Incorporating energy efficiency measurement into CI\CD pipeline

Artem Kruglov
Innopolis University
Innopolis, Russia
a.kruglov@innopolis.ru

Giancarlo Succi
Innopolis University
Innopolis, Russia
g.succi@innopolis.ru

Xavier Vasuez
Innopolis University
Innopolis, Russia
x.vasquez@innopolis.university

ABSTRACT

In this paper we present the method and tool for linking the analysis of a software at the development stage with the efficiency of the developed product during operation mode from the energy consumption perspective. The purpose of the method is to recognize the bottlenecks of a program and provide recommendations for improving the structure and run-time behavior of the software. The developed tool consists of two subsystems: first is responsible for static analysis of the code and relevant software metrics, second performs analysis of the power consumption of the application. Analysis of the outputs of both components allows us to create a close-loop system for continuous analysis and optimization of the developing software product.

CCS CONCEPTS

• **Software and its engineering** → *Software organization and properties*; • **Information systems** → *Data analytics.*

KEYWORDS

Software quality measurement, Energy efficiency, Code metrics, SonarQube, Resource utilization, Incremental software development

ACM Reference Format:
Artem Kruglov, Giancarlo Succi, and Xavier Vasuez. 2021. Incorporating energy efficiency measurement into CI\CD pipeline. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3501774.3501777

1 INTRODUCTION

CI/CD tools are essential part of DevOps culture nowadays. It is highly unlikely that industrial software development will be performed without using Jankins, Github, CircleCI or TeamCity tools, or their analogs [21]. A typical CI/CD pipeline includes actions related to building, packaging, testing, validating, verifying infrastructure, and deploying.

There are a lot of data generated throughout such pipeline, signaling the current state of a developed product. This data can directly or indirectly characterize the quality attributes of the software. The process of such analysis in some sources called Continuous Reliability - the practice of evaluating the overall quality of our code before it is promoted from environment to environment [22]. Apart from common quality attributes, such as maintainability, security, scalability, etc., the energy efficiency of the software becoming the aspect of a software product with utmost importance [2, 4].

In the light of above, the goal of this work is to:

- elaborate the method for the precise software-based energy efficiency analysis of the software,
- embed this measurement procedure into automated CI\CD pipeline,
- investigate the relation between energy consumption and various code metrics.

1.1 Related works

To this aim, researchers have proposed hardware-based tools as well as model-based [9] and software-based [11] techniques to approximate the actual energy profile of applications. Pinto et al. [17] argue that most of the research that connects computing and energy efficiency has concentrated on the lower levels of the hardware and software technological stacks. Maevsky et al. proposed the model-based approach for the analysis of the energy consumption which relies on the calculation of power required to switch bits of RAM [14]. Acar et al. [1] introduced a tool model for energy consumption estimation using CPU, memory, and disk due to the execution of an application at run time. Another tool was proposed by Marantos et al. [15], in which static analysis techniques applied at instruction level to compute energy consumption.

Eastep et al. have created the framework for profiling and optimizing energy efficiency of high performance computing (HPC) systems according to the discovered design patterns in an application [6]. A similar research on the runtime optimization of HPC applications was performed by Roberts et al. [19]. They developed the POSE model that allows developers to compare the potential benefits of power and runtime optimization and determine which approach is most suitable for their code.

Jagroep et al. stated that the software is treated as a single entity which in terms fails in providing detailed insight about energy consumption behavior [8]. Couto et al. suggested the notion of energy debt [5], and that idea is related to the concept of relative energy estimation, discussed in this paper. Moreover, Kehagias et al. investigated the correlation between energy consumption and quality attributes in the domain of computer networks [10].

All these solutions present their own advantages and disadvantages. Hardware-based tools are highly accurate, but requires costly hardware components. Model-based tools require the calibration for each specific hardware device. Software-based approaches are

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-8506-0/21/11…$15.00
https://doi.org/10.1145/3501774.3501777

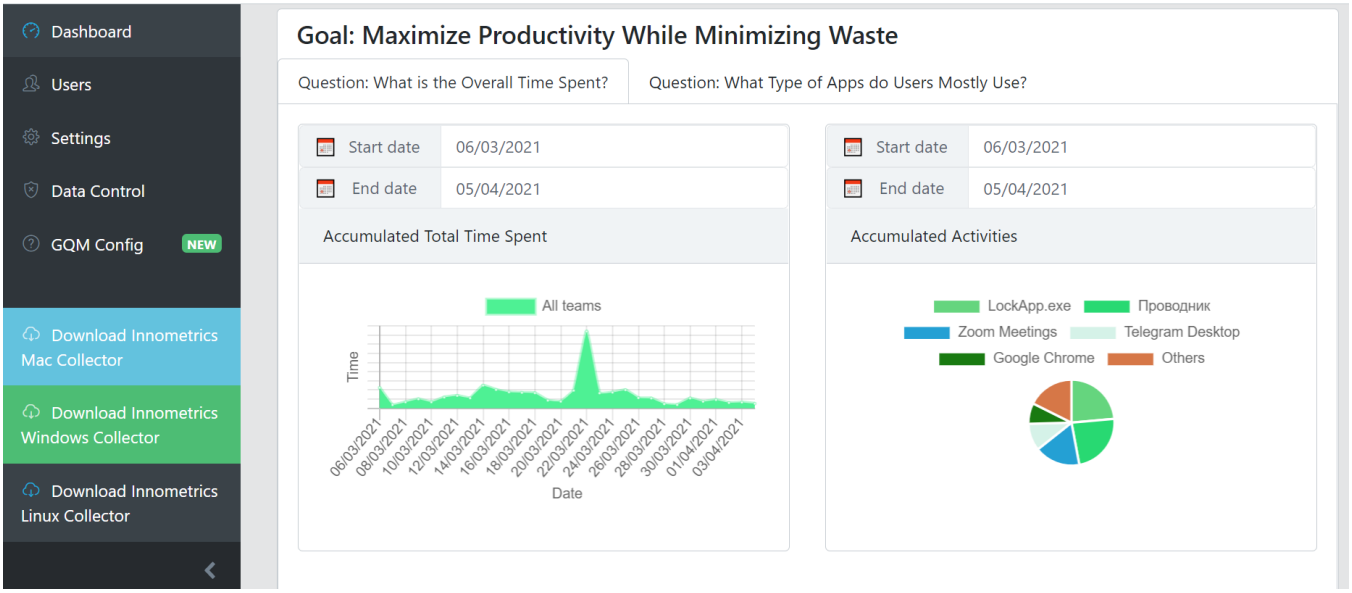

Figure 1: Innometrics framework

cheaper and easier to use than hardware-based tools, but they are believed to be less accurate.

1.2 About the Innometrics framework

The InnoMetrics framework used in this research is mainly focused on collecting and analyzing energy consumption metrics in Windows, macOS and Linux operating systems. As a result, it helps to improve the software development process based on the insights derived from the collected data. InnoMetrics automatically collects various data during the working process and periodically transmits it to the server for further analysis. This allows both developers and managers to be aware of the weaknesses of software development; in particular, the data reports on the energy consumption of both the software process and product [3].

Through processing and visualization of data, we receive up-to-date information about the aspects of development we are interested in real-time (see Figure 1). The framework is flexible and can be customized depending on the size and location of the company and development team.

Data collection is split into two independent processes: one for tracking user actions in the system by analyzing the active application, and another for analysis of resource consumption on the device.

The Innometrics framework is open-source solution, it is available for downloading and testing at https://innometrics-12856.firebaseapp.com/. The community is welcomed to use it for reproducing the experiment in their own environment to verify the results.

1.3 Paper structure

In Section 2 of this paper we discuss how the measurements can be automated and which metrics can be obtained throughout CI\CD pipeline. In Section 3 we present how the process of data collection is organized in the Innometrics framework. In Section 4 we present the idea on how the collected metrics could be used for the sake of energy efficiency analysis. Finally, in Section 5 we demonstrate the implementation of such an assessment in the small experiment and discuss the results. In Section 6 we sum up the finding of this paper.

2 INCORPORATING MEASUREMENTS IN THE PIPELINE

In order to assess the efficiency of the developed product from different perspectives, including energy efficiency, we need to be able to track its evolving structural and functional characteristics throughout the development lifetime. It can be achieved by integrating a static code analyzer (like SonarQube) and project management system (like Trello or GitLab) into CI/CD pipeline.

The advantage of the Innometrics in this case is that it provides an overview of the product evolving over time. After integration with third-party services' agents (see Figure 2) the system start collecting a set of project metrics on the regular basis (lets say, on merge to master's branch).

The data collected from the SonarQube and Trello systems is listed in Table 1. Thus, we can see the changes in various parameters of the codebase and product functionality at a given period of time.

3 RESOURCE CONSUMPTION ANALYSIS

According to the researches, the energy consumption of applications depends on, but is not limited to, various system resources such as device composition, processor type, memory architecture, active sensors. It is also influenced by the signal strength used for data transmission, user and system settings such as display brightness levels [18]. In general there are several solutions for measuring the resulting energy consumption. For the last years many kinds of methods were proposed for measurement, such as hardware

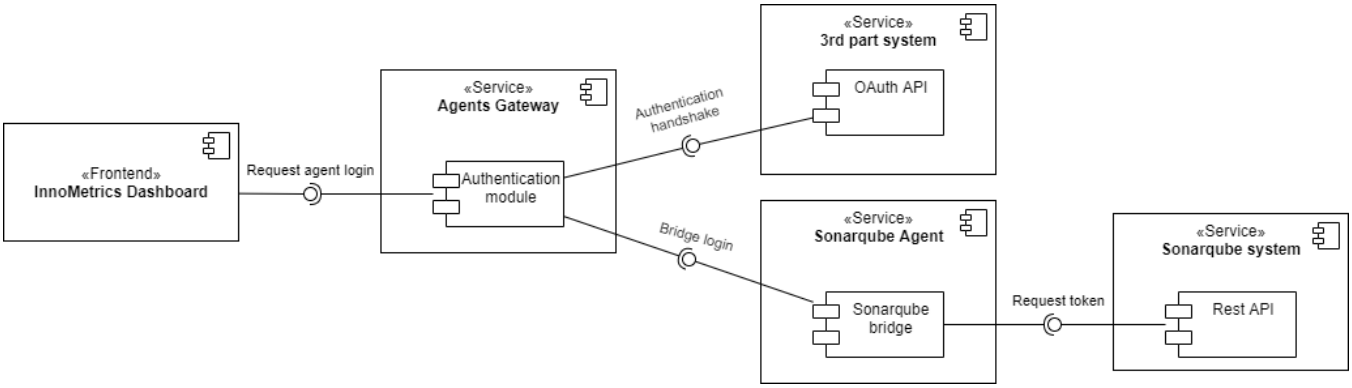

Figure 2: Agents integration with the Innometrics system

Table 1: Data dictionary

Field name	Description
SonarQube	
violations	Total count of issues in all states.
reopenedIssues	Total count of issues in the Reopened state
branchCoverage	For each line of code containing some boolean expressions, the condition coverage checks whether this expression has been evaluated both for true and false or not
lineCoverage	Percentage of lines of code being executed during the unit tests execution
new_technical_debt	Effort to fix all Code Smells raised for the first time on New Code
code_smells	Total count of Code Smell issues
bugs	Number of bug issues.
reliability_rating	rating (A - 0 bugs, B - Minor Bugs, C - Major Bugs, D - Critical Bugs, E - Blocker Bugs)
securityHotspots	Number of Security Hotspots
securityRating	Number of vulnerability issues.
classes	Number of classes (including nested classes, interfaces, enums and annotations).
lines	Number of physical lines (number of carriage returns).
Trello	
cardId	unique id of card on which action occurred
checkItemId	unique id of check item
checkItemState	state of check item
listId	unique id of list on which action occurred
date	date when action occurred
idMemberCreator	unique id of creator of an action
type	type of an action

based methods [20], software based methods [12, 16], CPU usage minimization methods and so on.

The framework we are proposing exploits software-based methods for energy consumption measurement in PCs. For now, we are referring to the resource consumption and time frequency of the processes the user is working on. We track the processes on a timely manner, similar to the time-out approach for power consumption management implemented in the majority of operating systems [13]. Here we collect the following set of metrics to assess the energy consumption:

- Battery status;
- Battery current capacity (if applicable);
- Battery design capacity (if applicable);
- Battery charge (if applicable);
- RAM;
- VRAM;
- Processor load (CPU);
- Graphic processor load (GPU) (if applicable).

The Data Collector receives the list of the processes operating on the user's machine at the given moment along with the information about the actual resources usage and calculates energy consumption. In this case, the frequency of measurements should be limited so that the work of the collector itself does not have a significant effect on the measurement process, although the influence of the instrumentation on the validity of the result cannot be completely excluded in this case.

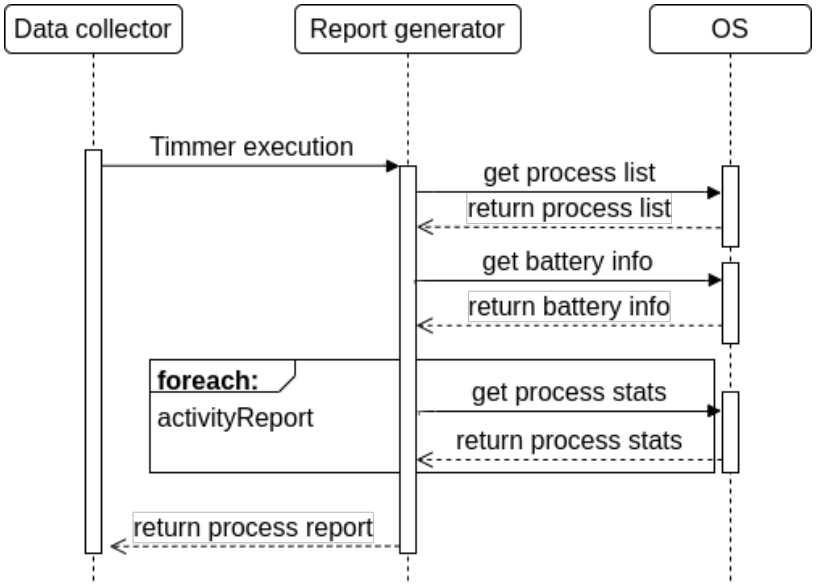

Figure 3: Data collection per process

The process of collecting the data about RAM, VRAM, CPU and battery is quite straightforward and follows the sequence presented in Figure 3.

However, for the case of GPU utilization the situation is more complex: among the most popular vendors - Nvidia, AMD, Intel - each one provides its own performance monitor for data gathering, which have to be preinstalled in the device:

- nvidia-smi for Nvidia;
- radeon-top for AMD;
- intel_gpu_top for Intel;

Thus, the implementation of GPU utilization is based on producer-consumer architecture and looks like the following (see Figure 4). There are different consumers for each type of GPU. They are pushing updates with predefined frequency into the queue, and bounded blocking queue is used to prevent memory leaks and "null" exceptions. Also, with this type of queue we does not care about pulling frequency of consumer.

4 METHOD FOR THE ENERGY EFFICIENCY ESTIMATION

To assess the energy efficiency of the developed software over time, we proposed the approach inspired by the effort estimation technique applied in the agile frameworks (and, actually, relying on it).

In incremental development, each iteration starts with planning when the team discusses and agrees to the scope of work to be done. The worldwide common practice is to use relative estimation, like story points, when analyzing the scope of work.

At the end of iteration the product increment goes through an acceptance procedure, when implemented features are accepted or rejected based on user acceptance testing (UAT). In fact, UAT is a script that should be successfully performed to verify the new functionality of the product. If the functionality was accepted, its story points are added to the velocity of the team for the given iteration.

Innometrics allows users to collect resource utilization data per particular process and correlate it with energy consumption [3]. It means that we can see how much energy was consumed during the time of UAT (in particular, during the time of performing *accepted* tests).

Thus, for each iteration we receive the following data:

- amount of new functionality in story points,
- data from static analyzer, such as new LOC, classes and methods,
- amount of energy consumed while performing accepted UAT.

By mapping this data with one another the following metrics can be achieved to provide not accurate (accuracy is unlikely to be achieved without introducing hardware measurement tools, [20]), but precise estimation of the energy efficiency throughout the development process (see Table 2).

The most interesting metric is the energy efficiency per the iteration. Similar to teams velocity, this metric should not have a high fluctuation, and the spikes indicate the potential treats to energy efficiency that should be refactored.

5 RESULTS AND DISCUSSION

For the experimental purposes we have analyzed the process of the development of a small macOS application. The data was collected since September till November, 2020. For this period, 4 iteration reviews were performed, and during each iteration we supplement UAT with metrics collection procedure. Metrics collected by Innometrics from SonarQube and further processed are LOC, number of classes, number of functions and number of statements (see Figures 5 – 6).

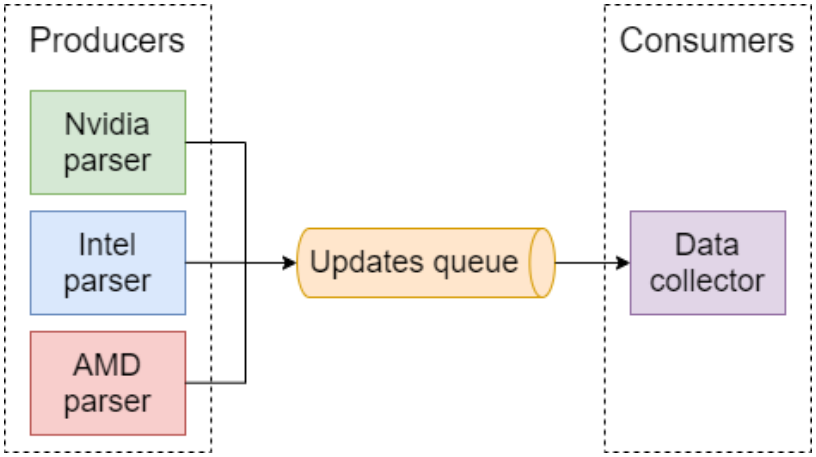

Figure 4: The implementation of GPU utilization

Table 2: Energy efficiency metrics

Name	Formula	Meaning
EE_m	$EE_m = {EE_{UAT}}/{nclasses}$, where *nclasses* - number of new classes as defined by SonarQube, EE_{UAT} - average energy consumption during UAT process	Metric shows the average energy efficiency per developed class (or interface) in product increment.
EE_{struct}	$EE_{struct} = {EE_{m_i}}/{\overline{EE_m}} \cdot 100\%$, where EE_{m_i} - energy efficiency per method for the particular iteration, $\overline{EE_m}$ - average energy efficiency per method	Structural coefficient of energy efficiency. The control limit is empirically determined at the level ±20%
EE_{sp_j}	$EE_{sp_j} = {EE_{UAT_j}}/{\#SP_j}$, where $\#SP_j$ - number of story points per i-th accepted user story, EE_{UAT_j} - average energy consumption during UAT of i-th user story	Metric that correlates functionality with energy consumption.
EE_{it}	$EE_{it} = \frac{\sum SP \cdot \overline{EE_{it}}}{\sum EE_{sp}} \cdot 100\%$, where $\sum SP$ - number of story points of accepted user stories, $\sum EE_{sp}$ - average energy consumption during UAT of accepted user stories, $\overline{EE_{it}}$ - average energy efficiency of previous iterations, $\overline{EE_{it}} = {\sum_{it-1} EE_{it}}/{(it-1)}$	Metric that shows the energy efficiency of the developed functionality per iteration.

Analysis of energy consumption was performed using the macOS Data Collector [7]. Data recorded only for the time of user acceptance testing which followed strict predefined script to minimize impact of the idle time and irrelevant user actions; CPU power consumption data was collected every two minutes. Results of average energy consumption mapped to Story Points completed on an accrual basis are shown in Figure 7. It should be mentioned that Data Collector does not use GPU, thus resource consumption rely on CPU only (Intel Core i7-4770HQ in our experiment).

As can be seen from the obtained preliminary results, the best correlation with the energy consumption during UAT is shown for the "number of story points". The correlation coefficient for this small batch of data is 0.998 with p=.0018. The less significant correlation is obtained for such metrics as "number of statements" and "number of classes". Even though the fact that obtained correlation is more than 0.96 with p<.035 for these metrics, we suggest that for the bigger dataset the results would be much worse.

The more important and informative from the visualization is the absence of spikes in energy consumption data - it indicates that newly added modules or methods have similar energy efficiency characteristic as previous ones. Thus,

$$EE_{inc_1} = 100\% \, (by \, default),$$
$$EE_{inc_2} = 103.9\%,$$
$$EE_{inc_3} = 104.2\%,$$
$$EE_{inc_4} = 109.4\%.$$

These results and measurement itself refer us to the notion of energy debt, introduced by Couto et al. [5]. EE_{inc} metric serves as indicator of the sustainability of the developing product from the energy efficiency perspective. In our small experiment it demonstrates even increasing trend from EE_{inc_1} to EE_{inc_4} which can be interpreted as slightly improvement in energy efficiency (probably, due to refactoring activities).

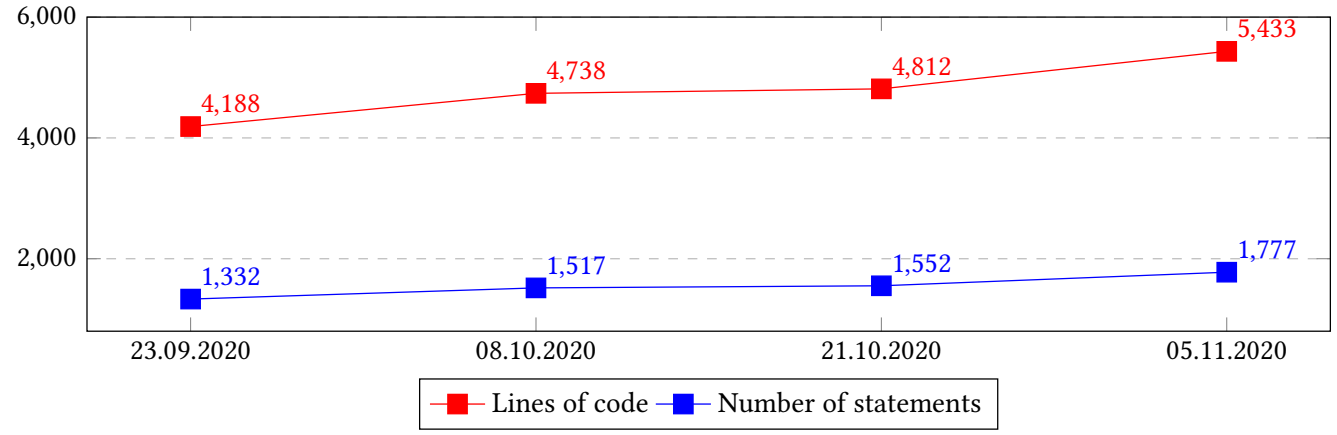

Figure 5: Size oriented metrics

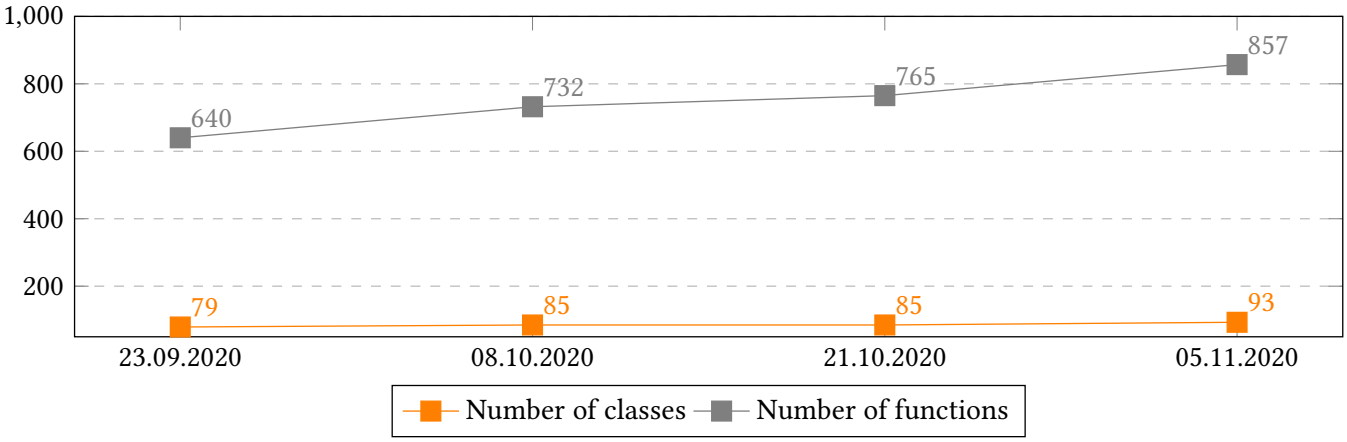

Figure 6: Structure oriented metrics

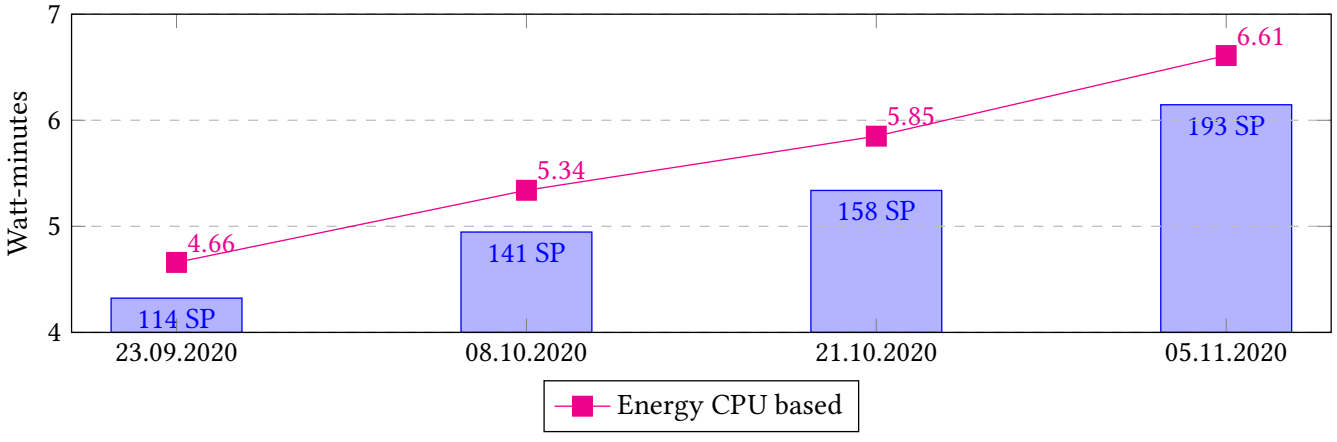

Figure 7: Average energy consumption

6 CONCLUSION

In this paper we present the approach of assessing the energy efficiency of the developing software. The approach is based on the idea of relative assessment of the resource consumption from iteration to iteration rather than absolute estimation. Such an approach provides an opportunity for measuring energy efficiency without additional hardware tools or complex device-specific models.

The Innometrics framework was designed for the purpose of software development process analysis. The data, which it collects from user machine's OS and third party systems provide us with all necessary metrics so we are able to analyse the energy efficiency trend throughout development time and perform corrective actions on purpose.

The idea of energy efficiency assessment depends not only on the usage of Innometrics and additional systems (such as SonarQube), but on the strict adherence to scripted UAT procedure and incorporating measurement in the CI/CD pipeline. Only the consistency of the measurement process can result valuable insights on energy efficiency of the product.

ACKNOWLEDGMENTS

This study was funded by Russian Science Foundation (grant number 19-19-00623).

REFERENCES

[1] Hayri Acar, Gülfem I Alptekin, Jean-Patrick Gelas, and Parisa Ghodous. 2016. The Impact of Source Code in Software on Power Consumption. *International Journal of Electronic Business Management* 14 (2016), 42–52. https://hal.archives-ouvertes.fr/hal-01496266

[2] Costas Bekas and Alessandro Curioni. 2010. A new energy aware performance metric. *Computer Science - Research and Development* 25, 3 (01 Sep 2010), 187–195. https://doi.org/10.1007/s00450-010-0119-z

[3] Paolo Ciancarini, Shokhista Ergasheva, Zamira Kholmatova, Artem Kruglov, Giancarlo Succi, Xavier Vasquez, and Evgeniy Zuev. 2020. Analysis of Energy Consumption of Software Development Process Entities. *Electronics* 9, 10 (Oct. 2020), 1678. https://doi.org/10.3390/electronics9101678

[4] Luis Corral, Anton B. Georgiev, Alberto Sillitti, Giancarlo Succi, and Tihomir Vachkov. 2014. Analysis of Offloading as an Approach for Energy-Aware Applications on Android OS: A Case Study on Image Processing. In *Mobile Web Information Systems - 11th International Conference, MobiWIS 2014, Barcelona, Spain, August 27-29, 2014. Proceedings*. 29–40. https://doi.org/10.1007/978-3-319-10359-4_3

[5] Marco Couto, Daniel Maia, João Saraiva, and Rui Pereira. 2020. On energy debt. In *Proceedings of the 3rd International Conference on Technical Debt*. ACM. https://doi.org/10.1145/3387906.3388628

[6] Jonathan Eastep, Steve Sylvester, Christopher Cantalupo, Brad Geltz, Federico Ardanaz, Asma Al-Rawi, Kelly Livingston, Fuat Keceli, Matthias Maiterth, and Siddhartha Jana. 2017. Global Extensible Open Power Manager: A Vehicle for HPC Community Collaboration on Co-Designed Energy Management Solutions. In *Lecture Notes in Computer Science*. Springer International Publishing, 394–412. https://doi.org/10.1007/978-3-319-58667-0_21

[7] Shokhista Ergasheva, Dragos Strugar, Artem Kruglov, and Giancarlo Succi. 2020. Energy Efficient Software Development Process Evaluation for MacOS Devices. In *IFIP Advances in Information and Communication Technology*. Springer International Publishing, 196–206. https://doi.org/10.1007/978-3-030-47240-5_20

[8] Erik Jagroep, Jan Martijn van der Werf, Sjaak Brinkkemper, Leen Blom, and Rob van Vliet. 2016. Extending software architecture views with an energy consumption perspective. *Computing* 99, 6 (2016), 553–573. https://doi.org/10.1007/s00607-016-0502-0

[9] Georgios Kalaitzoglou, Magiel Bruntink, and Joost Visser. 2014. A Practical Model for Evaluating the Energy Efficiency of Software Applications. In *ICT4S*.

[10] Dionysios Kehagias, Marija Jankovic, Miltiadis Siavvas, and Erol Gelenbe. 2021. Investigating the Interaction between Energy Consumption, Quality of Service, Reliability, Security, and Maintainability of Computer Systems and Networks. *SN Computer Science* 2, 1 (Jan. 2021). https://doi.org/10.1007/s42979-020-00404-8

[11] Ah-Lian Kor, Colin Pattinson, Ismail Imam, Ibrahim AlSaleemi, and Oluwafemi Omotosho. 2015. Applications, energy consumption, and measurement. In *2015 International Conference on Information and Digital Technologies*. IEEE.

[12] Weifeng Liu, Jie Zhou, Bin Gong, Hongjun Dai, and Meng Guo. 2018. Improving the energy efficiency of data-intensive applications running on clusters. *International Journal of Parallel, Emergent and Distributed Systems* (April 2018), 1–14. https://doi.org/10.1080/17445760.2018.1455835

[13] Yung-Hsiang Lu and G. De Micheli. 2001. Comparing system level power management policies. *IEEE Design & Test of Computers* 18, 2 (2001), 10–19. https://doi.org/10.1109/54.914592

[14] D. A. Maevsky, E. J. Maevskaya, and E. D. Stetsuyk. 2016. Evaluating the RAM Energy Consumption at the Stage of Software Development. In *Green IT Engineering: Concepts, Models, Complex Systems Architectures*. Springer International Publishing, 101–121. https://doi.org/10.1007/978-3-319-44162-7_6

[15] Charalampos Marantos, Konstantinos Salapas, Lazaros Papadopoulos, and Dimitrios Soudris. 2021. A Flexible Tool for Estimating Applications Performance and Energy Consumption Through Static Analysis. *SN Computer Science* 2, 1 (Jan. 2021). https://doi.org/10.1007/s42979-020-00405-7

[16] Dario Di Nucci, Fabio Palomba, Antonio Prota, Annibale Panichella, Andy Zaidman, and Andrea De Lucia. 2017. Software-based energy profiling of Android apps: Simple, efficient and reliable?. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE. https://doi.org/10.1109/saner.2017.7884613

[17] Gustavo Pinto and Fernando Castor. 2017. Energy efficiency. *Commun. ACM* 60, 12 (2017), 68–75. https://doi.org/10.1145/3154384

[18] Ahmad Rahmati, Angela Qian, and Lin Zhong. 2007. Understanding human-battery interaction on mobile phones. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services - MobileHCI '07*. ACM Press. https://doi.org/10.1145/1377999.1378017

[19] Stephen I. Roberts, Steven A. Wright, Suhaib A. Fahmy, and Stephen A. Jarvis. 2019. The Power-optimised Software Envelope. *ACM Transactions on Architecture and Code Optimization* 16, 3 (2019), 1–27. https://doi.org/10.1145/3321551

[20] Rubén Saborido, Venera Venera Arnaoudova, Giovanni Beltrame, Foutse Khomh, and Giuliano Antoniol. 2015. On the impact of sampling frequency on software energy measurements. (July 2015). https://doi.org/10.7287/peerj.preprints.1219v2

[21] Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. 2017. Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access* 5 (2017), 3909–3943. https://doi.org/10.1109/access.2017.2685629

[22] Tali Soroker. 2018. Continuous Reliability: The One Thing Your CI/CD Workflow is Missing. https://www.overops.com/blog/how-to-measure-the-reliability-of-your-software-throughout-the-cicd-workflow/. Accessed: 2021-07-04.