The organizational hurdles of structurally reducing the energy consumption of software

Roy van der Steen[1], Bernard van Gastel[1]

[1]Radboud University, Toernooiveld 212, 6525 EC, Nijmegen, The Netherlands

Abstract

Many studies discuss the high energy consumption of software, most of which attempt to measure it or focus on what software developers can do to reduce it. In contrast, this paper approaches the problem of energy-inefficient software from an organizational standpoint. It aims to discover why organizations do not structurally work on reducing the energy consumption of software. This study portrays a survey with 19 participants, a literature study, and five interviews. It points out several obstacles hampering organizations, such as the difficulty of measuring Software Energy Consumption (SEC), the high costs of reducing SEC, a lack of incentives for customers, and a lack of information dissemination. It furthermore describes potential solutions to resolve these obstructions, like improving information delivery to practitioners, using software-based measurement tools, more transparency in the pricing of cloud providers, creating a good business case, and standardization.

Keywords

software energy consumption, green software, sustainable software, organizational hurdles

1. Introduction

Within the last decade, interest in the energy consumption of software has been rising among practitioners [1] and researchers alike [2, 3, 4]. However, the number of practitioners adopting published research remains limited [5, 6]. Most recent studies focus on the technical aspect of energy consumption (i.e., code optimization & energy measurement) and only few on the development process [3]. Additionally, the target group of most field research is software developers [7, 1, 6]. It thus seems likely that there is ample technical knowledge available for Software Producing Organizations (SPOs) to start reducing Software Energy Consumption (SEC). Therefore, this paper hypothesises that the scarcity of effort on reducing SEC is not caused by software developers lacking technical ability, but rather by other parties involved and a lack of accessibility to comprehensible information.

To discover what causes the discrepancy between what is researched and what is used in practice, this paper formulates the research question: *Why are organizations not modifying their software development process to structurally work on reducing the energy consumption of software?* This question was used to research what SPOs require to reduce their SEC together with the following sub-questions: (1) *What are the obstacles that software developing organizations face when trying to reduce the energy consumption of their software?*; (2) *What can be done to remove*

Benevol'23: The 22nd Belgium-Netherlands Software Evolution Workshop, November 27–28, 2023, Nijmegen
roy.vandersteen@ru.nl (R. v. d. Steen); bernard.vangastel@ru.nl (B. v. Gastel)
https://www.sustainablesoftware.info/ (B. v. Gastel)

the obstacles that software-developing organizations face when attempting to reduce the energy consumption of their software? The research is limited to Dutch organizations.

This paper answers these questions using qualitative exploration, and shows focus points for future efforts to reduce software energy consumption. It adopts a pragmatic, qualitative research approach, and uses a combination research design consisting of an exploratory part and explanatory part. In both parts, data was gathered from participants by means of a survey and interviews respectively. This data was analyzed using thematic analysis with manual coding.

2. Methodology

This paper describes the approach that has three consecutive steps. All quotes and references to data are translated and anonymized. All survey participants are quoted under the pseudonyms P01 to P19, and all interviewees are quoted under the pseudonyms I1 to I5.

2.1. Survey

The first phase consists of an exploratory survey with 17 open-ended questions. The survey has three objectives. First, to discover obstacles and discern what prevents practitioners from doing more to reduce SEC. Second, to find out what organizations undertake to structurally work on reducing SEC and how they internally communicate about SEC. Third, to find potential candidates for the interviews. The survey also gathered participant and organizational demographics to contextualize the participant's answers. The survey was distributed on LinkedIn to find non-developer IT personnel with influence on / knowledge of the development process. In total, this study uses the results of 19 survey participants.

Initially, all survey results were read through carefully and inductively open-coded. A second open-coding session used deductive codes based on the now-established data familiarity as well as new inductive codes. Only the second open-coding session was used for further coding. Attribute codes were used for background/demographic information and descriptive codes for the remaining data. Axial coding was used to establish connections between the codes, and finally, selective coding to identify core ideas present within the survey results.

2.2. Literature Study

The second phase consists of a literature study with two objectives. First, to ascertain if the obstacles mentioned by the survey participants indeed are a cause of hinder or that there might be more to the problem. Second, to increase validity of the obstacles by confirming their occurrence in other studies. An exploratory search was conducted to identify relevant literature, including other surveys and interviews. Using the Google Scholar search engine, the following terms were searched: *"software energy consumption","software energy consumption survey","software energy consumption interview","measure software energy consumption","hardware to measure software energy consumption","software energy consumption awareness","software energy consumption user","sustainable software customer","sustainable software energy profiling","profiling software energy efficiency costs time"*[1].

[1]All search terms that include "energy" were also searched with "power" instead

Google, Google Scholar, GitHub, and GitLab were searched for software-based tools to measuring SEC. For Google Scholar, the previously mentioned terms were used. For GitHub and GitLab, the following terms were searched: *'software energy consumption', 'measure energy consumption', and 'code energy consumption'*[1]. Additionally, Google, GitHub, and GitLab were also searched for the tools mentioned in papers found with Google Scholar. To be included in this study, a tool: 1) must have development activity after 2015[2] 2) must not be officially deprecated/abandoned 3) must have at least a 'beta' version available online 4) must be able to run on Linux 5) must have an open-source repository.

2.3. Interviews

During the third phase, in-depth, semi-structured interviews were held with willing survey participants. The interviews are explanatory, based on an interview guide [9], and have two objectives. First, to obtain more details on the obstacles found by the survey. Second, to ascertain the viability of potential solutions devised from the survey and literature study and possible solutions from the interviewees. The interview questions are based on the survey results and questions from previous research[7, 6].

There are five interviewees, each of whom participated in a single interview with only one interviewer. The interviews were audio recorded on an encrypted drive, with participant permission, and lasted between 45 and 75 minutes. The interviews took place online, using the Microsoft Teams environment of Radboud University. To further inform them about the study, interviewees were sent additional information via email, which included information about the purpose of the study, the location, duration, and interview procedures, as well as information on the processing and storing of interview data. At the start of each interview, before recording, the interviewee was asked for their consent for the recording of audio and the other procedures mentioned in the information email. The recordings were manually transcribed and inductively open-coded. Codes from the survey were not transported to the interviews, to prevent results from the survey potentially affecting the analysis of the interviews. Then axial coding was used to establish connections between the codes, and finally, selective coding to identify core ideas within the interview results.

3. Results

The participants were all male, mostly aged 30-39 (8) and 40-49 (8). Their tasks were: management, strategy development, architecture, programming, meetings, and customer interaction.

3.1. The state of SEC within organizations

To better understand the current position of organizations, the participants were asked how they have tried to reduce SEC, the actions their organizations take and the communication about SEC within their organizations. One notable finding is that seven participants spoke of performance optimization as a means to reducing SEC. When asked if they had conducted

[2]The year 2015 was chosen for the development activity requirement because Jagroep et al. tested available tooling in this year to be too inaccurate for proper measurements [8].

energy measurements, P02 mentioned “Monitor tools for unused server/data capacity.” and P04 “IDE Performance Profilers.” In a related question, P17 said to “detect inefficient code with heat-maps” and P01 simply mentioned “Performance optimization” as a means to reduce SEC.

Communication about SEC, and sustainable IT in general, remains limited. Only four participants mentioned any form of communication about these topics within their organization. P02 said there is “internal communication” and “workshops” to inform their developers. P03 vaguely answered “knowledge sharing” whilst P13 specified to “explain choices to interns to create awareness” and that they “work in small teams with constant consultation and collaboration.” Five participants showed signs that *something* is structurally done by their organization, but in only two cases concrete examples were given. P18 mentioned that they choose to use “energy efficient datacenters, running on renewable energy” to deploy their software. Two other participants, likely from the same organization, both said their organization guides its decisions with a book that is developed in-house. “This book expands upon ISO 25010 with sustainability as a separate quality attribute, which can be used as a basis for non-functional requirements.”

3.2. Obstacles hindering organizations

3.2.1. A lack of information dissemination

The survey participants and the interviewees mention that customers, organizations, and developers lack awareness. However, this may also be interpreted as a symptom of a lack of information dissemination; because there is not enough information brought to the attention of IT practitioners, not enough awareness has been created. There are multiple signs of a lack of information dissemination. For example, three interviewees mentioned that the education landscape does not teach students about the sustainability of IT. I3, who frequently works with interns, explained that “students are almost completely ignorant.”

3.2.2. Measuring SEC is vital yet difficult

The analyzed survey results report that measurements may help verify supplier sustainability claims, provide a starting point to reduce SEC and ascertain the impact of software changes on SEC. The interviewees add that measurements are also helpful in creating more awareness, comparing software products, improving communication about SEC, and helping ascertain the economic viability of reducing SEC.

However, participants found it challenging to measure SEC. I4 stated that it demands skills that most software developers lack, while I3 said that the complexity of IT devices is a more significant problem. Interestingly, these difficulties indirectly hint that the interviewees were unaware of currently available measurement tools such as PowerJoular, Scaphandre, and codecarbon. Additionally, I4 said that “There are standard solutions from Azure and AWS to provide insight into the power consumption of an instance.” Further study reveals that the three largest cloud providers (Google, Azure, and AWS[3]) provide insights in energy use and estimated CO2 emissions[4]. Possibly, a lack of information dissemination is (partly) to blame for the difficulty developers have measuring SEC.

[3]According to Statista

[4]AWS, Azure, and Google have published information about their respective dashboards

3.2.3. Reducing SEC lacks a business case

According to the survey results, reducing SEC is expensive because it costs substantial development time. Moreover, I1 explained that it also increases maintenance costs because "optimizing software will likely mean adding more code." and "When looking at textbook examples of code optimizations, the resulting code is quite unreadable. This makes the code more sensitive to maintenance, and you will always need a very experienced developer to maintain it." Additionally, I5 claimed that "To switch to better-optimized alternatives, more training is necessary. Which can be a huge investment for an organization." Opposingly, I3 said it does not have to be expensive: "I find it shortsighted when people say it takes too much development time. I would say it only takes a few hours of thinking on an architectural level." Other interviewees also mentioned that it is better to start with architectural changes. According to I5, the costs do not matter much as long as the benefits outweigh them: "You're still talking about costs. Even at minimal costs, I think it will be very low on the priority list of companies." I1 claimed something similar: "I expect much more from a business case."

3.2.4. The Cloud

One theme that did not surface from the survey results, but is present within the interviews, is the influencing role of cloud providers. The performance gained by optimizing code for SEC can be easier and cheaper achieved with cloud up-scaling. The resulting issue was most clearly explained by I1: "How do those costs weigh up against a bit more energy consumption? Or, in this case, cloud costs? The cheaper the cloud costs are, the faster that comparison results in scaling up." In addition, I2 explained that up-scaling is effortless, and the ecological impact is often left unnoticed. Several answers also signify that the developers using the instances are not getting insights on costs or electricity usage and thus remain unaware of their influence.

3.2.5. Customers lack incentive

Several survey participants mention a lack of time/priority as an obstacle, which may be caused by a lack of incentives for organizations. However, in most cases, customers determine task priority. Sadly, there is also little to no incentive for these customers. P11 explained that "Customers are not/hardly interested. All they want is that the software provides value and is maintainable with in-house knowledge." To incentivize these customers, P16 said that more insights are needed: "Customers have little insight into the emissions, making it difficult to show the effects of energy saving measures." Moreover, cultivating demand for sustainable software among customers can lead to organizations being incentivized. When enough customers desire sustainable software, organizations will want to meet new market demand.

3.3. Solutions to the obstacles

3.3.1. Spreading information

The information available on SEC is not reaching IT practitioners. Sharing guidelines, handbooks, and tools will help organizations with policy making. Furthermore, cloud providers should clearly state the proportion of their pricing attributed to electricity consumption. This improved clarity will increase awareness of software electricity consumption and associated CO2 emissions. Cloud dashboards showing costs and electricity usage should be available to an organization's developers (and thus not only the organization's administration) so they realize the effects of their behavior. To better inform developers, current available tools should be more actively promoted. Lastly, educational institutions should incorporate IT sustainability. For example, computing science studies can teach students how to keep their code efficient.

3.3.2. Energy measurements

Developers struggle to measure SEC but argue for its necessity. Fortunately, existing tools can assist with this issue, but most developers are unaware of them. In addition, the interviewees said a lot can be done without measurements. For example, I2 said "As a developer and as an organization you can definitely reduce SEC without measuring. You can get a long way with common sense." Steps such as turning off unnecessarily running cloud instances or reducing file sizes are examples given by the interviewees. Considering SEC at the architectural level does not require energy measurement and is argued to have a greater impact than code changes.

3.3.3. A good business case

Minimizing costs is not a good solution on its own. Instead, it may be better to look at business cases. As I1 put it: "There is not much use to it without a business case." To clarify, a business case is "a justification for a proposed project or undertaking based on its expected commercial benefit" [10]. To help organizations create business cases for reducing SEC, this paper believes that it is important to (1) Decrease investment costs. (2) Increase potential commercial benefits. (3) Develop tooling to more accurately estimate the potential commercial benefit.

3.3.4. Standardization and auditing

I4 expressed a need for a guideline, "something like WCAG." I1 also expressed the same need: "That can be done perfectly fine with such an ISO standard. But then you'll have to add very concrete numbers so that you communicate about what you're actually being audited for." Standardizing methods can help organizations get started and create the opportunity to certify organizations on the sustainability of their IT. As said by I3, "It is the responsibility of external organizations to check others and verify that what they say is true."

4. Related Works

In their 2015 study, Pang et al. [6] conducted a survey with 122 participants and found that programmers had limited knowledge of SEC. Their study and others indicate that any form of SEC quantification will likely drive an increased awareness among people [6, 11]. Chowdhury et al. summarize that "the measurement of software energy consumption is expensive in terms of hardware and difficult in terms of expertise." There also exists methods to predict SEC using a hardware model, such as in [2, 3, 4]. These methods do not need measurement set-ups, but the results might be more difficult to interpret than measurement data for one specific run of a program, that is similar to what existing profilers produce. [12] on the other hand, Jagroep et al. say that software-based profilers can not be used to estimate SEC yet, but can be used to get a sense of the energy consumption habits of software." [8] The surveys and interviews from the studies by Ournani et al. [7] and Manotas et al. [13] show that practitioners give low priority to SEC. A 2021 study by Cico et al. shows that customers do not value environmental sustainability much in their software projects either [14].

5. Threats & validity

The number of survey and interview participants was low. Surveys and interviews from other papers were cross-checked to reduce the effect this has on the paper's validity. However, its generalizability remains affected. Furthermore, the participants were approached through LinkedIn. This could create a snowball process of recruiting more participants by followers of followers, potentially causing community bias. The codes also have a potential bias; despite not transferring codes from survey to interviews, all coding was done by the same individual.

6. Conclusion

This paper aims to discover why Dutch organizations are not modifying their software development process to structurally work on reducing the energy consumption of software. To achieve this goal, data on Dutch organizations, their efforts to reduce SEC, and related problems were gathered using 19 survey completions and five interviews. This paper shows that the organizations of the participants lack a structured approach to reducing SEC. The obstacles identified are the difficulty of measuring SEC, the high costs of reducing SEC, a lack of customer incentives, and a lack of information dissemination.

Potential solutions were derived from the survey responses, literature analysis, and interviews. The primary solution is to bring currently available information to the attention of practitioners. In addition, developers can use software tools to measure SEC. Despite their lower accuracy, they should suffice in most cases. Furthermore, a good business case is required for organizations to take action. Hence, investment costs should be minimized and the commercial benefit maximized. Additional tooling to calculate expected costs and savings may also help. Cloud providers need to become more transparent about how much of their pricing consists of electricity costs, to increase awareness among their customers. Lastly, standardization and auditing might increase the commercial potential of SEC by providing proof of software sustainability, and help organizations get started on reducing SEC. If successful, these solutions will likely lead to SEC requirements in software projects, affecting the field of software evolution.

References

[1] G. Pinto, F. Castor, Y. D. Liu, Mining questions about software energy consumption, in: Proceedings of the 11th Working Conference on Mining Software Repositories, 2014, pp. 22–31.

[2] B. van Gastel, R. Kersten, M. van Eekelen, Using dependent types to define energy augmented semantics of programs, in: M. C. J. D. van Eekelen, U. D. Lago (Eds.), Foundational and Practical Aspects of Resource Analysis - 4th International Workshop, FOPARA 2015, London, UK, April 11, 2015, Revised Selected Papers, volume 9964 of *Lecture Notes in Computer Science*, 2015, pp. 20–39. doi:10.1007/978-3-319-46559-3_2.

[3] S. Ergasheva, I. Khomyakov, A. Kruglov, G. Succil, Metrics of energy consumption in software systems: a systematic literature review, in: IOP Conference Series: Earth and Environmental Science, volume 431, IOP Publishing, 2020, p. 012051.

[4] B. van Gastel, Assessing sustainability of software - Analysing Correctness, Memory and Energy Consumption, Ph.D. thesis, Open University, 2016. URL: https://www.sustainablesoftware.info/docs/thesis-met-cover.pdf.

[5] E. Jagroep, G. Procaccianti, J. M. van der Werf, S. Brinkkemper, L. Blom, R. van Vliet, et al., Energy efficiency on the product roadmap: an empirical study across releases of a software product, Journal of Software: Evolution and process 29 (2017) e1852.

[6] C. Pang, A. Hindle, B. Adams, A. E. Hassan, What do programmers know about software energy consumption?, IEEE Software 33 (2015) 83–89.

[7] Z. Ournani, R. Rouvoy, P. Rust, J. Penhoat, On reducing the energy consumption of software: From hurdles to requirements, in: Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2020, pp. 1–12.

[8] E. Jagroep, J. M. E. van der Werf, S. Jansen, M. Ferreira, J. Visser, Profiling energy profilers, in: Proceedings of the 30th annual ACM symposium on applied computing, 2015, pp. 2198–2203.

[9] S. E. Hove, B. Anda, Experiences from conducting semi-structured interviews in empirical software engineering research, in: 11th IEEE International Software Metrics Symposium (METRICS'05), IEEE, 2005, pp. 10–pp.

[10] O. Languages, The Oxford English Dictionary, 2023. URL: https://www.oed.com/.

[11] E. Jagroep, J. Broekman, J. M. E. Van Der Werf, P. Lago, S. Brinkkemper, R. Blom, Leen anfd Van Vliet, Awakening awareness on energy consumption in software engineering, in: 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Society Track (ICSE-SEIS), IEEE, 2017, pp. 76–85.

[12] S. A. Chowdhury, A. Hindle, Greenoracle: Estimating software energy consumption with energy measurement corpora, in: Proceedings of the 13th international conference on mining software repositories, 2016, pp. 49–60.

[13] I. Manotas, C. Bird, R. Zhang, D. Shepherd, C. Jaspan, C. Sadowski, L. Pollock, J. Clause, An empirical study of practitioners' perspectives on green software engineering, in: Proceedings of the 38th international conference on software engineering, 2016, pp. 237–248.

[14] O. Cico, L. Jaccheri, A. N. Duc, Software sustainability in customer-driven courses, in: 2021 IEEE/ACM International Workshop on Body of Knowledge for Software Sustainability (BoKSS), IEEE, 2021, pp. 15–22.