases explained in section 2.2, we are using the Palladio Component Model (PCM) meta-model to represent resource profiles [11]. PCM allows the modeling of resource-demanding aspects of an app, workloads, and hardware environments independently of each other. PCM consists of several model layers [12]. One of the main models within the PCM meta-model is the repository model. The repository model contains the components of apps, their operation behavior, and resource demands as well as their relationships. These repository model components are combined into a system model. The available devices and their capacities (i.e. CPU cores) are specified in a resource environment model representing the hardware environment. An allocation model specifies how the system model elements are mapped to these devices. The workload on the system is represented by the usage model. The default simulation engine of PCM is called SimuCom [13]. It uses the five PCM model layers to predict performance metrics such as response time or CPU utilization of a software system. The PCM meta-model is represented using the Eclipse Modeling Framework[3] (EMF).

PCM models cannot be used to predict energy consumption without further extensions. The work of Brunnert et al. [11], therefore, contributed power consumption models to PCM to predict the energy consumption of EAs. These power consumption models specify a linear formula to calculate the power consumption of a hardware device in the resource environment model. The formula takes the utilization of different hardware resources into account. At the end of a simulation run, their extension automatically integrates the resulting power demand values over time to calculate the energy consumption of a device.

Mobile apps and devices require further extensions to the PCM meta-model, as the structure of their hardware resources and power consumers is different compared to EAs. Energy consumption predictions for EAs do not need to consider displays, Wi-Fi transmitters, or other sensors (e.g.

[3] http://www.eclipse.org/modeling/emf/

GPS, Accelerometer, etc.). This work extends the existing power consumption models for PCM to take smartphone and tablet energy characteristics into account. For this purpose, models that represent hardware environments (i.e. resource environment) and resource-demanding aspects of an app (i.e. repository model) need to be extended to represent GPS and similar sensors and an app's use of these sensors. Additional extensions are required to specify the battery capacity of a mobile device to analyze how long this capacity will last for a specific usage scenario.

PCM simulations aim to predict the load for different components with various workloads. The power consumption models calculate the power demand of a hardware device based on the load on the resources utilized by an app. Hardware devices are represented as so called resource container in the resource environment model. As part of these containers the power demand of different hardware components is specified. Furthermore, the power consumption model of such a container includes the power demand necessary to run the system in an idle state. In contrast to EA systems, mobile OS power consumption depends on the environment of the device. An example of these variances is the power consumption of the cellular connection, as it varies based on the quality of the connection to the next cellular base station. Therefore, the power consumption models are extended by adding an approximation of the base power consumption of the mobile device. This approximation includes the varying power consumption of the OS and the device in an idle state.

As in the work of Brunnert et al. [11], the power demand of resources like CPUs is calculated based on their utilization. The power demand based on the utilization is described by a linear function. Several authors such as Fan et al. [14] and Rivoire et al. [15], showed that such linear models are sufficiently accurate for representing these kind of resources [14, 15]. Increasing the CPU utilization will thus increase the power demand proportionally. The linear function calculates the current power demand based on the utilization of the CPU calculated by PCM.

The standard PCM resource environment model contains linking resources. These resources represent network components like a Wi-Fi or a cellular data transmitter. For mobile devices, the data transmission, especially over cellular, can be a significant power consumer. Therefore, this work extends the PCM meta-model with power consumption models for linking resources in order to simulate the power consumed by transferring data to and from the device. This extension of the linking resources provides developers with the ability to model several transmission units for the hardware environment with different levels of power demand. This allows the simulation of varying power demand of Wi-Fi compared to cellular data.

The display is considered a resource similar to a semaphore, as usually only one or no app can use the display at the same time. To represent this behavior, we use the active resource PCM meta-model element type that is also used to represent resources such as CPUs or HDDs. However, we modify the scheduling behavior for this component in a way that prevents other processes from accessing this component while in use. This ensures that the display can only be used by one consumer at a time. To simulate different brightness or color intensities that affect the power demand of a display, we use a distribution function that assumes a certain display behavior.

Sensors like GPS can be accessed by multiple processes at the same time. The constraint applied for displays is not valid here. The power consumption model therefore only takes into account whether such resources are used or not. The power demand of such a resource is therefore considered for the calculation of the device's power demand when at least one process accesses the sensor. The power demand of such sensors also varies depending on the required sensor accuracy. The required sensor accuracy is therefore also taken into account as a variable in the power consumption model.

2.4. Creating Resource Environment Models

The creation of the resource environment model is based on a calibration app. This app reads and parses manufacturer hardware profiles as a baseline, if present. The profile is then adjusted with measurements of the current battery drain. This approach can only measure the complete power consumption at a specific time [16]. In order to determine the power consumption of a specific component, we measure the electric current before, during, and after the activation of this component and calculate the difference between using and not using the component. This process is repeated several times to approximate a sufficiently accurate mean value or function. This mean value, multiplied with the battery voltage, results in the power demand of a component in our model. After the complete calibration we calculate a function over the base current drain on the battery and consider this as the OS power consumption for our device.

3. Related Work

Resource profiles are already being used in the area of EAs [11]. These profiles are based on architecture-level performance models and provide a common description of the performance-relevant aspects of an EA architecture including its resource-demands. The separation of the app descriptions from the hardware environment and the workload provides the ability for dynamic assemblies and individual evaluations. Such evaluations are performed by using resource profiles as input for a simulation engine. Therefore, the same resource profile can be used to evaluate the energy consumption and performance of an app on multiple hardware environments without deployment or the physical presence of the specified hardware. This work adapts resource profiles for mobile apps in order to create a hardware and platform independent energy evaluation and prediction engine.

Hönig et al. [6] suggest a model-based approach to provide developers information about the power demand of their code. This approach is based on hardware models provided by Android[4] manufactures. We reuse these models as calibration baseline for our resource environment models and refine them by calibrating the power demand of the hardware components of the device.

Josefiok et al. [3] outline that power measurements on Android devices differ between manufactures and OS versions. The multitude of application programming interfaces and variances in granulartiy complicate comparisons between different measurements. The work suggests a common energy abstraction layer as a profiling baseline technology [3]. We add a representation of the results in terms of a hardware environment model to this approach.

4. Conclusion and Future Work

This work proposed to adapt EA resource profiles for predicting the energy consumption of mobile apps. Simulations of the energy consumption of mobile apps are realized by extending PCM with power consumption models. We focus on a platform independent representation of the hardware characteristics and the simulation of the utilization of sensors, CPU, display and data transmitters. Using these extended models as input for a simulation engine allows predicting battery life of a device running a mobile app. The model extensions help developers to understand the varying power demands of different devices, to predict diverging battery life, and to optimize mobile apps in order to save battery power and reduce profiling effort.

The resource-demanding aspects of an app (i.e. repository model) are nowadays created manually. Brunnert et al. [17] described a way to create such models automatically using dynamic analysis for Java Enterprise Edition (EE) applications. An adapted approach for mobile apps could be used to

[4] http://www.android.com/

create repository models automatically. This could decrease the effort required to simulate mobile apps energy consumption.

PCM usage models represent the behavior of a user while working with a mobile app. Observing user behaviors automatically and detecting common usage patterns could create better usage models for the simulation. Tools like Google Analytics[5] already allow detailed user tracking for web and native apps for desktop and mobile. The most common interaction paths can be extracted from such tools as well as the probability distribution across all interaction paths. Analyzing this data for capturing more accurate workloads would reduce the effort for the model creation and improve the simulation results.

References

[1] C. Bunse and S. Stiemer, "On the Energy Consumption of Design Patterns," in *2nd Workshop EASED@ BUIS 2013*, Oldenburg, Germany, 2013, pp. 4-5.

[2] C. Wilke, S. Richly, S. Götz, and U. Aßmann, "Energy Profiling as a Service," *GI-Jahrestagung*, vol. 220, pp. 1043-1052, 2013.

[3] M. Josefiok, V. Offis, A. Winter, and C. V. O. Universit, "An Energy Abstraction Layer for Mobile Computing Devices," in *2nd Workshop EASED@ BUIS 2013*, Oldenburg, Germany, 2013.

[4] M. Gottschalk, M. Josefiok, J. Jelschen, and A. Winter, "Removing Energy Code Smells with Reengineering Services," in *GI-Jahrestagung*, Braunschweig, Germany, 2012, pp. 441-455.

[5] C. Bunse, S. Naumann, and A. Winter, "Entwicklung und Klassifikation energiebewusster und energieeffizienter Software," *IT-gestütztes Ressourcen-und Energiemanagement* pp. 557-566, 2013.

[6] T. Hönig, C. Eibel, K. Rüdiger, and W. Schröder, "SEEP: Exploiting symbolic execution for energy-aware programming," in *Proceedings of the 4th Workshop on Power-Aware Computing and Systems*, Cascais, Portugal, 2011, p. 4.

[7] A. Brunnert, C. Vögele, A. Danciu, M. Pfaff, M. Mayer, and H. Krcmar, "Performance Management Work," *Business & Information Systems Engineering*, vol. 6, pp. 1-3, 2014.

[8] E. Capra, G. Formenti, C. Francalanci, and S. Gallazzi, "The impact of MIS software on IT energy consumption," in *18th European Conference on Information Systems*, Pretoria, South Africa, 2010.

[9] A. Pathak, C. H. Y., and M. Zhang, "Bootstrapping Energy Debugging on Smartphones: A First Look at Energy Bugs in Mobile Devices," in *10th ACM Workshop on Hot Topics in Networks*, Cambridge, MA, USA, 2011.

[10] H. Höpfner and C. Bunse, "Towards an energy-consumption based complexity classification for resource substitution strategies," *Grundlagen von Datenbanken*, 2010.

[11] A. Brunnert, K. Wischer, and H. Krcmar, "Using Architecture-Level Performance Models as Resource Profiles for Enterprise Applications," in *Proceedings of the 10th ACM SIGSOFT International Conference on the Quality of Software Architectures (QoSA)*, Lille, France, 2014.

[12] R. Reussner, S. Becker, E. Burger, J. Happe, M. Hauck, A. Koziolek, *et al.*, "The Palladio Component Model," *Journal of Systems and Software*, vol. 82, pp. 3 - 22, 2009.

[13] S. Becker, "Coupled model transformations for QoS enabled component-based software design," Dissertation, Fakultät II - Informatik, Wirtschafts- und Rechtswissenschaften, Universität Oldenburg., 2008.

[14] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH Computer Architecture News*, vol. 35, pp. 13-23 2007.

[15] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models," in *HotPower '08 Conference on Power aware computing and systems*, San Diego, CA, USA, 2008, pp. 3-3.

[16] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Che, "AppScope: Application Energy Metering Framework for Android Smartphones using Kernel Activity Monitoring," in *USENIX ATC*, Bosten, MA, USA, 2012, pp. 387-400.

[17] A. Brunnert, C. Vögele, and H. Krcmar, "Automatic Performance Model Generation for Java Enterprise Edition (EE) Applications.," in *Computer Performance Engineering*, Venice, Italy, 2013, pp. 74-88.

[5] http://www.google.com/analytics/