STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIII**, Number 1, 2018
DOI: 10.24193/subbi.2018.1.01

MEASURING ENERGY EFFICIENCY OF SELECTED WORKING SOFTWARE

CSABA SZABÓ AND EMIRA MUSTAFA MOAMER ALZEYANI

Abstract. Energy consumption is a key performance indicator of any software run on mobile devices. Working or application software is the main category of software where such energy (in)efficient performance becomes accelerated between users and other stakeholders. Measuring energy efficiency is becoming a part of automated and manual performance testing as well – both answering to the increasing usage requirements and addressing acceptance testing optimization. In this paper, we select three software tools – an e-mail client and two social network applications, those energy consumption is being measured and analyzed. We decided to apply very generic profiles during our measurements, where the actions were performed all manually. Our results show that besides the difference in the number of features covered by the software, also their implementation plays an important role in energy consumption. Focusing on a specific feature within the working software does not imply that all quality indicators of it are the best among the software group's implementations.

1. Introduction

Our research focuses on energy consumption of software, which is a relatively new field of software engineering research.

In this paper, we present the results of the application of our measurement setup that relies on the presence of a tester performing usage actions over the system under test. To unify the measurements, i.e. to make them comparable, we decided to select web-based software. In our case, this kind of software also requires a web browser application to run. This application was the same in all cases: Google Chrome.

Received by the editors: April 16, 2018.

2010 *Mathematics Subject Classification.* 68N01, 68M20.

1998 *CR Categories and Descriptors.* D.2.5 [**SOFTWARE ENGINEERING**]: Testing and Debugging – *Monitors*; D.2.8 [**SOFTWARE ENGINEERING**]: Metrics – *Product metrics*.

Key words and phrases. energy efficiency, green software, performance testing.

This paper was presented at the 12th Joint Conference on Mathematics and Computer Science, Cluj-Napoca, June 14-–17, 2018.

These three websites were selected, and on the following basis:

iNotes email client: has been chosen because it contains large of texts and the possibility of downloading and uploading it, with the knowledge that it is possible to contain some pictures or videos sent to it.

Facebook: has been selected on the basis that it contains a different set of content such as (text - pictures - videos - some links, etc.) This is very important for our study of the possibility of monitoring changes in energy on the laptop's processor and more.

YouTube: was chosen on the basis of containing the whole on videos and it is known that because of the animation and sound, this needs high energy to run it.

Together with these assumptions, we also considered that an e-mail client will be consuming less energy during work and (almost) no energy in idle state than the other two applications. As the other two applications both present videos and video comments, it was also an aim to compare if YouTube can master video playback better than Facebook.

The structure of our paper is as follows. In the upcoming section, we present work related to the research field. Then, we present our measurements and results. Finally, we conclude and point out future research directions.

2. Related Work

Computer networks, communication systems [5], data centers [9, 16], huge production of mobile phones [4, 7], and other IT infrastructures have caused severe environmental problems by consuming significant amounts of power [15], increasing greenhouse gas emissions, and lead to pollution during the production and disposal.

This growing concern on energy efficiency may also be associated with the perspective of software developers. Unfortunately, developing energy-aware software [3] is still a difficult task. While programming languages provide several compiler optimizations[14], memory profiler tools, benchmark[10, 12] and time execution monitoring frameworks, there are no equivalent tools and frameworks to optimize energy consumption [8].

Mobile software platforms are targeted by a valuable research activity[6, 11, 13], mainly because of the speed of reaction of stakeholders as device battery drain is a much faster indicator than the energy bill integrating power consumption over days or months. The development is therefore also focusing on tool usage and research, which are capable to determine "energy leaks" or evaluate energy efficiency improvements between versions of software[1].

All these research results could be used in a different way as well. We show that way in this paper. Many researchers do also social networks or other web-based application research[2], where the application requires to run in a browser. Their focus on usability and user experience, on requirements on functionality or speed, but the combination of the above two research goals is missing. We are evaluating energy consumption of a browser running the specific applications – as these do not work without the browser, it would be worthless to evaluate energy efficiency without that required environment. Fuel economy of a car is the best if not used, or we have to consider the driver and road conditions as well while driving.

3. Measurement environment setup

Our study is based on measuring energy efficiency of the Google Chrome browser for the Microsoft Windows operating system using the following hardware/software specs:

- Intel Core i5 CPU @ 2.5 GHz
- 4 GB RAM
- Microsoft Windows 7 Home Premium 64-bit operating system with latest updates installed

The study is based on several factors:

- When using a browser for the purpose of browsing a site such as e-mail, this is known to contain a large text.
- When using the browser for the purpose of browsing a site such as Facebook and this site is known to contain many elements, including texts, images and videos.
- When using the browser for the purpose of browsing a site such as YouTube, especially on the video and contain the texts.
- In order to study, we will use an energy measurement program named "JouleMeter" to read and store the readings of energy, compare and analyze them. Hence, the differences will be identified when using the browser in several ways and how it affects the performance of the laptop and the life of the laptop battery. This tool is estimating energy consumption based on CPU, GPU, monitor brightness etc., therefore it needs to be configured before the first measurement. The setup is automatic, but requires some time as the configuration is preforming a performance benchmark on the host device.

We will use Chrome browser to see different results:

(1) When we use Chrome to open email page as more as static page.
(2) When we use the Chrome browser to open page as Facebook .

(3) When we use the Chrome browser to open page as YouTube.

For a successful measurement, we have to prepare our hardware setup from the operating system as well as to keep in mind technical limits of the Microsoft JouleMeter we used.

(1) We must ensure that the laptop is charged above 75%.
(2) Work on the browser for at least 45 minutes.
(3) Screen brightness was 100% in this study.
(4) Setting of battery: Here we will consider on some setting on battery of the laptop that we worked on it in the study, from the setting of Power Options there are option is Processor power management of battery this is option for these This option will change the speed of the processor clock, and this option gives us the possibility to increase or decrease the ratio according to user requirements. If the rate is changed to less than 100%, energy consumption and heat output will be reduced.

Each measurement is then a simple sequence of steps[1]:

(1) Start the energy consumption monitoring tool
(2) Start the browser or open a new browser window
(3) Enter the application page
(4) Use the application
    - open the newest e-mail, set read/unread 5 emails, click "reply" but close without sending, write an e-mail without attachments, write an e-mail in HTML form, write an e-mail with embedded pictures, write an e-mail with attachments – picture, video, archive file etc.
    - log in to Facebook, set messages read/unread, write a status then delete it, comment, browse statuses/wall, check groups, browse picture galleries, write suggestions, watch posted videos, post something and delete etc.
    - search for artists/publishers on YouTube, watch their activity, write and delete comments, create/modify/delete own watch lists, upload videos, watch videos (turn on/off automatic playback) etc.
(5) Close the browser window
(6) Stop the energy consumption monitoring tool and save collected data

[1]Based on the Intellectual output O1 part/topic "How Green Is Your Process?" of project No. 2017-1-SK01-KA203-035402 (for more details see Acknowledgment).

4. Results

As mentioned above, energy can be estimated after a proper configuration of the JouleMeter tool. The data are stored in a comfortable way that allows direct analysis (Excel or csv file).

We can compare the readings but before that we explain the details on the charts we get:

- The total power chart expresses the value of energy used as a whole.
- The CPU chart expresses the value consumed by the feeder from the total power value as a whole.
- The chart of the program expresses the value of the energy consumed from the CPU value.

4.1. **E-mail client.** The e-mail client was stressed by actions as stated earlier in this paper. The following three figures, Fig. 1, Fig. 2 and Fig. 3 reflect charts from measurements with sampling of 100 milliseconds.

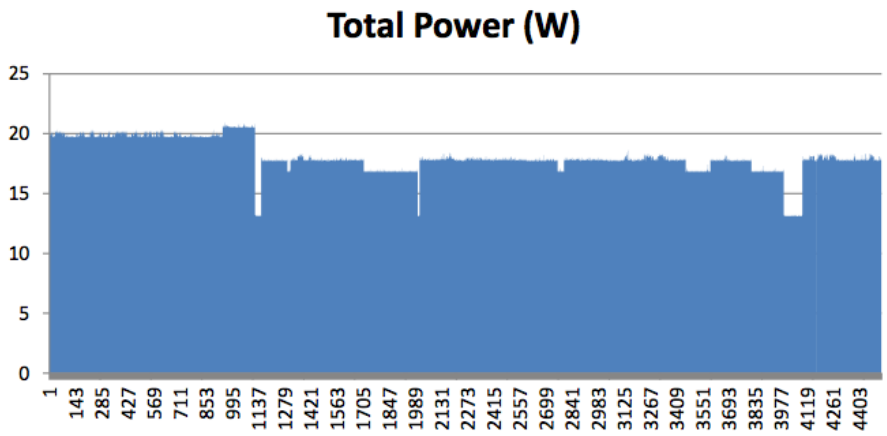

Figure 1. iNotes total power consumption data

Especially Fig. 3 reflects that the application – Chrome, but running the e-mail client only, consumed valuable energy only in the case of user actions. Reading of e-mails is "hidden" as the measurement tool includes the energy consumption of the display as part of the total power consumption chart in Fig. 1.

To finalize our data analysis, Tab. 1 statistically concludes our measurements.

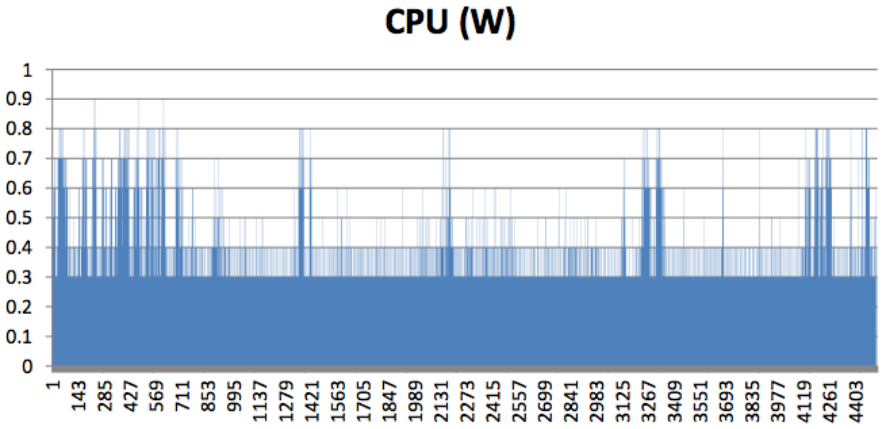

FIGURE 2. iNotes CPU power consumption data

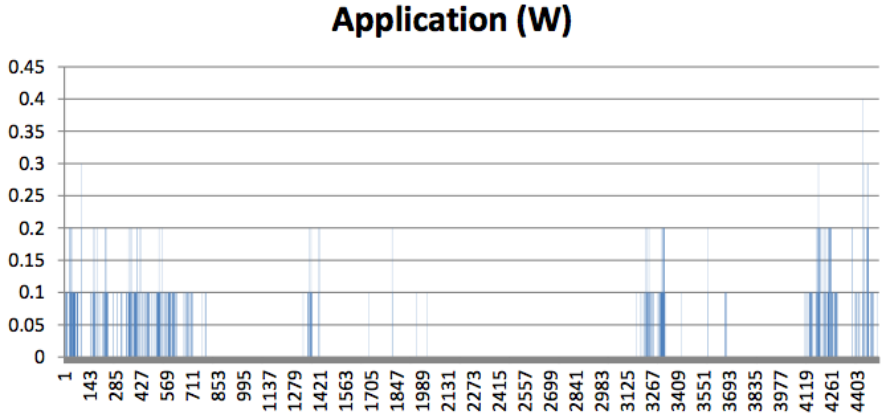

FIGURE 3. iNotes browser power consumption data

TABLE 1. Measured data recapitulation – iNotes (all values displayed in Watts)

	Total power	CPU	Display	Disk	Base	Application
avg	18.066	0.381	4.905	0.002	12.8	0.012
min	13.1	0.3	0	0	12.8	0
max	20.9	0.9	7.4	0.9	12.8	0.4

4.2. **Facebook.** The typical Facebook user (by our definition) used to read the web content, to interact with other users in different ways and discover the dynamics of the web page offered, including watching or commenting videos.

These activities are influenced by the content, but all acceptance testing is. A sample energy consumption profile is presented in Fig. 4, CPU related energy consumption data are displayed in Fig. 5 while browser power consumption data are in Fig. 6.

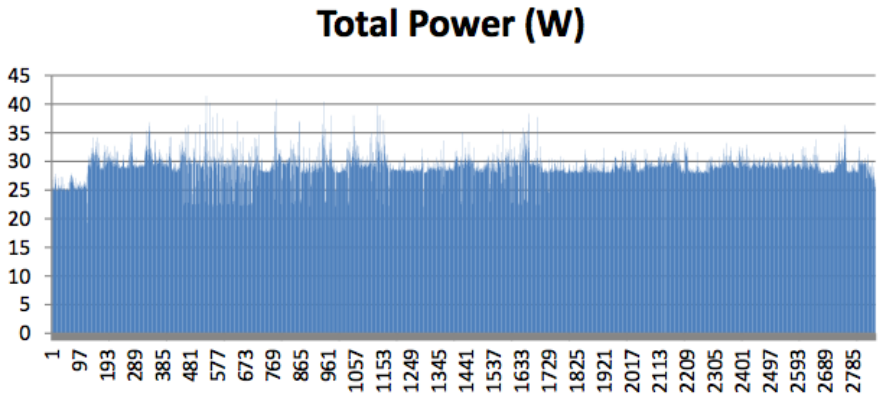

FIGURE 4. Facebook total power consumption data

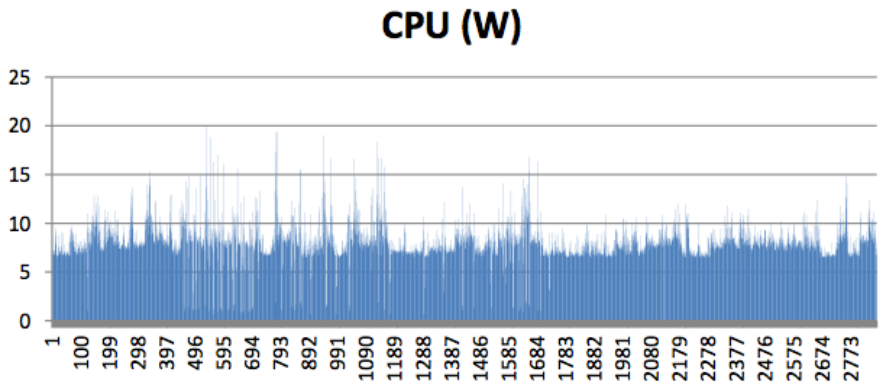

FIGURE 5. Facebook CPU power consumption data

Analysis of browser power consumption can be used to identify actions performed by the user such as starting a video playback, upload of a picture to the gallery or scrolling down the wall or a group page.

Reading of other users' posts is "hidden" as the measurement tool includes the energy consumption of the display as part of the total power consumption chart in Fig. 4.

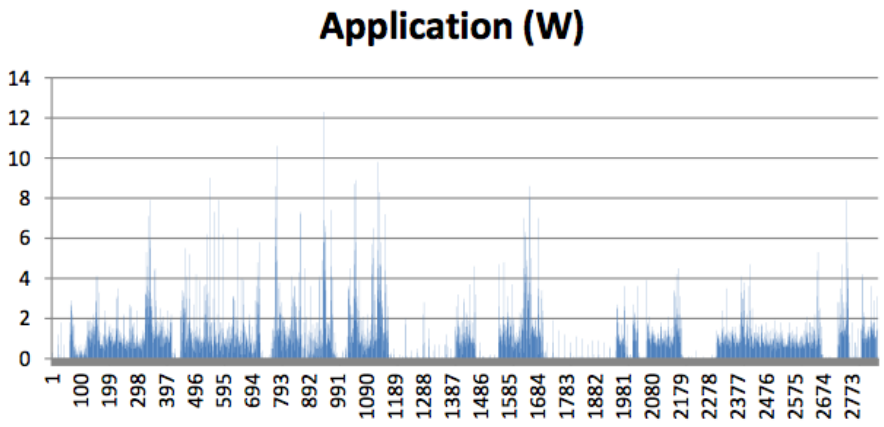

FIGURE 6. Facebook browser power consumption data

To finalize our data analysis, Tab. 2 statistically concludes our measurements.

TABLE 2. Measured data recapitulation – Facebook (all values displayed in Watts)

	Total power	CPU	Display	Disk	Base	Application
avg	29.307	8.05	9.352	0.002	11.9	0.994
min	19.3	0.6	6.3	0	11.9	0
max	41.4	20	9.5	0.7	11.9	12.3

4.3. **YouTube.** YouTube was chosen on the basis of containing the whole on videos and it is known that because of the animation and sound, this needs high energy to run it. At least this was our assumption, but this assumption was accompanied by another one related to a certain level of optimization that one can require from a specialized application.

Figure 7 shows total power consumption measured during an execution sample, more details are picked out in Fig. 8 and Fig. 9.

To finalize our data analysis, Tab. 3 statistically concludes our measurements.

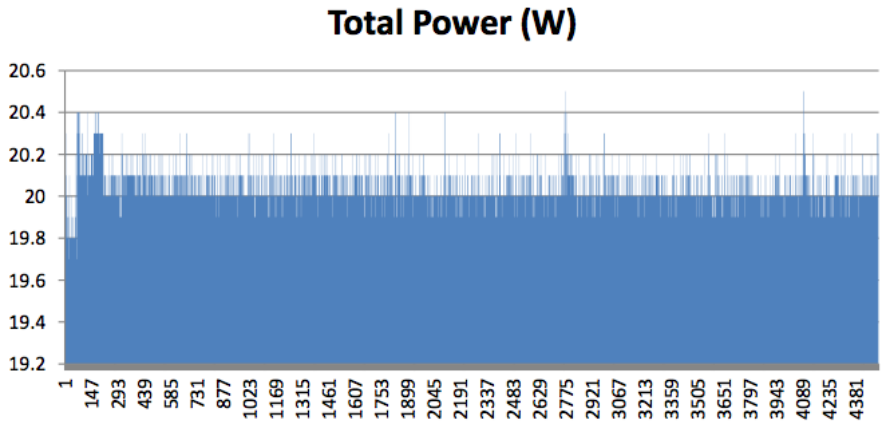

FIGURE 7. YouTube total power consumption data

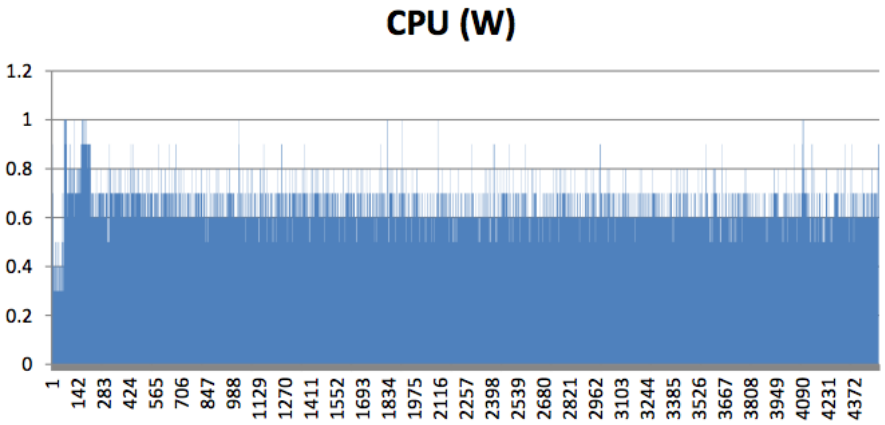

FIGURE 8. YouTube CPU power consumption data

TABLE 3. Measured data recapitulation – YouTube (all values displayed in Watts)

	Total power	CPU	Display	Disk	Base	Application
avg	25.313	9.084	4.302	0.0005	11.9	2.9493
min	17	0.8	4.3	0	11.9	0.1
max	40.8	24.1	4.8	0.3	11.9	14.9

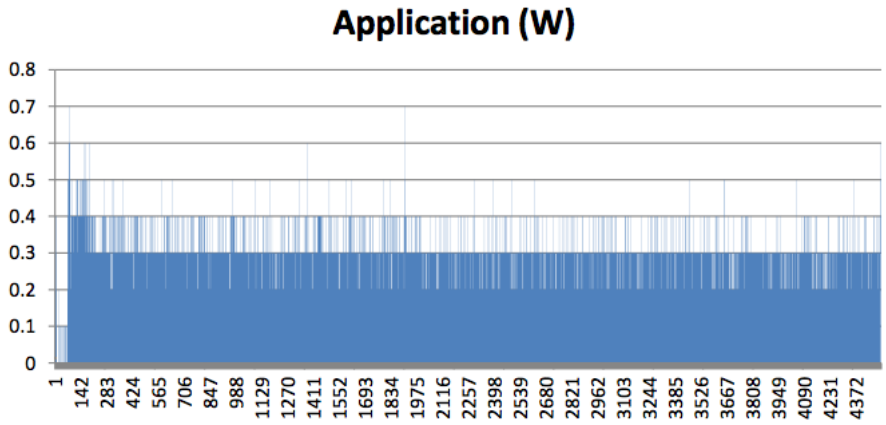

FIGURE 9. YouTube browser power consumption data

One has to note that YouTube has a different energy efficiency profile than the other two applications analyzed – it seems to consume energy all the time that cannot be only in the automatic playlist creation property. Something might be not OK in the environment.

5. Conclusion and Future Work

As mentioned in the last part of the paper, we discovered an anomaly in YouTube behavior related to energy consumption that could be partially caused by continuous preloading of content, but it could be also indicating an error in the solution used. It needs longer analysis to discover the reasons. We will start with replaying the scenarios on another operating system platform, on the top of different version of the browser and/or other browsers before making any further conclusions.

Another difference to the assumptions was the way Facebook could deal with the videos. Even that these are much shorter than the ones in the case of YouTube, the playback did not ask for so much battery power.

When talking about the actions of the measurement itself, as these are very easy, in the future we plan to automate them to provide an automated environment for measurements of comparable setups. Our future work in this area will focus on configuration of a portable integrated development, testing and energy consumption estimation environment. This environment could be used in software development or in the frame of software evolution or initial development teaching subjects to support education on energy efficiency measurement during software testing. It might limit students' creativity by

offering a semi-closed sandbox, as hardware plays a very important role by the current architecture of energy consumption estimation. Some research aims to brake this limitation – we are looking forward to those results to get them also integrated.

Another interesting continuation of our work will be an evaluation of the exactly same test cases using the integrated internet browser on an Android mobile phone or tablet device. Here, the first limitation is that this platform forces the user to use specific apps instead of the browser – E-mail client, Facebook App and YouTube App. Therefore, the comparison results might look different as the limitations given by the browser are partially eliminated, which probably introduces significant differences we will have to face while running the experiments.

Acknowledgment

This paper is part of dissemination of results of the Erasmus+ Key Action 2 (Strategic partnership for higher education) project No. 2017-1-SK01-KA203-035402: "Focusing Education on Composability, Comprehensibility and Correctness of Working Software".

Co-funded by the Erasmus+ Programme of the European Union

The information and views set out in this paper are those of the author(s) and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

References

[1] C. Brandolese, W. Fornaciari, F. Salice, D. Sciuto, *The impact of source code transformations on software power and energy consumption*, Journal of Circuits, Systems, and Computers, Vol. 11, No. 5, pp. 477–502, 2002.

[2] E. Chovancová, M. Chovanec, D. Mičuta, *Social network and forum hybrid*, in Scientific Conference on Informatics, 2015 IEEE 13th International, Nov 2015, Košice, TU, 2015, pp. 124-127, ISBN 978-1-4673-9868-8.

[3] M. Couto, J. Cunha, J. P. Fernandes, R. Pereira, J. Saraiva, *Greendroid: A tool for analysing power consumption in the android ecosystem*, in Scientific Conference on Informatics, 2015 IEEE 13th International, Košice, TU, Nov 2015, pp. 73–78.

[4] J. Flinn, M. Satyanarayanan, *Powerscope: A tool for profiling the energy usage of mobile applications*, in Proc. of the Second IEEE Workshop on Mobile Computer Systems and Applications, WMCSA'99, IEEE Computer Society, 1999.

[5] M. Hudák, Š. Korečko, B. Sobota, *On architecture and performance of LIRKIS CAVE system*, in CogInfoCom 2017, Danvers, IEEE, 2017, pp. 000295-000300, ISBN 978-1-5386-1264-4.

[6] R. Jabbarvand, A. Sadeghi, J. Garcia, S. Malek, P. Ammann, *Ecodroid: An approach for energy-based ranking of android apps*, in Proc. of the Fourth International Workshop on Green and Sustainable Software, GREENS'15, IEEE Press, 2015, pp. 8–14.

[7] D. Li, W. G. J. Halfond, *An investigation into energy-saving programming practices for android smartphone app development*, in Proc. of the 3rd International Workshop on Green and Sustainable Software, GREENS 2014, ACM, 2014, pp. 46–53.

[8] D. Li, Y. Jin, C. Sahin, J. Clause, W. G. J. Halfond, *Integrated energy-directed test suite optimization*, in Proc. of the 2014 International Symposium on Software Testing and Analysis, ISSTA 2014, ACM, 2014, pp. 339–350.

[9] K. Liu, G. Pinto, Y. D. Liu, *Data-oriented characterization of application-level energy optimization*, in Fundamental Approaches to Software Engineering, ser. LNCS, A. Egyed, I. Schaefer, eds., Springer Berlin Heidelberg, 2015, Vol. 9033, pp. 316–331.

[10] N. Pataki, Á. Sipos, Z. Porkoláb, *Measuring the Complexity of Aspect-Oriented Programs with Multiparadigm Metric*, in QAOOSE 2006 Proceedings: 10th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, M. Lanza, F. B. e Abreu, C. Calero, Y.-G. Guéhéneuc, H. Sahraouri, eds., Nantes, Universitá della Svizzera italiana, 2006, pp. 1-10.

[11] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, Y.-M. Wang, *Fine-grained power modeling for smartphones using system call tracing*, in Proc. of the Sixth Conference on Computer Systems, EuroSys'11, ACM, 2011, pp. 153–168.

[12] M. Santos, J. Saraiva, Z. Porkoláb, D. Krupp, *Energy Consumption Measurement of C/C++ Programs Using Clang Tooling*, in Proceedings of the SQAMIA 2017: 6th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications, Z. Budimac, ed., Belgrade, Serbia, 11-13.9.2017, Paper No. 15, 8 pages, Also published online by CEUR Workshop Proceedings No. 1938 (`http://ceur-ws.org`) ISSN 1613-0073.

[13] J. Saraiva, M. Couto, Cs. Szabó, D. Novák, *Towards Energy-Aware Coding Practices for Android*, Acta Electrotechnica et Informatica, Vol. 18, No. 1, 2018, pp. 19–25, DOI: 10.15546/aeei-2018-0003 .

[14] M. Sulír, J. Porubän, *Exposing Runtime Information through Source Code Annotations*, Acta Electrotechnica et Informatica, Vol. 17, No. 1, 2017, pp. 3–9, DOI: 10.15546/aeei-2017-0001 .

[15] Cs. Szabó, J. Saraiva, *Focusing software engineering education on green application development*, in Conference of Information Technology and Development of Education – ITRO 2017, Novi Sad, Serbia, pp. 165–169, ISBN 978-86-7672-302-7.

[16] P. R. Theja, SK. K. Babu, *Evolutionary computing based QoS oriented energy efficient VM consolidation scheme for large scale cloud data centers using random work load bench*, Annales Mathematicae et Informaticae, 46 (2016) pp. 217–241 `http://ami.ektf.hu/uploads/papers/finalpdf/AMI_46_from217to241.pdf` .

Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, Slovakia

Email address: `csaba.szabo@tuke.sk, emira.mustafa.moamer.alzeyani@student.tuke.sk`