Maximilian Eibl, Martin Gaedke. (Hrsg.): INFORMATIK 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 1591

Measuring the Capability of Smartphones for Executing Context Algorithms

Dennis Kroll[1] and Klaus David[2]

Abstract: While the rise of context aware apps remains, the question arises whether algorithms for context processing can be applied to any smartphone and for any user. In this paper, we propose to test context algorithms for each specific smartphone, user, and situation to which a context algorithm is applied. Towards this, we present our framework which automatically measures all relevant information about an app´s runtime properties, i.e. resource utilization and the time taken to process sensor values by a context algorithm. Both information can be automatically returned to the app developers in order to improve the app. We implemented the framework on Android and tested it using the example of a state-of-the-art context algorithm to find out whether our test smartphone is capable of running the context algorithm without delays and deadlocks.

Keywords: smartphones, apps, context awareness, context algorithms, evaluation, usability

1 Introduction

The first device which was called a smartphone and which was available to the public was the *IBM Simon* in the year 1994. With a price of about 900 $, it came with a touch screen and served basic applications such as calendars, calculators, and an email application. After the first smartphones were developed, the trend has kept rising during the last two decades. Nowadays, smartphones are the daily companion of a large fraction of people worldwide. In 2016, the global population was 7.39 billion, and 2.1 billion of them already owned a smartphone. This means about 30 % of the global population are smartphone owners. This is not surprising, since nowadays smartphones are often at low cost and come with a variety of useful features. Modern apps connect billions of people via social media, enable secure online banking, some apps are even aware of their users´ contexts. Context, as defined by A. K. Dey et. Al., is "any information that can be used to characterize the situation of an entity." [Ab99]. The entity can be a smartphone user, and examples for user contexts are the user´s location or activity. More and more apps are aware of such contexts: A fitness app is aware of the activities its user does; An Ambient Assisted Living app detects and reacts when its user falls, and so on. There are two basic requirements for supporting context aware apps on smartphones. First, sensors such as accelerometers, magnetometers, and compasses are required. These sensors measure information about the user´s activity and environment. Secondly, a processor and memory is required to apply

[1] University of Kassel, Chair for Communication Technology, Wilhelmshöher Allee 73, 34121 Kassel, dennis.kroll@comtec.eecs.uni-kassel.de
[2] University of Kassel, Chair for Communication Technology, Wilhelmshöher Allee 73, 34121 Kassel, david@uni-kassel.de

doi:10.18420/in2017_159

context algorithms to sensor data. Context algorithms input sensor values, process them and interpret them as contexts. Those algorithms consist of several steps, i.e. value pre-processing, feature extraction, context classification and context prediction. Note that context algorithms do not nessesarily consist of all of these steps, and a concrete step implementation varries for different algorithms. In earlier days, smartphones were mainly used for data acquisition. The acquired data was not directly processed by a context algorithm on the smartphone up to the smartphones´s little processor and memory performance. Instead, data was transferred to a server with higher ressource performance. Context algorithms were then applied at the server´s side. Modern smartphones gained much ressource performance what makes the transmission of data to external computers often unnessesary. However, when applying context algorithms on smartphones, the question arises whether context algorithms can be applied to any smartphone, in any situation and for any user. While app developers test and adjust their apps in limitted test environments, they cannot reproduce any situation to which an app might be applied. In real world conditions there is a high number of different devices and users, and innumerable situations. In some cases, an app might lead to problems such as delays or deadlocks of a smartphone´s operating system. Thus, during the development of an app, it is not possible to make a prognosis of the general feasibility of using a context algorithm in a smartphone. One option to get user and smartphone specific feedback is to lookup app ratings at app suppliers such as the Google Play Store or the Apple App Store. Unfortunately, it takes time till ratings are created, and ratings are often more general (such as “the app doesn´t work well”) what makes it hard for developers to spot problems. Therefore, we propose to test context algorithms in the real world for each specific smartphone, user, and situation to which the algorithm is applied. Towards this, in this paper we present our framework which automatically measures all relevant information about the utilized ressources and the times taken to process sensor values by a context algorithm. This information can be returned as feedback to the app developers and can be used to further improve the app. We implemented our framework on the mobile operating system Android and tested it using the example of a state-of-the-art context algorithm to find out whether our test smartphone is capable for executing the algorithm without delays and deadlocks. The contributions of this paper are: 1. explaining how we measure a smartphone´s capability for executing a context algorithm; and 2. presenting and using our framework to evaluate the capability of a specific smartphone to execute a specific context algorithm.

2 Related Work

For the evaluation of several context algorithms, meaningful information about a smartphone´s capability for executing context algorithms were measured, i.e. resource utilization and processing times for sensor values. In [Be10], a service for recognizing activities on mobile phones is presented. To answer the question of capability, the authors preliminary measured the CPU utilization when running and when not running an algorithm. The authors of [La14] present an extended version of *BeWell*, a mobile app to

measure a person´s wellbeing. As a part of the app´s evaluation, the battery drain when running their algorithm on different smartphones was analyzed. In [Kj12], another study on battery drain is presented. The authors focused on building low-power location-based services and analyzed the power consumed by GPS, Wi-Fi, and other sensors which can be used for localization. Besides the energy consumption and the resulting battery drain, the authors of [Re10] also investigated the memory utilization of mobile phones which run an context algorithm. In [LL12], a mobile platform for real-time human activity recognition is presented. The authors not only investigated resource utilization, but also the *response time* which is the time their test smartphone needed to process the pre-processing, feature extraction, and context classification steps of their algorithm. The main difference of the above approaches and the framework presented in this paper is that our framework automatically measures complete and detailed runtime information without requiring any user action. Furthermore, our framework is not limited to the app development process, but can also be applied after an app was released. This allows to evaluate context algorithms on the user´s side, and thus, beyond any situations which could be reproduced in a test environment.

3 Measuring the Capability of Smartphones for Executing Context Algorithms

To find out whether a smartphone is capable for executing a context algorithm, our framework answers two questions: 1. How much CPU, memory, and battery is utilized? and 2. How much time takes it to process sensor values? The results of the first question indicate whether a user can use a smartphone as usual. Otherwise, if the CPU utilization is at maximum for a longer duration, the smartphone will no more response to the user´s commands swiftly. The same effect appears when the memory runs full since the CPU is then busy with managing and shifting data from one memory (RAM) to the other (hard disc) and back. The third resource, the battery level, indicates how long the smartphone will operate. From this information it turns out whether the operation time of a smartphone is still sufficient when executing a context algorithm. From the second question one can find whether the smartphone is able to process sensor values swiftly. Otherwise, when the processing takes too long, new sensor values might be sampled faster than they can be processed. In this case, a context algorithm does no more operate in real time. Furthermore, a delay in processing times will get larger with time. We measure processing times separately for each step of a context algorithm in order to ease searching faults, if any.

The design of our framework is presented in Fig. 1. At the right side, a user who can interact with an app which contains a context algorithm is shown. At the left side, important hardware components are shown, i.e. the sensor(s) to sample sensor values, and the smartphone´s storage to write measurement results. The middle component is our framework, or “Measurement-App” since it is an app itself.

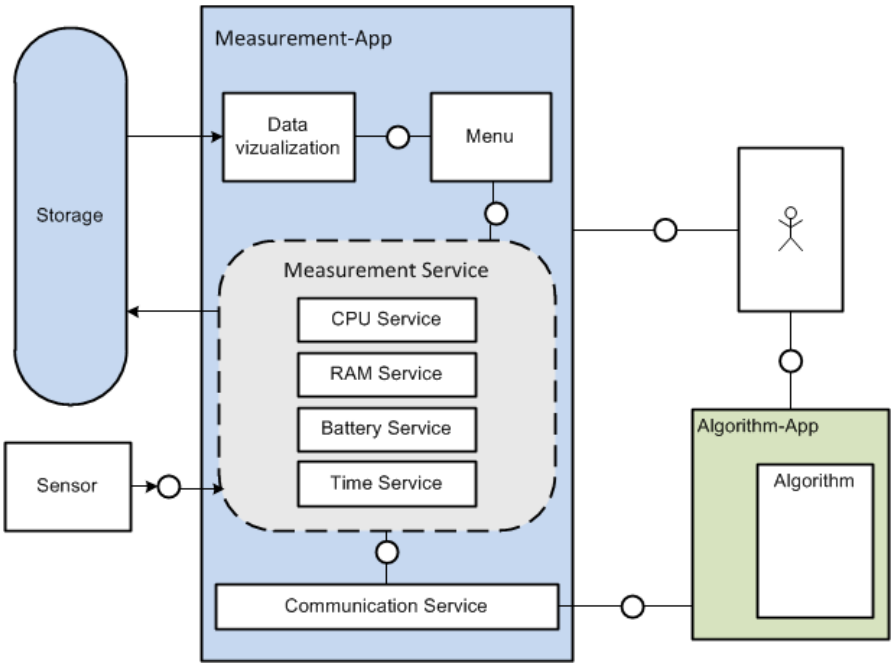

Fig. 1: Component diagram of our framework

The components of the Measurement are:

- **Communication Service:** This service forwards sensor values to the Algorithm-App. The Algorithm-App in turn reports when values are processed. While the Measurement-App is static, the Algorithm-App must implement a simple, pre-defined interface to enable the communication.
- **Measurement Service:** This service encapsulates the three services which log the resource utilization and one service which logs the sensor value processing durations.
- **Menu:** A graphical user interface through which a user could start and stop measuring both, properties when the Algorithm-App is running and properties when the smartphone is in idle mode. These user functions are optional and can be triggered automatically as well.
- **Data visualization:** Users can quick-analyze the measurement results themselves. This is optional since the framework was designed to transfer measurements to the app developers, of course only in case that the user permitted this.

4 Prototype and Experiments

We implemented our framework as Android OS application. For experiments, we used our framework to test a context aware app which runs the context algorithm *Implicit Positioning* [KKD16]. This algorithm analyses compass values of a smartphone carried by a person and finds, learns, and recognizes patterns in it. Our experiments results answer whether our test smartphone, a Samsung Galaxy S5, is capable for execution if the sensor sampling rate is 5 Hz. Our framework did a one-hour measurement while *Implicit Positioning* was running, and one while it was not running. In Fig. 2., the resource utilization for both cases is compared. First, the raw and filtered CPU utilization is shown. The CPU is utilized about 30 % more when *Implicit Positioning* is running. However, a CPU utilization about 60 % does not influence the smartphone usability. The memory utilization stays more or less constant and is similar in both cases. This shows that Implicit Positioning does not utilize much memory. Lastly, the battery drain is compared. As expected, the battery drains more when Implicit Positioning is running. From these results one can say that Implicit Positioning works in real-time, but consumes more energy. Next, we analyzed the processing times which are concluded in Tab. 1. Often, the processing

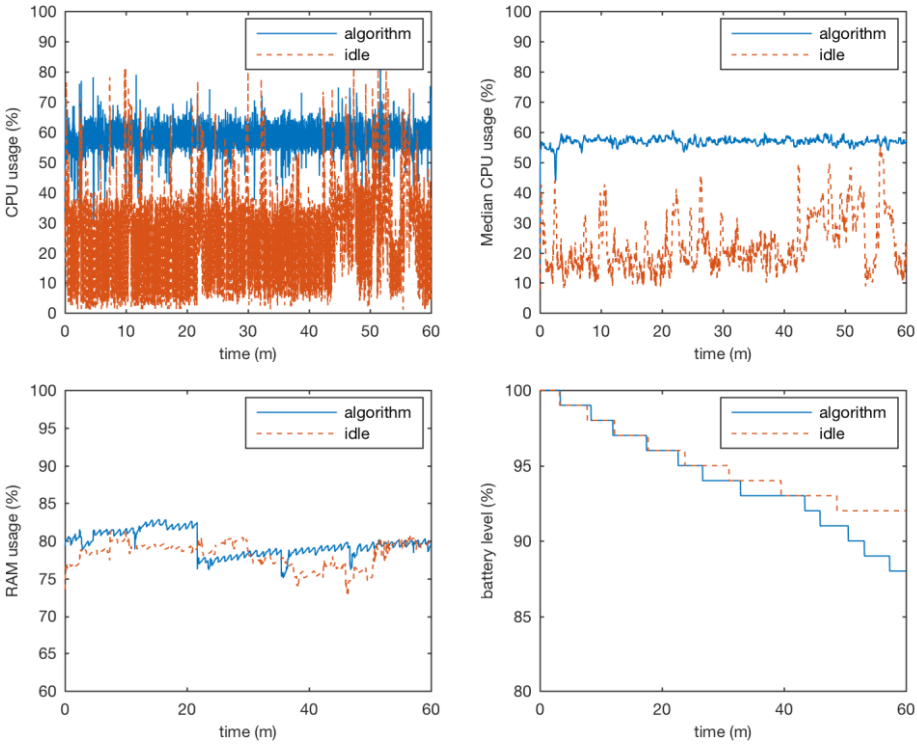

Fig. 2: Resource utilization for a smartphone in idle mode and while running the context algorithm *Implicit Positioning*

times were shorter than 1 ms and rarely longer than a couple of milliseconds. However, neither the longest nor the delta processing times are longer than the timespan between two sensor readings which is 200 ms. This means, that Implicit Positioning processes all steps much faster than the compass sensor is sampled, and no delays appear.

	Shortest processing duration (ms)	Longest processing duration (ms)	Delta
Pre-processing	0 ms	168 ms	3 ms
Feature Extraction	0 ms	8 ms	1 ms
Context Classification	0 ms	7 ms	1 ms

Tab. 1: Sensor value processing durations for steps of the context algorithm *Implicit Positioning*

5 Conclusion

In this paper, we presented our novel framework for measuring the capability of smartphones for executing context algorithms. We explained that both resource utilization as well as processing times of sensor values are important indicators for capability evaluations. We implemented our framework on Android OS and tested it with a context algorithm in order to find out whether our test smartphone is capable of running the context algorithm without delays and deadlocks.

References

[Ab99] Abowd, G. D.; Dey A. K.; Brown P. J.; Davies N.; Smith M.; Steggles P.: Lecture Notes in Computer Science - Towards a Better Understanding of Context and Context-Awareness, vol. 1707, 1999, Berlin, Germany.

[Be10] Berchtold M.; Budde M.; Gordon, D.; Schmidtke H. R.; Beigl M.: ActiServ: Activity Recognition Service for mobile phones. In Proc. of the 14th Int. Symp. on Wearable Comp., Seoul, South Korea, pp. 1–8, 2010.

[Kj12] Kjasrgaard M. B.: Location-Based Services on Mobile Phones: Minimizing Power Consumption. In IEEE Perv. Comp., vol. 11, no. 1, pp. 67–73, 2012.

[KKD16] Kroll D.; Kusber R.; David K.: Implicit Positioning Using Compass Sensor Data. In Proc. of the Int. Conf. on Perv. and Ubiquitous Comp., pp. 732–741, 2016.

[La14] Lane N. D. et.al.: BeWell: Sensing Sleep, Physical Activities and Social Interactions to Promote Wellbeing. In Proc. of Mob. Netw. Appl., vol. 19, no. 3, pp. 345–359, 2014.

[LL12] Lara O. D.; Labrador M. A.: A Mobile Platform for Real-Time Human Activity Recognition. In IEEE Consumer Commun. and Netw. Conf., pp. 667–671, 2012.

[Re10] Reddy S.; Mun M.; Burke J.; Estrin D.; Hansen M.; Srivastava M.: Using Mobile Phones to Determine Transportation Modes. In ACM Trans. Sens. Netw., vol. 6, no. 2, pp. 1–27, 2010.