Profiling Energy Profilers

Erik Jagroep, Jan Martijn E.M. van der Werf, Slinger Jansen
Utrecht University
Dept. of Information and Computing Sciences
Princetonplein 5
Utrecht, The Netherlands
{e.a.jagroep, j.m.e.m.vanderwerf, slinger.jansen}@uu.nl

Miguel Ferreira, Joost Visser
Software Improvement Group B.V.
Amstelplein 1
Amsterdam, The Netherlands
{m.ferreira, j.visser}@sig.eu

ABSTRACT

While energy is directly consumed by hardware, it is the software that provides the instructions to do so. Energy profilers provide a means to measure the energy consumption of software, enabling the user to take measures in making software more sustainable. Although each energy profiler has access to roughly the same data, the reported measurements can differ significantly between energy profilers. In this research, energy profilers are evaluated through a series of experiments on their functionality and the accuracy of the reported measurements. The results show that there is still work to be done before these software tools can be safely used for their intended purpose. As a start, a correction factor is suggested for the energy profilers.

Keywords

Energy profilers, Accuracy, Sustainable software.

1. INTRODUCTION

The search for more environmental friendly information technology (IT) has already had an impact on the energy awareness and energy consumption of the sector [3]. A distinction is often made between 'greening by IT', i.e., using IT to make other industries more sustainable, and 'greening of IT', the process of making IT itself more sustainable. This research focuses on the latter, i.e., on sustainable software. Sustainable software is "software whose direct and indirect negative impacts on economy, society, human beings, and the environment resulting from development, deployment, and usage of the software is minimal and/or has a positive effect on sustainable development" [9].

Typically, sustainability research focuses on hardware aspects, i.e., making the hardware more energy efficient [8]. However, it is the software that determines the use of the hardware, and can therefore be seen as the true consumer of power [13]. In addition, "a single ill-behaving power-unfriendly software component on the system can thwart all of the power management benefits built into the hardware" [12].

An energy profiler (EP) is a software tool that estimates the energy consumption of a system based on the computational resources used by applications, and by monitoring the hardware resources [1, 3, 11]. The use of EPs enables practitioners to investigate the energy consumption behavior of their software without having to invest in specialized hardware. However, many different EPs exist and each has its own internal model to estimate the energy consumption. Although all base themselves on the same set of variables, their estimations differ. Consequently, reliability, and hence accuracy, of the estimations is unclear. For example, these tools interpret variables such as multi-core processors, data access and memory consumption differently [10].

In this research, we focus on the question: *Can energy profilers be used to accurately estimate the energy consumption of software?* In a laboratory environment at the Software Energy Footprint Laboratory (SEFLab) [5], we performed experiments with the objective to evaluate the reliabilty of EPs. In the evaluation we focus on (1) the maturity of the EPs in terms of operationalization, and (2) on the accuracy of the reported energy consumption estimations compared to the actual power consumption.

This paper is structured as follows. In Section 2 we introduce the experiment that was performed with the EPs. The results of the experiments are presented in Section 3 and in Section 4 the threats to validity are discussed. Finally Section 5 concludes the paper and provides suggestions for future work.

2. EXPERIMENT SETUP

In this research we evaluate the accuracy of EPs by comparing the actual energy consumption of the hardware with the reported outcomes of the EPs under study. To this end, idle, varying load and full load scenario's were simulated. In the idle scenario, the system is in rest. For the variable load scenario we chose to simulate a representative real usage scenario of the system by inducing a variable load on the CPU at random intervals using 'SEFLab-Experiments' [1]. Last, in the full load scenario, a continuous full load is placed on the CPU using 'HeavyLoad' [2] (Microsoft Windows) and the

[1] `https://github.com/SEFLab/SEFLab-Experiments`
[2] `http://www.jam-software.com/heavyload/`

SAC'15 April 13-17, 2015, Salamanca, Spain.

ACM 978-1-4503-3196-8/15/04...$15.00.
http://dx.doi.org/10.1145/2695664.2695825

'Stress' package [3] (Linux), maxing out its capacity.

Per EP 35 runs, of approximately one minute, were simulated per scenario, during which three resources were monitored:

- the output of the EP under study;
- the performance measurements of the system; and
- the real energy consumption of the system, monitored by the SEFLab.

Although we acknowledge the importance of the other hardware components [7], we only induce CPU load in this experiment, as it has been identified as the main driver for energy consumption [2, 5, 12].

2.1 Energy profilers

An exploratory search was conducted to identify relevant EPs for this research. For this search, we formulated the following requirements to include an EP in our experiment:

- there is at least a 'beta' version available online;
- intervals between two measurements are at most one second;
- it should run on Ubuntu Linux or Microsoft Windows; and
- it should be compatible with the hardware available in the SEFLab.

For the exploratory search the Google, Scholar Google and Bing search engines were used for querying the terms 'energy profiler(software)', 'software energy profiler', 'energy consumption software'. The term energy was also exchanged for 'power'.

Table 1 presents the selected EPs and their main properties in terms of measurement level and detail. The search showed a clear difference in capabilities between the found EPs. EPs that could not be properly installed in the SEFLab were excluded beforehand from further research.

2.2 Equipment

The experiments were performed on a Dell PowerEdge SC1425 server, referred to as the "test-server". To measure the real energy consumption, we used a Watts Up? PRO (WUP) power consumption meter, which is a physical device used to measure the total power consumption of the test-server directly from the power socket. As some EPs run on different operating systems, we swapped between identical hard disks on which Microsoft Windows 7 and Ubuntu version 12.04 were installed. Further details on the available hardware can be found in [5].

Ideally, considering the nature of electrical power, we would have liked for EPs to measure more frequent than once per second since the equipment in the SEFLab allowed us to measure more than ten thousand times per second. However, no EP found in the exploratory search possessed this ability. Hence the choice was made to only measure with the WUP, which is accurate to 1.5%.

[3] `http://packages.ubuntu.com/search?keywords=stress`

2.3 Experiment protocol

The experiment consists of three scenarios per EP. For each scenario and EP, we performed the following activities.

Preparation Setup of the test-server, including synchronization of the system clocks using the 'network time protocol' (NTP) and the installation and operationalization of the EP under study. Finally, we ensured that unnecessary applications were closed for clean measurements.

Perform run Execution of the different scenarios using the 'SEFLab-Experiments' tool. This tool automatically records time pulses to indicate the beginning and end of a run, providing consistency in the duration, and collecting the data from the different sources.

The tools report the energy consumption in joules per time unit. As the typical time interval is one second, this equals to power (W). To calculate the total energy consumed during a run, we aggregate these measurements and report in Watthour (Wh). Throughout this paper the measurements provided by the EPs are referred to as the *reported measurements*, whereas the *actual measurements* are obtained from the SEFLab, i.e. WUP.

2.4 Reboot vs no reboot

To determine the influence of rebooting the test-server between runs, a small experiment was performed comparing five full load runs with reboot to five full load runs without reboot. Although the runs with reboot reported a higher average (Mdn = 298.78 W) and energy consumption (Mdn = 403 Wh) than without reboot (Mdn = 298.28 W and Mdn = 402 Wh), the difference was not significant (U = 9.00, z = -0.731, p >.05, r = -.23; for both tests). Measurements for the scenario without reboot were more stable, i.e., showing less outliers than the scenario with reboots. A possible explanation is that the initial start-up processes cause these outliers, resulting in the higher measurements. Therefore, we decided not to reboot the test-server after each run.

3. RESULTS

The evaluation of an EP consists of (1) operationalization of the tool, and (2) its accuracy in terms of energy consumption and timeliness. To evaluate the energy consumption, the averages and total energy consumption per run are compared between actual and reported measurements. Timeliness considers whether measurements are reported on the right moment in time, and is done by comparing the energy consumption graphs of each run (cf. Fig. 1).

3.1 Operation

We first consider the operationalization of the EPs.

Joulemeter is straightforward to install and, after calibration using the WUP meter, provides a proper dataset to compare against the actual measurements. Calibration requires the WUP to be connected to the test-server and pressing the 'calibrate' button on the interface of Joulemeter. The interval between measurements is one second; the results are stored in a comma separated values (CSV) file.

Energy Consumption Tools (EC Tools) is a collection of tools, including a core library with sensors and power estimators that can be re-utilized in other programs, a data

Table 1: Specification of the EPs and the extent in which they are included in this research.

	Level					Detail										
Profiler	**IT Environment**	**System**	**Application**	**Process**	**Line of code**	**Hardw. dependent**	**System**	**Process**	**CPU**	**Memory**	**HDD**	**Network**	**Base**	**Calibration required**	**Installable**	**Operationalizable**
Active Energy Manager (W, L)	✓					✓								?	-	-
Computer Power Log (W)		✓					✓							?	-	-
EC Tools (L)			✓				✓	✓						✓	✓	✓
Energy-aware profiler (W)		✓							✓	✓	✓		✓	?	-	-
Eprof (OS unknown)		✓	✓		✓				✓	✓	✓		✓	✓	-	-
ESSaver (W)	✓		✓	✓				✓	✓	✓	✓	✓		✓	✓	-
Hardware Sensors Monitor * (W)		✓							✓	✓	✓	✓		?	-	-
Joulemeter (W)		✓	✓						✓	✓	✓		✓	✓	✓	✓
PowerAPI (W, L)		✓		✓					✓	✓	✓			-	-	-
PowerTOP (L)		✓		✓				✓						✓	✓	-
powometer (W)		✓				Unknown								?	-	-
pTop (L)			✓						✓		✓	✓		✓	✓	-
pTopW (W)			✓						✓		✓	✓		✓	✓	-
Sensorsview * (W)		✓							✓	✓	✓	✓		?	-	-

* = voltages only, W = Windows, L = Linux

acquisition tool, a monitoring tool on the level of processes, and an application power profiler. It provides real-time resource usage and power estimations for the running processes and is built to allow calibration based on the power consumption reported by the machine's Advanced Configuration and Power Interface (ACPI) subsystem or by external power meters.

Although the installation is straightforward, calibration is required before EC Tools can be used. Unfortunately, EC Tools had to be adjusted [4] to work with the WUP device. Once fully operational EC Tools provides a CSV file containing the power consumption measurements. The interval between two measurements is one second.

pTop [4] relies on a MySQL database to store data it collects and produces. Despite a successful installation and calibration, we did not manage to get pTop to produce any energy estimations. After code inspection we found that two functions responsible for inserting data in the "*process_energy*" and "*device_energy*" tables, named "*insert_ process_energy*" and "*insert_device_energy*", were never called at in the pTop code base. We have been in contact with the developer of pTop, but contact was broken and the decision was made to exclude pTop from further research.

pTopW does not share the same code-base of its Linux counterpart and is calibrated by providing power characteristics of the hardware (e.g. thermal design power of the CPU) in a calibration file. Although most hardware characteristics could be obtained via the respective vendors, there were parameters in the file that we did not understand properly. Documentation on this matter was lacking and contact with the developer did not provide the required clarity. Using a configuration file containing all information that we could provide at that time, pTopW produced estimations of 50+ kWh for the CPU alone. The inability to calibrate led to the exclusion of pTopW from further research.

PowerTop is fundamentally different from the other EPs as it uses an external measurement of the power being drawn and breaks this down per process using the computational resource consumption. Since the powermeter that PowerTop is designed to work with was not available during the experiment, the source code was adapted to read the power consumption from the WUP meter [5]. Then, conform described functionality, PowerTop ran for over half an hour to collect enough data points to produce estimations. Unfortunately we were not able to get PowerTop to produce energy consumption estimations and thus excluded the tool from further research.

One other potential problem we noticed in the source code of PowerTop, was that the external power sensor was only called before measurement starts. In the varying load scenario this could mean that that PowerTop will not be able to properly keep track of variations in power consumption.

ESSaver is composed of a data collection agent, that runs as a windows service, and a reporting tool, that produces energy consumption reports based on the collected data. With the help of the developer, we were able to configure the EP to work with the SEFLab hardware and perform the experiments. Unfortunately, we were not able to transform the data collected by ESSaver into valid power estimations.

We can only speculate about the reasons behind this problem, but one observation is that the agent installed on the test-server is more mature than the reporting tool. The agent is installed with just a few clicks of the mouse, whereas much more configuration is involved for the reporting tool which was, at that time, not shipped to the users of ESSaver. As a result, ESSaver was excluded from further research.

[4] `https://github.com/cupertino/ectools/pull/7`

[5] `https://github.com/pyrovski/watts-up`

Table 2: Joulemeter example data and correction factor for the varying load scenario.

Run	SL (Wh)	JM (Wh)	Diff. (Wh)	SL error (%)	JM error (%)	JM corr. (Wh)	Corr. diff.	New error (%)
1	229.925	220.840	9.085	3.95	4.11	232.695	2.770	1.20
2	230.544	217.763	12.780	5.54	5.87	229.453	1.091	0.47
3	253.106	241.310	11.796	4.66	4.89	254.264	1.158	0.46
4	232.446	222.420	10.027	4.31	4.51	234.359	1.913	0.82
5	242.479	227.599	14.881	6.14	6.54	239.816	2.663	1.10
...	...	...	...	...	...	...	...	...
Averages	236.223	224.186	12.037	5.09	5.37	236.221	1.141	0.48

3.2 Accuracy

After first inspection, the accuracy was evaluated of the EPs that could be made operational Tbl. 1. The Joulemeter and EC Tools data were tested for normality [6] to determine the correct tests to apply. For the normally distributed data, the independent samples t-test is used, whereas the non-parametric Mann-Whitney U test has been applied for non-normally distributed data [6].

3.2.1 Joulemeter

Joulemeter provides the richest dataset, containing valid 35 runs for all three scenarios.

Idle: the Joulemeter averages (Mdn = 193 W) are *significantly* higher than the SEFLab averages (Mdn = 184 W), U = 306, z =-3.60, p <.05, r = -0.43. This in contrast to the energy consumption figures where Joulemeter (Mdn = 181 Wh) reports *significantly* lower figures than the SEFLab (Mdn = 182 Wh), U = 393, z = -2.548, p <.05, r = -.30.

Varying load: the averages for Joulemeter (M = 218 W, SE = .6) are lower than the SEFLab (M = 230 W, SE = .56). This difference is *significant* t(68) = 14.27, p <.05 and represents a large effect size r = .87. Concerning the energy consumption again Joulemeter (M = 224 Wh, SE = .0012) reports lower figures than the SEFLab (M = 236 Wh, SE = .0013). This difference is *significant* as well t(68) = 6.67,p <.05 with a medium effect size r = .40.

Full load: For the full load scenario, opposite to the idle scenario, Joulemeter (Mdn = 316 W) reports *significantly* higher averages than the SEFLab (Mdn = 308 W), U = 98, z =-6.043, p <.05, r = -.72. This also holds for the energy consumption, Joulemeter (Mdn = 330 Wh) compared to SEFLab (Mdn = 320 Wh), however this difference is *not significant*, U = 454, z =-1.862, p >.05, r = -.22.

The results are summarized in Tbl. 3, which shows the significant differences from Joulemeter compared to the SEFLab. Figure 2 visualizes the values of the different scenarios as boxplots. Surprisingly, the plots show relatively large overlaps in the idle and varying load. Given the non significant difference this was only expected for the full load scenario. Looking at the average differences in absolute terms we find 2 Wh for the idle, 12 Wh for the varying load and 8 Wh full load scenario.

With regard to the timeliness of Joulemeter, we observe no significant difference compared to the SEFLab. The figures (varying load example in Fig. 1) show spikes and 'lows' at approximately the same moments in time during a run, enabling users to assign power consumption patterns to specific activities that are being performed.

3.2.2 Energy Consumption Tools

Unfortunately the EC Tools dataset is less stable compared by Joulemeter, as the tool turned out to not always produced reliable measurements. For this reason, we obtained 31 for the idle, 33 for the varying load and 34 valid runs for the full load scenario.

Idle: The figures reported for the idle scenario indicate that EC Tools (Mdn = 196 W) reports *significantly* higher figures than the SEFLab (Mdn = 183 W), U = .00, z = -6.765, p <.05, r = -.86. This line continues with the energy consumption where EC Tools (Mdn = 193 Wh) reports *significantly* higher figures than the SEFLab (Mdn = 180 Wh), U = .00, z = -6.765, p <.05, r = -.86.

Varying load: In the varying load scenario, EC Tools (Mdn = 215 W) reports *significantly* lower averages than the SEFLab (Mdn = 226 W), U = 10, z =-6.855, p <.05, r = -.84. For the energy consumption the statistics also indicate that EC Tools (M = 212 Wh, SE = .0004) on average reports lower figures than the SEFLab (M = 225 Wh, SE = .001) and again a *significant* difference is found t(52) = 12.939, p <.05 that represents a large-sized effect = .76.

Full load: The statistics for the full load scenario show that EC Tools (Mdn = 282 W) reports *significantly* lower averages than the SEFLab (Mdn = 318 W), U = .00, z = -7.09, p <.05, r = -.86. The same holds for the energy consumption where EC Tools (M = 278 Wh, SE = .00007) reports lower figures than the SEFLab (M = 313 Wh, SE = .0002). This difference is *significant* t(43.9) = 176.322, p <.05 and represents a large-sized effect r = 1.0.

Statistics indicate that the EC Tools measurements in all cases *significantly* differ from the SEFLab. Looking at the boxplots presented in Fig. 3, this finding is supported through the minimal overlap between measurements. In absolute terms we found an average difference of 11 Wh, 10 Wh and 34 Wh for respectively the idle, varying and full load sce-

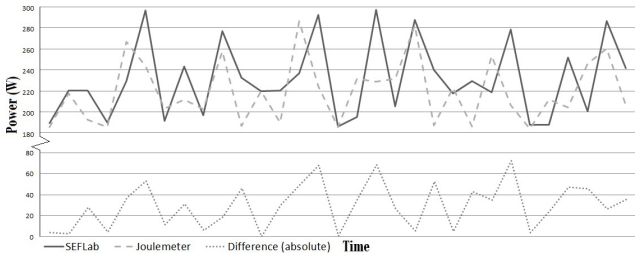

Figure 1: Actual (SEFLab), reported (Joulemeter) and difference in energy consumption over time.

Table 3: EP results compared to the SEFLab.

	Joulemeter		EC Tools	
	Avg.	Consumption	Avg.	Consumption
Idle	↑	↓	↑	↑
Varying	↓	↓	↓	↓
High	↑	-	↓	↓

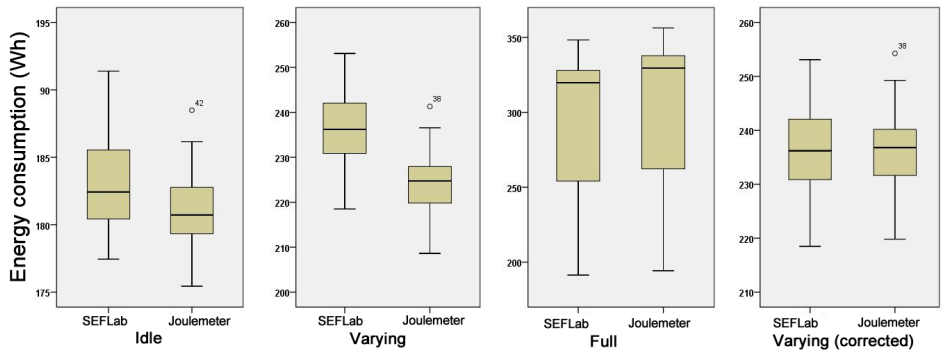

Figure 2: Joulemeter and SEFLab energy consumption averages per scenario.

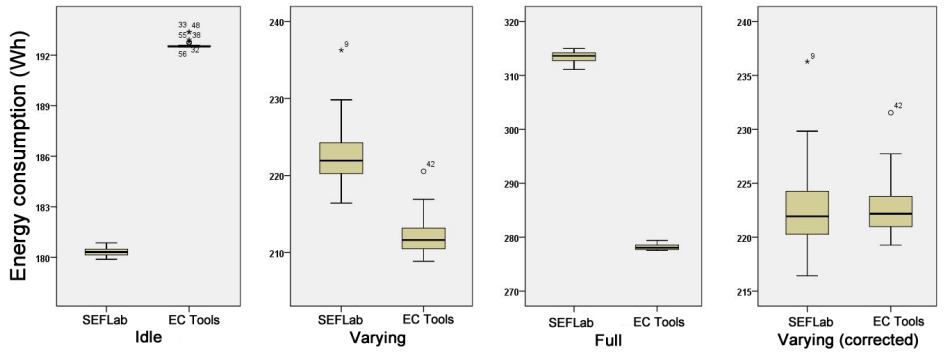

Figure 3: EC Tools and SEFLab energy consumption averages per scenario.

nario. An interesting observation is the fact that the idle and full load measurements are within a dense range of around 3 Wh, which might be caused by the type of operating system.

On the aspect of timeliness, using similar graphs as Fig. 1, less overlap and in general a lower responsiveness was perceived, though still acceptable. Overall, a doubt remains on the operationalizability of EC Tools, considering the stability issues and large differences found in the experiment.

3.3 Correction factor

Looking at the Joulemeter measurements for the varying load scenario Tbl. 2, the error between the actual and reported measurements seems to be relatively constant at 5%. The error percentages presented in this table are calculated by dividing the difference in energy consumption by respectively the SEFLab and Joulemeter measurements. The corrected figures for Joulemeter are calculated using the average error percentage for Joulemeter, i.e., 5.37%.

Table 2 shows that after correcting the Joulemeter varying load measurements with this factor, the average difference between Joulemeter and the SEFLab becomes 0.48%. Repeating these calculations for the varying load scenario of EC Tools, we find that a correction of 4.98% can be applied to reduce the average difference to 0.78%. Looking at the difference in absolute terms, supported by the utmost right boxplots in Fig. 2 and Fig. 3, we hypothesize that the corrections bring the measurements within acceptable bounds.

After correction Joulemeter (M = 236 Wh, SE = 1.29) reports the same mean as the SEFLab (M = 236 Wh, SE = 1.33), and a slightly higher median (see Fig. 2, which is *not significant* t(68) = -.001, p > .05 and has no effect size r = .00. EC Tools (Mdn = 222.2 Wh) also reports a slightly higher median than the SEFLab (Mdn = 221.8 Wh), and again this difference is *not significant* (U = 514, z = -.391, p>.05, r = .05). Although the initial results are promising, further research to investigate and determine the correction factors is required.

4. THREATS TO VALIDITY

The threats to validity are identified according to the four major classes of validity aspects [14, 15]. Construct validity covers identifying the correct operational measures for the concepts being studied. Considering the nature of the experiments and the measurements that were obtained, we ensured that there is no room for ambiguity in interpreting the results.

In the light of the internal validity, it cannot be 100% certain that the only load generated on the test-server was caused by the tools used for experimentation. Although all unnecessary applications were closed, the behavior of services can not be completely controlled. To exclude the influence of (start-up) services as much as possible, the choice was made not to reboot the server between runs. In order to control environmental conditions that could influence the experiment, e.g. room temperature, the test-server was situated in a former server room.

For the external validity we argue that the installation and configuration difficulties will be experienced by anyone wanting to use a specific EP. However, although the measurements themselves are not questioned, the availability or lack of specific hardware might have enabled or inhibited us to get an EP operational. Different hardware setups could

yield different results in getting an EP operational.

Concerning the reliability of the experiment, we argue that the described experiment protocol should yield similar results. One aspect that might differ is the choice to take the measurements as given and not to transform the data when not normally distributed.

5. CONCLUSIONS

In this paper we question EPs in terms of their ability to accurately estimate the energy consumption of software. Through experimentation we evaluate the ability to operationalize EPs and whether the reported energy consumption estimations are accurate compared to the actual energy consumption figures.

For our experiment we were only able to install six out of fourteen EPs found and only two out of those six fully operational for experimentation. Although installation could be performed successfully, configuration turned out to be problematic. Even with the help of the respective developers we could not solve the issues that we came across. Although hardware independence was claimed, the main flaw remained the ability to cope with different hardware configurations.

The actual experiments were performed with Joulemeter and EC Tools, on respectively the Windows 7 (x64) and Ubuntu 12.04 operating systems, where we considered the power consumption averages and the total energy consumption during idle, varying load and full load scenarios. Except for the energy consumption in the full load scenario with Joulemeter, we found significant differences compared to the actual usage. Concerning the timeliness of the measurements, we found that both EPs are acceptable.

Hence we argue that EP in general can not be used to estimate the energy consumption of software yet. Although the tested EPs can be used to get a sense of the energy consumption habits of software, both EPs showed a significant differences in the one minute runs. We expect this difference to become unacceptable when longer periods of measurement are performed. The proposed correction factors of 5.37% for Joulemeter and 4.98% for EC Tools, although further research is required, are a first step in bringing the measurements within acceptable bounds.

Based on our experiment, we identify several directions for future research. A first direction would be to evaluate multiple EPs on different platforms and hardware setups and also evaluate cross-platform EPs. Apart from clarifying the generalizability of our results, more insight could be gained on differences between platforms. A second direction is to investigate how our results, and EPs in general can be used by different stakeholders, e.g. software engineers. Third, is to investigate the results when the experiment consists of lengthier runs. It is our belief that in this case all measurements will significantly differ compared to the actual usage and could provide a test for the proposed correction factor. A final direction is to repeat the experiment at a later moment in time to determine whether progress is made with regard to the development of EPs. Apart from determining their accuracy, EPs might also allow for more detailed measurements (e.g. individual hardware components).

Acknowledgments.

A special word of thanks to the developers for their support to get the EPs operational in the SEFLab.

6. REFERENCES

[1] N. Amsel and B. Tomlinson. Green tracker: a tool for estimating the energy consumption of software. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 3337–3342, New York, NY, USA, 2010. ACM.

[2] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis. Energy-efficient cloud computing. *The Computer Journal*, 53(7):1045–1051, 2010.

[3] G. G. Castane, A. Nunez, P. Llopis, and J. Carretero. E-mc2: A formal framework for energy modelling in cloud computing. *Simulation Modelling Practice and Theory*, 39(0):56 – 75, 2013. S.I.Energy efficiency in grids and clouds.

[4] T. Do, S. Rawshdeh, and W. Shi. ptop: A process-level power profiling tool. In *Proceedings of the 2nd Workshop on Power Aware Computing and Systems*, 2009.

[5] M. A. Ferreira, E. Hoekstra, B. Merkus, B. Visser, and J. Visser. Seflab: A lab for measuring software energy footprints. In *Proc. GREENS*, pages 30–37. IEEE, May 2013.

[6] A. Field. *Discovering Statistics Using SPSS*. Introducing Statistical Methods Series. SAGE Publications, 2007.

[7] A. Kipp, T. Jiang, M. Fugini, and I. Salomie. Layered green performance indicators. *Future Generation Computer Systems*, 28(2):478 – 489, 2012.

[8] P. Lago and T. Jansen. Creating environmental awareness in service oriented software engineering. In E. Maximilien, G. Rossi, S.-T. Yuan, H. Ludwig, and M. Fantinato, editors, *Service-Oriented Computing*, volume 6568 of *Lecture Notes in Computer Science*, pages 181–186. Springer Berlin Heidelberg, 2011.

[9] S. Naumann, M. Dick, E. Kern, and T. Johann. The greensoft model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems*, 1(4):294 – 304, 2011.

[10] A. Noureddine, R. Rouvoy, and L. Seinturier. A review of energy measurement approaches. *SIGOPS Oper. Syst. Rev.*, 47(3):42–49, Nov. 2013.

[11] S. Schubert, D. Kostic, W. Zwaenepoel, and K. Shin. Profiling software for energy consumption. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 515–522, 2012.

[12] B. Steigerwald and A. Agrawal. Green software. *Harnessing Green IT: Principles and Practices*, page 39, 2012.

[13] Y. Sun, Y. Zhao, Y. Song, Y. Yang, H. Fang, H. Zang, Y. Li, and Y. Gao. Green challenges to system software in data centers. *Frontiers of Computer Science in China*, 5(3):353–368, 2011.

[14] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[15] R. Yin. *Case Study Research: Design and Methods*. Applied Social Research Methods. SAGE Publications, 2009.