Exploring Developer Views on Software Carbon Footprint and its Potential for Automated Reduction

Haozhou Lyu
Chalmers | University of Gothenburg
Gothenburg, Sweden
haozhou@student.chalmers.se

Gregory Gay
Chalmers | University of Gothenburg
Gothenburg, Sweden
greg@greggay.com

Maiko Sakamoto
University of Tokyo
Tokyo, Japan
m-sakamoto@k.u-tokyo.ac.jp

ABSTRACT

Reducing software carbon footprint could contribute to efforts to avert climate change. Past research indicates that developers lack knowledge on energy consumption and carbon footprint, and existing reduction guidelines are difficult to apply. Therefore, we propose that automated reduction methods should be explored. However, such tools must be *voluntarily adopted and regularly used* to have an impact.

In this study, we have conducted interviews and a survey (a) to explore developers' existing opinions, knowledge, and practices with regard to carbon footprint and energy consumption, and (b), to identify the requirements that automated reduction tools must meet to ensure adoption. Our findings offer a foundation for future research on practices, guidelines, and automated tools that address software carbon footprint.

CCS CONCEPTS

• **Software and its engineering → Search-based software engineering**; **Software development techniques**; **Extra-functional properties**.

KEYWORDS

Carbon Footprint, Energy Consumption, Sustainability, Genetic Improvement, Genetic Programming

ACM Reference Format:
Haozhou Lyu, Gregory Gay, and Maiko Sakamoto. 2023. Exploring Developer Views on Software Carbon Footprint and its Potential for Automated Reduction. In *GECCO '23, The Genetic and Evolutionary Computation Conference, July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recent reports on climate change paint a stark picture: "Climate change creates new risks and exacerbates existing vulnerabilities... presenting growing challenges to human health and safety, quality of life, and the rate of economic growth." [22]. The carbon dioxide emitted through development and use of software may contribute to climate change. In 2015, data centers accounted for an estimated 3.00% of global energy consumption—double that of the United Kingdom and matching the airline industry [3]. Training a single neural network—the core of modern Machine Learning—can emit as much carbon as the entire lifetime of five cars [11].

That carbon footprint *must be reduced*, but this is not a straightforward task. There are sources of emissions at multiple stages of development, produced through development and use of software [26]. Further, while carbon footprint is largely a product of energy consumption, the *quantity*, *sources*, and *location* of that consumption are all important.

Researchers have begun to make recommendations on how to develop software with a reduced carbon footprint (e.g., [15, 23, 27, 33]). Such guidelines are highly important, but can be difficult to apply in practice—especially after the code has been written. Further, it is not clear that developers have a clear understanding of carbon footprint, energy consumption, or how either can be reduced during development [18–20].

Therefore, we are interested in exploring *automated* reduction of carbon footprint through transformation of existing source code. Such transformations should preserve the semantic meaning of the code while reducing the carbon footprint by, e.g., reducing energy consumption or controlling where energy is consumed. A promising technique to impose such changes is genetic programming [13], where populations of program patches are scored according to qualities of interest, then evolved over many generations to maximize or minimize these scores. Genetic programming has been applied to reduce energy consumption [4, 8, 15, 16, 30, 31]. We propose that such approaches could be extended or new approaches could be developed specifically to target carbon footprint.

The development of automated carbon footprint reduction could impact the environmental sustainability of the IT industry. However, this can only be the case if such tools are *voluntarily adopted and regularly used by developers*. Such tools must be effective, trustworthy, easy to use, and fit within existing developer workflows with minimal friction.

The goals of this study are (a) to explore developers' existing opinions, knowledge, and practices with regard to carbon footprint and energy consumption, and (b), to identify the requirements and constraints that automated carbon footprint reduction tools must meet to ensure voluntary adoption. To meet these goals, we have conducted interviews with software developers in various domains and of varying experience. Thematic analysis was then performed to develop a survey for wider distribution.

Regarding knowledge, opinions, and practices:

- Many developers lack knowledge on carbon footprint or energy consumption. However, some respondents had basic

GECCO '23, July 15–19, 2023, Lisbon, Portugal
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

understanding of factors affecting energy consumption, and developers gain knowledge of these concepts over time.

- A plurality of respondents are neutral on whether software contributes to climate change and whether carbon footprint should be controlled. However, a greater proportion agree than disagree with both. The majority of participants feel that both development organizations and regulatory agencies bare responsibility for controlling software energy consumption or carbon footprint.
- In current practice, energy is considered most often during design and implementation, and is reduced by improving performance or reducing resource consumption. Carbon footprint is considered most often during deployment and design—in relation to energy consumed both by the software and during development. Energy consumption and carbon footprint are most commonly measured through CPU or memory usage. Energy consumption is often evaluated using manual or exploratory testing, carbon footprint through measurements during Continuous Integration (CI).

Regarding the requirements of an automated reduction tool:

- 70.00% of participants are willing to try a tool. However, 35.00% were skeptical of its capabilities and 50.00% were unlikely to trust the tool. To ensure trust, reductions in carbon footprint must not compromise security, correctness, or other important qualities. Developers would also like it to be easy to integrate into a CI pipeline, be reasonably priced, and complete changes quickly.
- Developers would apply the tool as part of CI, generally on a regular basis (e.g., daily—possibly at non-peak times).
- To encourage adoption, the tool should have a record of transparent and trustworthy results. It should be easy to install and use in existing development workflows, with sufficient guidance. It should be open-source, so it can be understood or improved.

This study provides a foundation for future research by exploring requirements for automated carbon footprint reduction tools. We also offer insights to those interested in the existing opinions, knowledge, and practices of developers. To help enable future research, we also make our interview and survey responses available [2].

2 BACKGROUND

Carbon Footprint and Energy Consumption of Software: Carbon footprint is the total quantity of carbon dioxide emissions associated with an individual, product, or organization's activities. This can include direct emissions (e.g., fossil fuels consumed during production) or indirect emissions (e.g., energy consumption) [32].

The carbon footprint of a product can be assessed at multiple stages or activities in a product's lifecycle (i.e., from material acquisition, to production and use, to waste management) [10]. Typically, assessment includes setting activity scope and then data collection [29]. The scope can incorporate direct emissions, emissions in purchased energy, or indirect emissions. Data collection can be done through direct measurement or estimation based on emission factors and models that account for fuel and energy consumption and other inputs that contribute to emissions.

Software has been found to be a major source of carbon emissions [21]. There are sources of both direct and indirect emission at multiple phases of development, including implementation, testing, delivery, usage, and maintenance [26]. In this research, our scope is primarily restricted to indirect emissions associated with energy consumption during software *usage*—i.e., when interactions take place with the software [6].

Within this scope, carbon footprint is affected both by the *quantity* of energy consumed and *where and how* that energy is produced or consumed, as some energy sources have a greater carbon footprint than others. Calculating and reducing software carbon footprint is not simple, as it is affected by energy usage on both the client-side (i.e., on consumer devices) and server-side (i.e., at data centers in disparate geographic areas), as well as on network transmissions between the two [1]. Automated approaches must consider not just energy quantity, but also aspects such as the location of computing elements.

Genetic Improvement: Genetic improvement is the automated improvement of non-functional qualities (e.g., performance) of software through semantics-preserving transformations to the source code [4, 12]. Transformations are imposed using genetic programming [30]. In genetic programming, a population of patches are produced and then judged using one or more fitness functions related to the non-functional quality of interest. The patches that yield the best scores form a new population through mutation—where stochastic changes are introduced—or crossover—where aspects of "parent" patches combine to form new "children". Carbon footprint can be considered a quality, and genetic programming could potentially be used to automate carbon footprint reduction through appropriate program transformations and fitness functions.

3 RELATED WORK

Pang et al. found that most programmers are largely unaware of energy consumption, lack knowledge on the causes or how to measure consumption, and rarely address energy efficiency issues [19]. Pinto et al. also found that developers are aware of the importance of energy consumption, but have a limited and vague understanding of it [20]. Zakaria et al. found that most developers have incorrect or incomplete knowledge of energy consumption, and that some developers regarded green software design as a "threat" that could disrupt their workflow [18]. Our study yields similar findings on energy consumption, but extends our understanding with regard to developer opinions on carbon footprint, current practices regarding both carbon footprint and energy consumption, and opinions on automated improvement tools.

Past research has offered guidelines on how to reduce energy consumption and carbon footprint (e.g., algorithm selection [15], code structure [23, 33], considering server distribution and location [15], or controlling image quality [27]). Such guidelines are highly important, but are not always easy to apply. Nor is it simple to manually improve code after it has been written. Therefore, we are interested in automated carbon footprint reduction techniques.

To date, we are unaware of any automated tools targeting carbon footprint. However, there have been several approaches targeting energy consumption, mostly based on genetic programming [4, 8, 15, 16, 30, 31]. Other approaches include specialized

Table 1: Demographic information on interviewees, including location, position, job responsibilities (self-described), and development experience (years).

ID	Country	Position	Responsibility	Exp.
P1	Sweden	Manager	Overlook technical road maps	25
P2	Japan	Developer	Data analysis and development	5
P3	Sweden	Student	Developer testing televisions	4
P4	Sweden	Manager	Technical strategy and development process	20
P5	Japan	Developer	Develops software	5
P6	Japan	Developer	Network operation and maintenance tools	4
P7	Japan	Developer	Service planning and development	7
P8	Japan	Researcher	AI and robotics development	6
P9	Sweden	Student	Machine Learning development	4
P10	Sweden	Developer	C software development	4

compilers [24] and data migration strategies [7]. We hypothesize that genetic programming can also reduce carbon footprint, potentially by extending existing approaches to consider both client and server-side components and additional fitness functions. However, it is important that such tools meet the needs and constraints of developers to ensure voluntary adoption. The intent of our study is to identify those requirements.

4 METHODOLOGY

Our research is guided by the following questions:

RQ1: What knowledge do developers have about the carbon footprint or energy consumption of software?

RQ2: How do developers assess and control the carbon footprint or energy consumption of their software?

RQ3: What requirements and constraints must be satisfied for developers to trust carbon footprint reduction tools?

RQ4: How should a carbon footprint reduction tool fit into the development workflow?

RQ5: How can voluntary adoption of carbon footprint reduction tools be encouraged?

To answer these questions, we conducted semi-structured interviews, then performed thematic analysis following Cruzes and Dyba’s guidelines [5], to gain an initial understanding. Then, we distributed a survey—based on the interview results—to gain insights from a broader range of participants.

4.1 Interviews

Population Definition: Our population consists of participants with experience developing software, including professionals and university students studying a related discipline.

Sampling: We interviewed 10 participants. The sampling method was a mix of purposive and convenience sampling [9]. The professionals were gathered from companies in Sweden and Japan using the online platform LinkedIn, as well as through personal contacts. The students include one M.Sc. student and one Ph.D. student from Chalmers University of Technology. We stopped after 10 interviews, as we had achieved result saturation.

Demographics: Table 1 shows information on participants. To maintain confidentiality, we omit participants’ names and instead use IDs. These participants come from various roles, with experience ranging from 4–25 years, and experience in a variety of domains (e.g., robotics, machine learning, web applications).

Table 2: Explanation of themes (bold) and sub-themes.

Theme	Explanation
Energy Use	How software consumes energy.
PC	Energy use related to client-side actions.
Server	Energy use related to server-side actions.
Prior Knowledge	Interviewees’ prior knowledge of software energy consumption and carbon footprint.
Prior Experience	Interviewees’ prior experience related to energy consumption or carbon footprint.
Measurement Experience	Measurements of energy consumption or carbon footprint.
Reduction Experience	Experience in reducing energy consumption or carbon footprint of software.
Energy Responsibility	Whether developers should responsibility for reducing carbon footprint.
Individual	The steps developers should take regarding energy consumption reduction.
Organization	The steps organizations should take regarding energy consumption reduction.
Integration	Fitting automated reduction tools into the development workflow.
Use Frequency	The frequency and situations where interviewees would apply the tool during development.
Acceptance	Interviewees’ attitudes towards the adoption of automated reduction tools and how to use them.
Voluntariness	Attitudes towards voluntary or mandated use of automated reduction tools.
Required Constraints	Conditions that should be met for interviewees to adopt and trust automated reduction tools.

Interview Guide: The interview questions and their mapping to our research questions can be found in our replication package [2]. The questions were open-ended, so we could ask follow-up questions if needed, while ensuring we answered the core research questions. The 13 interview questions were divided into three sections: (1) participants’ backgrounds and prior knowledge, (2) participants’ experiences and opinions on carbon footprint and energy consumption, and (3), acceptance criteria and voluntary adoption of automated carbon footprint reduction tools.

Data Collection: During interviews, we introduced the background and purpose of the research. We then conducted the semi-structured interview. Following completion, we answered their questions and shared information on the project.

From November to December 2022, all interviews were conducted online and lasted between 20 and 30 minutes. Participants were interviewed in English. To analyze the results, we recorded both video and audio. We transcribed our records using a denaturalism approach [17] following the completion of each interview. Transcriptions were performed using a speech-to-text tool. We referred to the recordings to make clarifications.

Data Analysis: We adopted the thematic coding method based on the guidelines of Thomas et al. [28]. We familiarized ourselves with the data by reading the transcript repeatedly and identifying relevant segments (codes). These codes were organized and aggregated into themes and sub-themes. After each interview, we modified the codes and themes, and paused for discussion when needed. We stopped when no new codes were found in transcripts. Throughout developing the themes, attention was paid to ensuring that each code was as accurate as possible to the original response.

Coding was conducted primarily by the first author. However, the second author independently coded one interview to ensure reliability. The results were compared. As only minor differences

Table 3: Survey respondent demographics. Domain is from either a pre-selected list or self-described.

ID	Country	Position	Software Domain	Experience
I1	China	Developer	Embedded system	1-3 years
I2	Japan	Developer	Sever web service	1-3 years
I3	Denmark	Researcher	Web applications	9 years+
I4	Japan	Developer	Internal tools	4-6 years
I5	Sweden	Developer	Embedded system	7-9 years
I6	Ireland	Researcher	Programming environment	9 years+
I7	Japan	Manager	Web applications	7-9 years
I8	Sweden	Student	Embedded system	4-6 years
I9	UK	Developer	Web applications	<1 year
I10	China	Student	Embedded system	1-3 years
I11	UK	Developer	Web applications	1-3 years
I12	Sweden	Student	Various	1-3 years
I13	USA	Developer	Web applications	<1 year
I14	Sweden	Manager	Analytics	9 years+
I15	Sweden	Developer	Web and desktop applications	7-9 years
I16	Romania	Developer	Web and desktop applications	1-3 years
I17	Sweden	Developer	Data analytics	1-3 years
I18	Sweden	Developer	Various applications	1-3 years
I19	Sweden	Manager	Web applications	9 years+
I20	Sweden	Developer	Data analysis application	4-6 years
I21	Sweden	Developer	Web applications	9 years+
I22	India	Manager	Web applications	9 years+
I23	Ireland	Developer	Windows and Web applications	9 years+
I24	Sweden	Manager	Enterprise software	9 years+
I25	Sweden	Manager	Business intelligence	9 years+
I26	Sweden	Developer	Visual analysis software	9 years+
I27	Sweden	Developer	Visual analysis software	9 years+
I28	Sweden	Developer	On-prem client, cloud service	9 years+
I29	France	Developer	Web applications	4-6 years
I30	Japan	Developer	Mobile mini applications	<1 year
I31	Japan	Developer	Embedded system	1-3 years
I32	Japan	Developer	System software	1-3 years
I33	Japan	Developer	Web applications	<1 year
I34	Japan	Developer	Web applications	1-3 years
I35	Japan	Developer	Cloud web applications	1-3 years
I36	Japan	Developer	Web applications	1-3 years
I37	Japan	Developer	Artificial Intelligence	1-3 years
I38	Japan	Developer	Web applications	1-3 years
I39	Japan	Unemployed	N/A	1-3 years
I40	Sweden	Developer	SAAS	1-3 years

were observed, coding of the remaining interviews proceeded. The identified themes and sub-themes are explained in Table 2.

4.2 Survey

Population and Sampling: Our population and sampling method are the same as in the interviews. We sent the electric questionnaire directly to some participants, and also distributed the survey on LinkedIn, Facebook, Twitter, and Mastadon. Between November–December 2022, 40 respondents completed the survey.

Participant Demographics: Table 3 shows demographic information on the 40 respondents. The survey participants come from various countries—although most are still from Japan and Sweden—with different roles, experience levels, and development domains.

Survey Guide: The survey questions and their mapping to our research questions are in our replication package [2]. To ensure a high response rate, the survey was designed to be as brief—lasting between 10-15 minutes. The 26 questions consist of open-ended, multiple choice, ordinal scale, and interval scale questions [25].

Data Collection: The partially-structured questionnaire design method was adopted to ensure participants had freedom to express their opinions. The questionnaire is divided into three parts, mapped to the same topics as the interviews.

Data Analysis: We use descriptive statistics [14] to analyze quantitative data. Qualitative data was incorporated into our thematic mapping from the interviews.

5 RESULTS AND DISCUSSION

5.1 Existing Knowledge (RQ1)

As shown in Figure 1, 80.00% of survey participants are either unfamiliar with or only have a little knowledge on carbon footprint. Participants had somewhat more knowledge on energy consumption (Figure 2), but 62.50% of participants still had no or little knowledge.

Encouragingly, however, many interviewees had—at least—basic knowledge of factors that may impact energy consumption or carbon footprint. For example:

> "That's not something that I think about daily. The only thing I can think about is all the things that we store on service, in the cloud, of course, those computers need electricity. There's been a lot of talk about mining cryptocurrency. It's not a sustainable way of handling money." - **P4**
>
> "I'm not so familiar with that area, but I know GPU uses a bunch of electricity. - **P8**

Another noted how sources of energy affect carbon footprint:

> "[Carbon footprint] depends on where the energy comes from. If the energy is carbon neutral, then the footprint is still small, even if you consume lots of energy ... Most of our customers are not in Sweden, but most of our development is in Sweden. I would say majority of the energy in Sweden is not carbon-based ... water, wind, nuclear, and so on." - **P1**

As shown in Figure 3, as the experience of developers grows, there is also some increase in their median level of knowledge on both topics. Over time, developers tend to acquire knowledge of specialized topics and be more confident in their knowledge. The median knowledge of carbon footprint remains at a relatively low level—between "little" and "some" knowledge—but does rise. The median level of knowledge on energy consumption rises more noticeably to "some" knowledge.

5.2 Developer Opinions and Practices (RQ2)

Developer Opinions: Figure 4 shows a largely balanced view on whether software carbon footprint contributes to climate change, with a plurality (47.50%) expressing a neutral view. However, somewhat more participants agree (30.00%) than disagree (22.50%).

Figure 5 shows that more participants (40.00%) agree that carbon footprint should be considered than disagree (17.50%). Again, however, a plurality are neutral (42.50%). In both cases, there was no discernible change in opinion as developers gained experience.

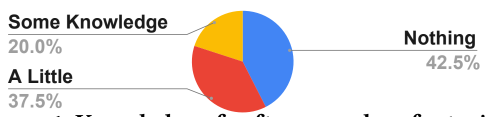

Figure 1: Knowledge of software carbon footprint.

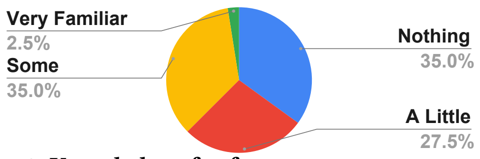

Figure 2: Knowledge of software energy consumption.

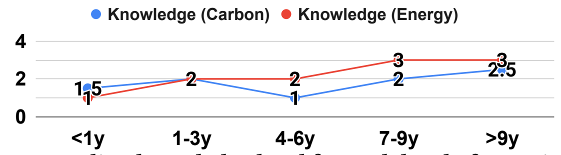

Figure 3: Median knowledge level for each level of experience (1="Nothing", 4="Very Familiar").

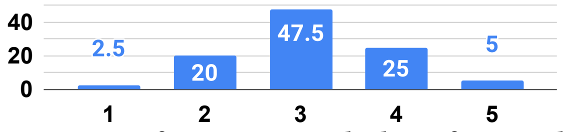

Figure 4: Degree of agreement on whether software carbon footprint contributes to climate change (1 = "strongly disagree", 5 = "strongly agree").

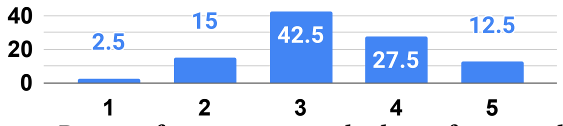

Figure 5: Degree of agreement on whether software carbon footprint should be considered and controlled (1 = "strongly disagree", 5 = "strongly agree").

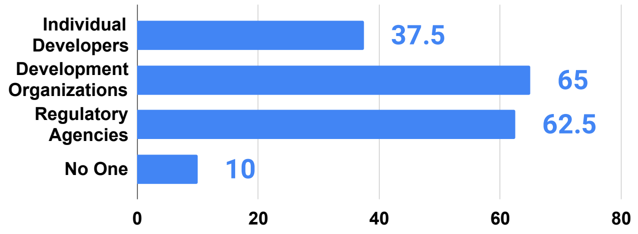

Figure 6: Percentage of respondents that believe a particular entity bares responsibility for controlling carbon footprint.

Figure 6 shows participants' views on who holds responsibility for considering or controlling energy consumption or carbon footprint. Participants could select more than one option. The majority of respondents felt that both organizations (65.00%) and regulatory agencies (60.00%) should bare responsibility. Only 10.00% of participants believe that no one holds responsibility. Interviewees also noted that the developer must comply with the company's development rules, business goal, and release criteria. Therefore, they may not have the option of reducing energy consumption.

Development organizations could take a larger responsibility by raising awareness among their staff, e.g., hosting seminars with professionals in climate-related areas or training programs that teach ecologically sustainable development practices. Companies could also prioritize energy consumption or carbon footprint as part of project goals and impose policies to control their impact on the environment. Regulatory agencies can also formulate stronger policies regarding the IT industry.

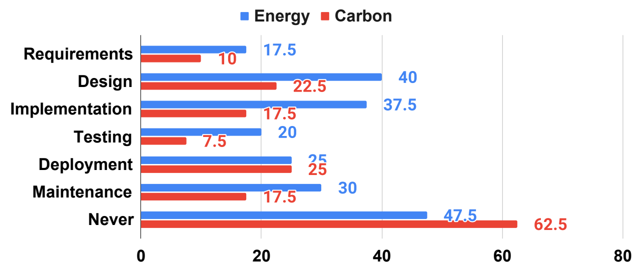

Figure 7: Percentage of survey participants that considered energy consumption or carbon footprint at a stage.

Development Stages: Carbon footprint and energy consumption can be taken into account during multiple development stages. Interview participants considered both during design, testing, and maintenance. During the design phase, developers estimate the number of end users and identify resources (e.g., number and capability of servers) they will need:

> "During the design phase, we're going to make some estimations. We're going to decide what technologies are we going to use, how many servers we are going to have, what is the load that we are expecting." - **P2**

Additionally, problematic energy consumption can be observed and mitigated after the system is deployed:

> "... When we actually run something in production and see that ... we are actually overloading the systems, we need to do something about that." - **P3**

Figure 7 illustrates when survey participants considered energy consumption and carbon footprint. In both cases, the largest percentage of participants—47.50% for energy consumption and 62.50% for carbon footprint—never took action. For energy consumption, the most consideration is given during design (40.00%) and implementation (37.50%) phases. During design, decisions are made on the system architecture, which often must incorporate consideration of energy. These decisions are realized during implementation.

Many also considered carbon footprint during design (22.50%). However, deployment (25.00%) was the most common phase. Gaining an accurate estimation of carbon footprint may be difficult before the system is in operation, where usage statistics can be gathered as well as knowledge of where users and data centers are located. Some energy consumption decisions can be made without such information, so energy consumption could also be used early in development to control carbon footprint. Then, once the system is deployed, statistics on carbon footprint can be gathered and changes can be made, if needed.

Actions Taken: Most interviewees have not taken concrete actions to address energy or carbon footprint. However, many had reduced resource usage, e.g., by compressing elements or bypassing unnecessary interactions:

> "To make the web pages faster, or save time or resources, we reduced our resources, e.g., we compressed images or music, or [added] some easier way to click to the bottom or go to the next pages." - **P6**

These actions indirectly affect carbon footprint:

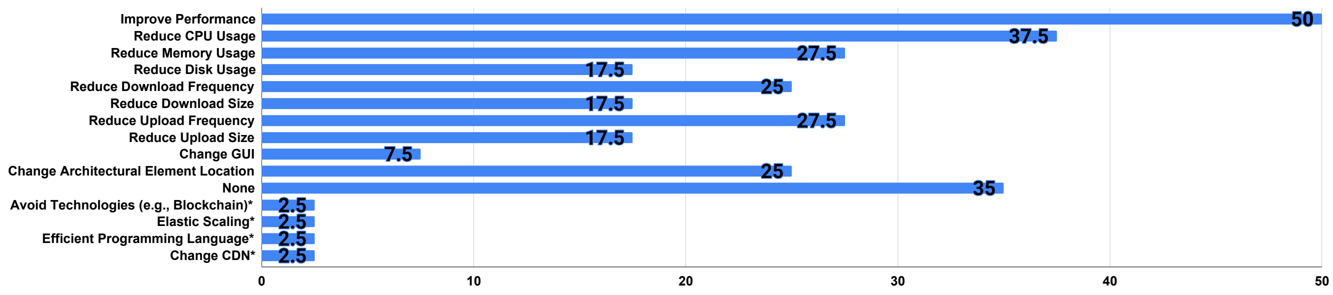

Figure 8: Percentage of participants that took actions to reduce energy consumption (* indicates a participant-added response).

> "If I'm a good engineer ... I indirectly reduce cost, indirectly improve code, indirectly reduce carbon footprint." - **P5**

Figure 8 illustrates the percentage of participants that took certain concrete actions to reduce energy consumption. The most common practice was to improve performance (50.00%), followed by reducing CPU (37.50%) and memory consumption (27.50%) or data upload frequency (27.50%). Survey participants were also asked what actions they had taken to reduce carbon footprint. Many responses specifically discussed energy and resource consumption:

> "I have mostly cared about energy consumption in battery powered devices where the goal is to have the device in a lower power state ("deep sleep") as much as possible." - **I15**
>
> "Elastic scaling so services are scaled up when needed." - **I20**
>
> "Reduce memory consumption: tried to make the software efficient starting from design. Tried to use objective language and reuse data using instances." - **I31**

Another participant noted actions developers can take to reduce the footprint of the development process:

> "I have taken steps to reduce energy consumption needed for developing software (by shutting down unused machines and reducing background processes and other things on development machines to reduce power consumption)." - **I21**

Measurement and Evaluation: While most interviewees have not formally evaluated energy or carbon footprint, some have measured energy consumption through costs and resource usage:

> "We don't evaluate energy consumption, but we do evaluate cost, which is directly related ... [To reduce] cost of service, we design good systems ... which are more efficient, consume less cost, and improve user experience." - **P8**
>
> "... It's more about how can we make sure that we are not using so much resources on the service, more than [evaluating] the energy consumption ... It's more like, well, we discovered that we are loading the the service too much ... We're not doing any formal assessment of it." - **P1**

Figure 9 shows the direct or indirect measures survey participants have used to assess energy or carbon footprint. Many have never measured either (42.50% for energy, 70.00% for carbon). Among the respondents who measured either, CPU (50.00%, 22.50%) and memory usage (45.00%, 20.00%) were the most common methods.

As shown in Figure 10, most developers have also never formally evaluated either (67.50% for energy, 90.00% for carbon). For energy consumption, the most common method was manual or exploratory

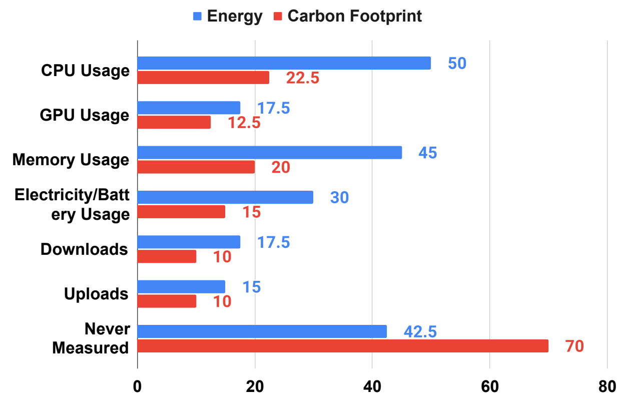

Figure 9: Percentage of survey participants that applied a measurement for energy consumption or carbon footprint.

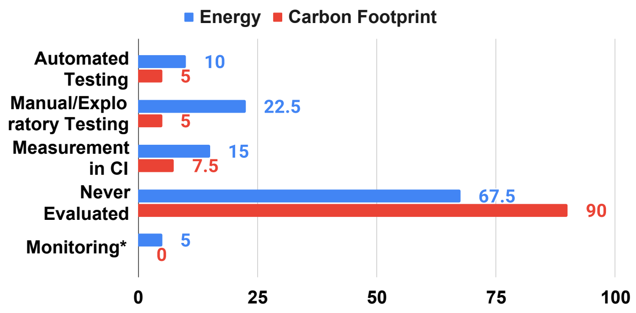

Figure 10: Percentage of participants that applied a technique to evaluate energy consumption or carbon footprint.

testing (22.50%), followed by measurement during CI (15.00%). For carbon footprint, the most common method was during CI (7.50%).

5.3 Requirements for Automated Tools (RQ3)

During the interviews, all participants expressed a positive attitude toward the idea of an automated carbon footprint reduction tool and regarded the concept as exciting and potentially helpful:

> "We want to reduce the amount of power our software requires. ... it's part of the green message we need to send to the world ... I don't really like bitcoin mining and that kind of thing, so I like what the theory is doing, changing to a much more energy-reduced algorithm for the mining." - **P1**

70.00% of survey participants were either willing or strongly willing to try a tool (Figure 11), and only 7.50% of participants were unwilling. However, Figure 12 shows that 35.00% of respondents

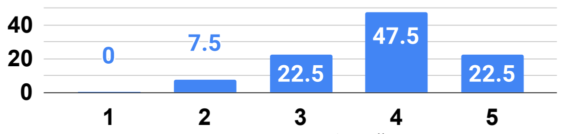

Figure 11: Willingness to use a tool (1 = "Strongly Unwilling", 5 = "Strongly Willing").

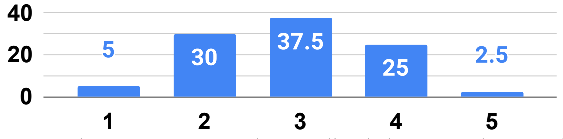

Figure 12: Skepticism towards a tool's ability to reduce carbon footprint (1 = "Highly Skeptical", 5 = "Highly Unskeptical").

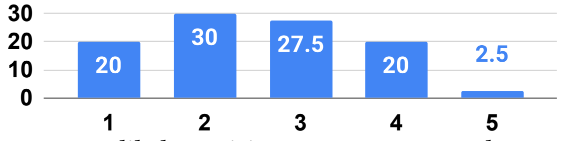

Figure 13: How likely participants are to trust a tool to modify their code (1 = "Very Unlikely", 5 = "Very Likely").

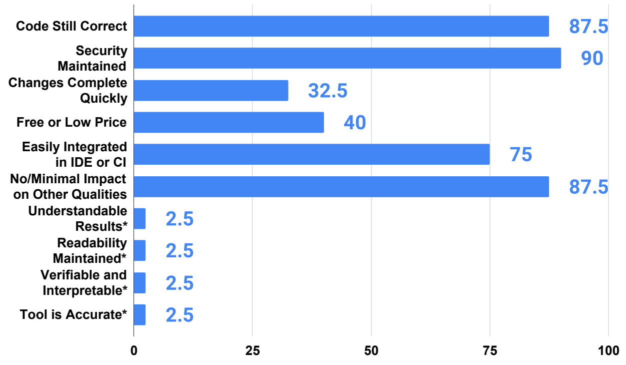

Figure 14: Percentage of respondents that believe a requirement must be met to trust an automated tool (* indicates a participant-added response).

were either skeptical or highly skeptical of the ability of a tool to successfully reduce carbon footprint, and a further 37.50% had neutral expectations.

Figure 13 indicates that participants may not trust a tool to modify their code, with 50.00% either unlikely or very unlikely to trust the tool, and a further 27.50% neutral. Only one respondent indicated that they would be very likely to trust a tool. A participant's experience had little impact on willingness or skepticism. However, participants with <3 years of experience were less likely (median of 2.00) to trust than those with more experience (median 3.00).

Some skepticism seems reasonable. Even if a tool could reduce carbon footprint without supervision, the changes it made could result in incorrect behavior or reduction in other qualities. Care must be taken to reduce the likelihood of such an occurrence.

Interview and survey respondents were asked about the requirements that would have to be met to trust a tool. Among survey participants (Figure 14), the top requirements are that security is maintained (90%), that the code still operates correctly (87.50%), and that there is no—or minimal—negative impact on other qualities (87.50%). Security is one of the most significant qualities of software, with major legal and financial implications. Tools should also not introduce faults, and reducing carbon footprint may not a priority if it comes at the expense of qualities such as performance.

Other participants asked that the tool be easily integratable into a Continuous Integration (CI) pipeline, be available for free or a low price, and complete changes quickly:

> "If I think I can trust it, it depends on how well it will fit into our development processes and how easy it is to integrate it ... If it's very expensive, then you have to weigh that against how much are we willing to spend on detecting energy consumption." - **P1**

To support integration into CI, the tool should offer an API or CLI, and should be installable through a package manager, (e.g., `pip`).

Further, the tool should produce understandable code and interpretable results—that is, users should understand how and why changes were made. For example, the tool could provide a report with an explanation of code changes and impact on carbon footprint. Such data and rationale can enable verification and trust.

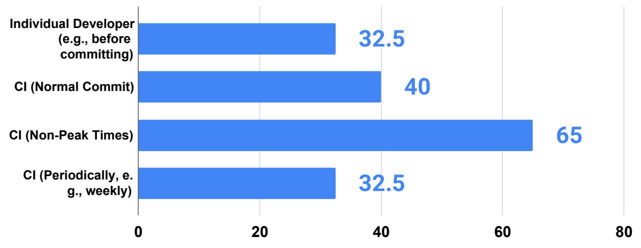

Figure 15: Percentage of respondents that would use the tool in a particular point in the development workflow.

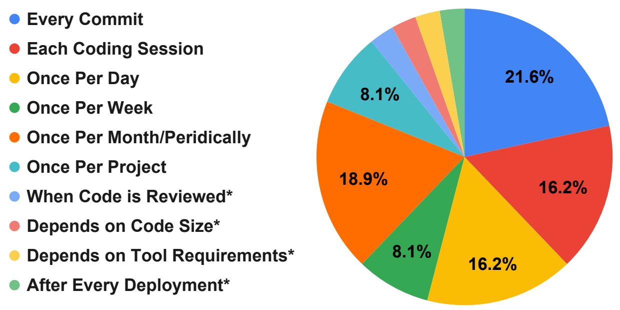

Figure 16: Frequency that survey participants would apply the tool (* indicates a participant-added response, each 2.5%).

5.4 Use in Development Workflow (RQ4)

Participants were asked about how (and how often) they would apply a tool. Many were interested in integrating this tool into an existing CI pipeline—as part of the steps performed to build and test software when changes are pushed to a shared repository.

As shown in Figure 15, 65.00% of participants would apply the tool in CI during non-peak times (e.g., overnight) to not interfere with normal development and to increase the likelihood that code is in a working state. The tool could also take more time if results are not needed quickly. 40.00% of respondents would apply the tool as part of CI after a normal commit. 32.50% of respondents would also apply the tool periodically in CI or before committing code.

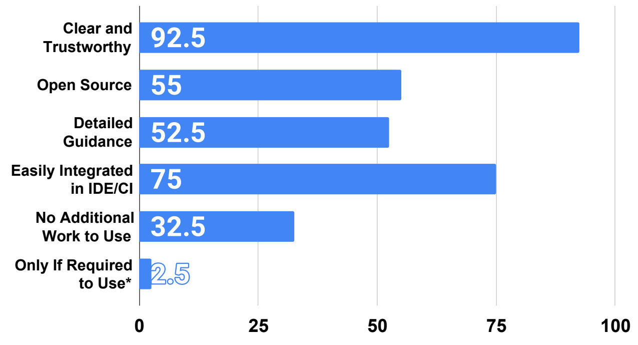

Figure 17: Percentage of respondents that believe that a factor would encourage voluntary adoption (* indicates a participant-added response)

Many indicated (Figure 16) that they would apply the tool at a relatively high frequency—split between after every commit (21.60%), after every coding session (16.20%), or daily (16.20%). However, a significant portion would instead apply it periodically (18.90%), possibly to allow code to stabilize before being improved.

5.5 Voluntary Adoption (RQ5)

Participants were also asked how to encourage voluntary adoption of tools that reduce software carbon footprint. The views of survey participants are shown in Figure 17.

The most important factor is that the tool shows a track record of clear and trustworthy results (92.50%). Testimonials from existing users could encourage adoption. One interviewee noted:

> "I will wait until the wider community accepts it. First, I will see if this is doing good. I will try to understand as much as possible what it is doing before I start using this tool."- **P6**

The tool could offer a dashboard or produce a report that visualizes changes to the source code, CPU usage, RAM usage, and carbon footprint so developers can easily understand the changes made.

Apart from that, the ability to easily integrate the tool was essential for 75.00% of the participants. 32.50% of respondents also asked that no additional work be needed to use the tool. This tool should be easy to install, integrate, and use. Interviewees added:

> "I think the main constraint in this case would be that it does not use a lot of resources and ... should be not interrupting any development work. It's okay if it takes longer to analyze once the changes are pushed, because it can just run overnight and I don't need to worry about that. But if I use it during development, I would like it to not affect any of my development experience." - **P7**

Other important factors included the tool being open-source and there being detailed guidance on how to use the tool. If the tool is open-source, those who are interested in it can gain a better understanding of its algorithm and methods. A community could form that expands the tool's capabilities and performance. Detailed documentation should provide clear and understandable instructions on how to install and use the tool.

Table 4: Distribution of responses from Swedish and Japanese participants on who bares responsibility.

	Individuals	Organizations	Regulatory Agencies	No One
Japan	38.00%	85.00%	70.00%	7.00%
Sweden	20.00%	40.00%	67.00%	20.00%

5.6 Contrasting Respondents

Many respondents are located in either Sweden or Japan, enabling a contrast between two countries with different traditions and cultures. First, we observed that Swedish respondents have a higher average level of knowledge on both carbon footprint and energy consumption than Japanese respondents.

Participants in Sweden and Japan placed different emphasis on where responsibility should lie. Table 4 contrasts opinions on responsibility. Japanese respondents place the strongest emphasis on organization. Swedish respondents placed the strongest emphasis on regulatory agencies. Swedish respondents also potentially have greater uncertainty on whether any party bares responsibility. Across all three parties, lower proportions of Swedish respondents believed that any was responsible. A higher proportion of Swedish respondents also specifically stated that no one was responsible.

Developers in Japan and Sweden were almost equally interested in trying an automated carbon footprint reduction tool. However, there were differences in their expectations. On average, Japanese respondents were rated at 3.23 on skepticism—neutral—while Swedish respondents were rated at 2.40—skeptical. Similarly, Japanese developers rated neutral—2.76—in their likeliness to trust a tool, while Swedish developers rated 2.06, (unlikely). Increased skepticism could result from increased knowledge of the core concepts.

6 THREATS TO VALIDITY

Conclusion Validity: The number of responses may affect conclusion reliability. However, our thematic findings reached saturation. Further interviews or survey results could enrich our findings, but may not produce significant additions.

Construct Validity: The interviews or survey could have missing or confusing questions. There is also a risk that participants may not be familiar with particular terminology. We provided an introduction to reduce this risk. The use of semi-structured interviews allowed us to ask follow-up questions. We also conducted pre-testing of the interview and survey.

External Validity: Generalizability of our findings is influenced by the number and background of participants. Our participants represent many development roles, experience levels, and product domains. Therefore, we believe that our results are relatively applicable to the software development industry. We also contrast results between Sweden and Japan.

Internal Validity: Thematic coding suffers from known bias threats. We mitigated these by performing independent coding and comparing results, finding sufficient agreement. We make our data available, increasing transparency.

7 CONCLUSIONS

We observed that many developers lack knowledge. However, some had basic understanding of factors affecting energy consumption and gain knowledge over time. Many felt neutrally on whether software contributes to climate change and whether carbon footprint should be controlled. However, a greater proportion agree than disagree with both. The majority of participants feel that both development organizations and regulatory agencies bare responsibility for controlling carbon footprint. We also explored when energy and carbon footprint are considered, how they are measured, and what practices are used to perform evaluations of both.

Many participants are willing to try an automated tool, but there was significant skepticism and lack of trust. It is clear, however, that such tools must not compromise security, correctness, or other important qualities. They also must integrate into a CI pipeline, be well-documented, be reasonably priced or—preferably—open source, and offer transparent and trustworthy results.

This study provides a foundation for future research addressing software carbon footprint. Attention should be paid in future work to increasing developer awareness of these issues, offering recommendations and best practices, exploring policy implications, and developing automated tool support.

REFERENCES

[1] Anders SG Andrae. 2020. New perspectives on internet electricity use in 2030. *Engineering and Applied Science Letters* 3, 2 (2020), 19–31.

[2] Anonymous. 2023. *Replication Data for "Exploring Developer Views on Software Carbon Footprint and its Potential for Automated Reduction"*. https://doi.org/10.5281/zenodo.7597662

[3] Tom Bawdin. 2016. Global warming: Data centres to consume three times as much energy in next decade, experts warn. *The Independent* (2016). Available from https://www.independent.co.uk/environment/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html.

[4] Bobby R Bruce, Justyna Petke, and Mark Harman. 2015. Reducing energy consumption using genetic improvement. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. 1327–1334.

[5] Daniela S Cruzes and Tore Dyba. 2011. Recommended steps for thematic synthesis in software engineering. In *2011 international symposium on empirical software engineering and measurement*. IEEE, 275–284.

[6] Sarah Darby et al. 2006. The effectiveness of feedback on energy consumption. *A Review for DEFRA of the Literature on Metering, Billing and direct Displays* 486, 2006 (2006), 26.

[7] Victor De La Luz, Mahmut Kandemir, and Ibrahim Kolcu. 2002. Automatic data migration for reducing energy consumption in multi-bank memory systems. In *Proceedings 2002 Design Automation Conference (IEEE Cat. No. 02CH37324)*. IEEE, 213–218.

[8] Jonathan Dorn, Jeremy Lacomis, Westley Weimer, and Stephanie Forrest. 2017. Automatically exploring tradeoffs between software output fidelity and energy costs. *IEEE Transactions on Software Engineering* 45, 3 (2017), 219–236.

[9] Ilker Etikan, Sulaiman Abubakar Musa, Rukayya Sunusi Alkassim, et al. 2016. Comparison of convenience sampling and purposive sampling. *American journal of theoretical and applied statistics* 5, 1 (2016), 1–4.

[10] Göran Finnveden, Michael Z Hauschild, Tomas Ekvall, Jeroen Guinée, Reinout Heijungs, Stefanie Hellweg, Annette Koehler, David Pennington, and Sangwon Suh. 2009. Recent developments in life cycle assessment. *Journal of environmental management* 91, 1 (2009), 1–21.

[11] Karen Hao. 2019. Training a single AI model can emit as much carbon as five cars in their lifetimes. *MIT Technology Review* (2019).

[12] Mark Harman, William B Langdon, Yue Jia, David R White, Andrea Arcuri, and John A Clark. 2012. The GISMOE challenge: Constructing the pareto program surface using genetic programming to find better programs (keynote paper). In *2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 1–14.

[13] Mark Harman, S Afshin Mansouri, and Yuanyuan Zhang. 2009. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. (2009).

[14] Anne Lazaraton. 2000. Current trends in research methodology and statistics in applied linguistics. *TESOL quarterly* 34, 1 (2000), 175–181.

[15] Irene Manotas, Lori Pollock, and James Clause. 2014. Seeds: A software engineer's energy-optimization decision support framework. In *Proceedings of the 36th International Conference on Software Engineering*. 503–514.

[16] Vojtech Mrazek, Zdenek Vasicek, and Lukas Sekanina. 2015. Evolutionary approximation of software for embedded systems: Median function. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. 795–801.

[17] Daniel G Oliver, Julianne M Serovich, and Tina L Mason. 2005. Constraints and opportunities with interview transcription: Towards reflection in qualitative research. *Social forces* 84, 2 (2005), 1273–1289.

[18] Zakaria Ournani, Romain Rouvoy, Pierre Rust, and Joel Penhoat. 2020. 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)*. 1–12.

[19] Candy Pang, Abram Hindle, Bram Adams, and Ahmed E Hassan. 2015. What do programmers know about software energy consumption? *IEEE Software* 33, 3 (2015), 83–89.

[20] Gustavo Pinto, Fernando Castor, and Yu David Liu. 2014. Mining questions about software energy consumption. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. 22–31.

[21] K Ratheeswari. 2018. Information communication technology in education. *Journal of Applied and Advanced research* 3, 1 (2018), 45–47.

[22] David R Reidmiller, Christopher W Avery, David R Easterling, Kenneth E Kunkel, Kristin LM Lewis, TK Maycock, and Bradley C Stewart. 2017. Impacts, risks, and adaptation in the United States: Fourth national climate assessment, volume II. (2017).

[23] Adrian Sampson, Călin Caşcaval, Luis Ceze, Pablo Montesinos, and Dario Suarez Gracia. 2012. Automatic discovery of performance and energy pitfalls in html and css. In *2012 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 82–83.

[24] Stefan Steinke, Lars Wehmeyer, Bo-Sik Lee, and Peter Marwedel. 2002. Assigning program and data objects to scratchpad for energy reduction. In *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 409–415.

[25] Hamed Taherdoost. 2016. How to design and create an effective survey/questionnaire; A step by step guide. *International Journal of Academic Research in Management (IJARM)* 5, 4 (2016), 37–41.

[26] Juha Taina. 2010. How green is your software?. In *International Conference of Software Business*. Springer, 151–162.

[27] Narendran Thiagarajan, Gaurav Aggarwal, Angela Nicoara, Dan Boneh, and Jatinder Pal Singh. 2012. Who killed my battery? Analyzing mobile browser energy consumption. In *Proceedings of the 21st international conference on World Wide Web*. 41–50.

[28] David R Thomas. 2003. A general inductive approach for qualitative data analysis. (2003).

[29] WRI Wbcsd. 2004. The greenhouse gas protocol. *A corporate accounting and reporting standard, Rev. ed. Washington, DC, Conches-Geneva* (2004).

[30] David Robert White. 2009. *Genetic programming for low-resource systems*. Ph. D. Dissertation. University of York.

[31] David R White, John Clark, Jeremy Jacob, and Simon M Poulding. 2008. Searching for resource-efficient programs: Low-power pseudorandom number generators. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. 1775–1782.

[32] Thomas Wiedmann and Jan Minx. 2008. A definition of 'carbon footprint'. *Ecological economics research trends* 1, 2008 (2008), 1–11.

[33] Yuhao Zhu and Vijay Janapa Reddi. 2013. High-performance and energy-efficient mobile web browsing on big/little systems. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 13–24.