Raising developers' awareness of source code energy consumption

Maquoi Jérôme
Faculty of Computer Sciences, NADI
University of Namur
Namur, Belgium
jerome.maquoi@unamur.be

***Abstract*—In the recent years, sustainability concerns have emerged in the software engineering community. However, few researches are focused on the link between the evolution of a source code and energy consumption. As a consequence, most developers don't have knowledge on the energy consumption of their software during its development. To address this problem, we want to build a theory that binds the evolution of a source code to its energy consumption profile. This relies on the measurement of the energy consumption evolution of multiple projects using repository mining approaches, the analysis of the reasons of this evolution, the design of proper data visualization to make this information available for the developers and the evaluation of these objectives through the development of a framework for developers. This theory and the framework associated may help developers to have better knowledge on the energy consumption evolution of their softwares and to have tools to reduce its consumption.**

I. Introduction

Nowadays, the climate and environmental crisis become more and more significant [1]. There is an urgent need for decreasing the carbon footprint of multiple areas of the society. One of these areas is Information and Communication Technology (ICT), that is responsible for about 10% of the global electricity consumption [2]. This number is expected to grow in the comming years as ICT takes more and more place in our daily lives. Reducing the impact of ICT on the environement and especially its energy consumption is ever more important to reach a sustainable society.

In the recent years, the software engineering community has begun to consider sustainability in software engineering as a research field [3]. Software testing can play a role on that matter, as it can be used to track non-functional properties of software, e.g. energy consumption, through two approaches: sustainable testing and testing for sustainability.

Sustainable testing denotes the collection of approaches dedicated to making software testing itself more sustainable, e.g. by decreasing its energy consumption. Among these approaches, there is the minimization of test suites whose objective is to reduce the number of test cases while preserving a certain level of quality assurance [4]. There is also the comparison of similar testing frameworks that seeks to find the framework which has the lowest energy consumption [5]. This thesis will not talk about sustainable testing approaches and will rather focus on testing for sustainability approaches.

Testing for sustainability denotes the collection of approaches dedicated to the production of sustainable software. The majority of the approaches discuss static and dynamic analysis of source code. Among these approaches, there is the measurements or estimation of the energy consumption of a software during its execution [6]–[8]. There is also the creation of a test generation framework that detects energy hotspots in a software [9]. Another approach is the comparison of different versions of source code in term of energy consumption, like for the GreenAdvisor project [10].

The purpose of this thesis is to create a theory that binds the evolution of a source code to its energy consumption profile. We will measure the energy consumption of multiple projects with software testing techniques and analyse their evolution using repository mining approaches. Using those data, we will seek how to explain this evolution. Finally, we will devise proper data visualization to make this information available for the developers. We will evaluate these objectives with the development of a framework bound for developers, throughout the research.

II. Background and related work

Sustainable software engineering is a new research field that denotes the creation of softwares that answer our needs without compromising the capacity to answer to our future needs [11]. Software testing can play a role in this research field because it can track functional as well as non-functional properties like energy consumption. It can be divided in two approaches: sustainable testing and testing for sustainability. Sustainable testing focuses on the way to reduce the impact of software testing techniques. It contains especially test-suite minimization techniques and the comparison of the energy consumption of similar software testing frameworks. This thesis will not focus on these approaches.

Testing for sustainability focuses on the approaches used to produce sustainable software. It mainly discusses source code analysis and especially static and dynamic analysis. For example, EcoCode is a static analysis plugin for sonarQube that detects and highlights energy code smells based on a catalogue of Android energy smells [12].

The estimation of energy consumption is the bigger subfield of dynamic analysis. Many tools estimate or measure the energy consumption of softwares, especially for Android apps.

For example, eLens is a tool that estimates the energy consumption of Android apps with different levels of granularity such as application, method, class and code line [7]. Another example is GreenScaler, a model created to help Android developers to estimate the energy consumption of their application without the need of physical monitoring through a power meter [8]. It is based on a machine learning algorithm that trained on hundreds of applications for which test cases are generated with random inputs. The followed heuristic for the generation of the test cases is the CPU-utilization of the application. Finally, GreenAdvisor is a tool that compares different versions of an Android application through its system calls to analyse the evolution of its energy consumption [10]. If the consumption increases, GreenAdvisor will identify the energy hotspots that caused this modification.

The comparison of different versions of a software has some known issues like the regression faults that may be introduced in a newer version of a project. A regression fault is the result of a software modification that makes this software regressing. Regression testing is an approach that seeks to insure that regression faults will not be introduced in a new version of a software by comparing this new version to the previous ones [13]. The regression faults can be functional or non-functional. Regression testing related to energy consumption is an underexplored research field. It could be interesting to use these testing techniques with energy consumption as guide. However, an obstacle can emerge. It may be difficult to compare similar execution traces of a project as it evolves. This issue is well known in transfer learning techniques. Transfer learning is a machine learning technique whose objective is to transfer knowledge across versions of a software to accurately model variability of systems [14]. It is difficult to apply transfer learning techniques on a permanently evolving project, especially if this evolution produces modifications of existing source code or architecture. Some researchers proposed a new model that tackles this issue [15].

Another way of comparing different versions of a software is through the continuous integration with commits analysis. Indeed, commits analysis can reveal lots of information on the development of a software. For instance, researchers presented a study explaining how the developers employ to save energy during the development of their software by analysing a set of commits [16]. They identified a dozen themes concerning energy-saving commits. Another example is the Coming tool, that can analyse a Git repository to find change patterns defined by the user between multiple versions of a software [17]. A change pattern is defined as a set of modifications (e.g., insert or remove) performed on fragment of source code.

Another subfield of dynamic analysis is the creation of test generation frameworks. Researchers have created a framework that generates test cases to detect energy hotspots and energy bugs from Android applications through test inputs likely to capture energy hotspots and energy bugs [9]. Energy hotspots denote scenario where the execution of a smartphone application makes the energy consumption of the smartphone abnormally high. Energy bugs denote scenario where a smartphone application that has abnormal functioning prevents the smartphone from becoming idle after the execution of the application.

Android application is the biggest scope of the papers talking about sustainable software testing but there are also papers talking about other scopes. For example, JoularJX is a tool that estimates the energy consumption of methods from Java applications, based on measures from hardware components such as the CPU [6].

As we can see, the main subfield of dynamic analysis of source code is the estimation of the energy consumption of a software. Regression testing techniques are mainly oriented towards the detection of functional regression faults and less oriented towards non-functional faults as the energy consumption. It could be relevant to analyse a software through regression testing and transfer learning techniques in light of energy consumption faults. The analysis of different types of commits and their impact on the consumption of the software is a relevant angle to understand the evolution of its energy consumption and is underexplored in the literature. Java is one of the most used programming languages in industry with C and C++ [18]. Developers can use this language for web, cloud computing as well as desktop applications. Thanks to the Java Virtual Machine, Java projects can run on multiple platforms, which makes it easy to maintain and to code with. Regression testing could be used to analyse Java projects in a context of continuous integration with commit analysis to understand the inner working of energy consumption and its evolution in a software development. It could also be useful to think of ways of making developers aware of their software energy consumption.

III. Project description

A. Objectives

1) Measure the evolution of the energy consumption of a software: Choosing a method to measure the energy consumption of a software at a precise moment and applying it to a benchmark of multiple softwares from git repositories. Then, we will use this method to measure the evolution of the consumption of these software through their different versions. We need to understand the energy consumption evolution of these softwares as a whole.

2) Seek the reasons of the energy consumption evolution: Based on the results of the energy consumption measures gathered from the git repositories, we will explore the reasons of the energy consumption evolution of the softwares through commit analysis. We will identify different types of commit and bind them to their energy consumption to understand which kind of commit have impacts on the evolution.

3) Make the information available for the developer: The objective is to make the information processed in the two points above go back up to the developers so that he can more easily change what needs to be changed in his code to make his software more energy efficient. We will search for the different ways to do it and think of the better for our use case.

4) Evaluation through the development of a framework: Throughout the objectives mentioned above, a framework will be developed to test and evaluate the ideas discussed in the other objectives during the thesis.

B. Work plan

1) First year:

- During the first steps of this thesis, I will review the literature on energy consumption measures to begin working on energy measurement of a software at a given time.
- Afterwards, I will start digging in Git repositories to find multiple projects to compare their energy consumption.
- Then, I will measure and analyse the energy consumption of projects at a given time.

2) Second year:

- I will use regression testing techniques to measure and analyse the evolution of the energy consumption of the found projects.
- I will think of a way to develop the framework headed for the developers with the gathered information and I will start developing it.

Publication attempt on the evolution of energy consumption of the selected projects.

3) Third year:

- I will think of a way to make the evolution of energy consumption of a project available for the developer.
- I will add the evolution of energy consumption to the framework.
- I will make an evaluation of the framework.

Publication attempt on the way to make the evolution of energy consumption available for the developer.

4) Fourth year:

- I will review the literature on the ways to compare Git commits to identify commit patterns.
- I will identify commit patterns in the selected projects and compare them to their evolution of the energy consumption.

Publication attempt on the evolution of energy consumption of a project in function of its commits.

5) Fifth year:

- I will identify a way to make the comparison of commits in function of their energy consumption available to the developers.
- I will add the comparison of commits in the framework.

6) Sixth year:

- I will start to compile my thesis.
- I will evaluate the developed framework.

Publication attempt of the framework.

References

[1] H.-O. Pörtner, D. C. Roberts, E. Poloczanska, *et al.*, "Ipcc, 2022: Summary for policymakers," 2022.

[2] R. Verdecchia, P. Lago, C. Ebert, and C. De Vries, "Green it and green software," *IEEE Software*, vol. 38, no. 6, pp. 7–15, 2021.

[3] B. Penzenstadler, V. Bauer, C. Calero, and X. Franch, "Sustainability in software engineering: A systematic literature review," 2012.

[4] S. U. R. Khan, S. P. Lee, R. W. Ahmad, A. Akhunzada, and V. Chang, "A survey on test suite reduction frameworks and tools," *International Journal of Information Management*, vol. 36, no. 6, pp. 963–975, 2016.

[5] L. Cruz and R. Abreu, "On the energy footprint of mobile testing frameworks," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2260–2271, 2019.

[6] A. Noureddine, "Powerjoular and joularjx: Multi-platform software power monitoring tools," in *2022 18th International Conference on Intelligent Environments (IE)*, IEEE, 2022, pp. 1–4.

[7] S. Hao, D. Li, W. G. Halfond, and R. Govindan, "Estimating mobile application energy consumption using program analysis," in *2013 35th international conference on software engineering (ICSE)*, IEEE, 2013, pp. 92–101.

[8] S. Chowdhury, S. Borle, S. Romansky, and A. Hindle, "Greenscaler: Training software energy models with automatic test generation," *Empirical Software Engineering*, vol. 24, pp. 1649–1692, 2019.

[9] A. Banerjee, L. K. Chong, S. Chattopadhyay, and A. Roychoudhury, "Detecting energy bugs and hotspots in mobile apps," in *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, 2014, pp. 588–598.

[10] K. Aggarwal, A. Hindle, and E. Stroulia, "Greenadvisor: A tool for analyzing the impact of software evolution on energy consumption," in *2015 IEEE international conference on software maintenance and evolution (ICSME)*, IEEE, 2015, pp. 311–320.

[11] S. McGuire, E. Shultz, B. Ayoola, and P. Ralph, "Sustainability is stratified: Toward a better theory of sustainable software engineering," *arXiv preprint arXiv:2301.11129*, 2023.

[12] O. Le Goaer and J. Hertout, "Ecocode: A sonarqube plugin to remove energy smells from android projects," in *37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–4.

[13] A. Shi, P. Zhao, and D. Marinov, "Understanding and improving regression test selection in continuous integration," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, 2019, pp. 228–238.

[14] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[15] H. Martin, M. Acher, J. A. Pereira, L. Lesoil, J.-M. Jézéquel, and D. E. Khelladi, "Transfer learning across variants and versions: The case of linux kernel size," *IEEE Transactions on Software Engineering*, vol. 48, no. 11, pp. 4274–4290, 2021.

[16] I. Moura, G. Pinto, F. Ebert, and F. Castor, "Mining energy-aware commits," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, IEEE, 2015, pp. 56–67.

[17] M. Martinez and M. Monperrus, "Coming: A tool for mining change pattern instances from git commits," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, IEEE, 2019, pp. 79–82.

[18] L. B. A. Rabai, B. Cohen, and A. Mili, "Programming language use in us academia and industry," *Informatics in Education*, vol. 14, no. 2, p. 143, 2015.