An Investigation of Green Software Engineering

Martina Freed[1], Sylwia Bielinska[1], Carla Buckley[1], Andreea Coptu[1], Murat Yilmaz[2] [0000-0002-2446-3224], Richard Messnarz[3], and Paul M. Clarke[1,4] [0000-0002-4487-627X]

[1] School of Computing, Dublin City University, Dublin, Ireland
{martina.freed2,sylwia.bielinska2,carla.buckley38,andreea.coptu2} @mail.dcu.ie

[2] Department of Computer Engineering, Gazi University, Ankara, Turkey
my@gazi.edu.tr

[3] ISCN, the International Software Consulting Network, Graz, Austria
rmess@iscn.com

[4] Lero, the Science Foundation Ireland Research Center for Software
paul.m.clarke@dcu.ie

Abstract. The urgency of sustainability concerns has intensified in recent years, sounding alarm bells over the planet's condition and prompting nearly every industry and practice to reassess their contributions to the climate crisis. Software engineering is not immune to this scrutiny. Software engineering practices significantly affect the environment and may not align with sustainability goals. Although sustainability is a relatively recent focus in software engineering, it has garnered increased attention, with numerous studies addressing various concerns and practices. Green software engineering aspires to develop dependable, enduring, and sustainable software that fulfills user requirements while minimizing environmental impacts. As this green paradigm gains traction in software engineering, practitioners must incorporate sustainability considerations into future software designs. However, despite the surge in green software engineering research, a universally accepted definition and framework remain elusive. This paper outlines green software engineering by explaining its principles, challenges, and methods for measuring and evaluating software effectiveness in this context.

Keywords: sustainability, energy efficiency, software engineering, green software engineering

1 Introduction

Software has the power to support the environment, and create environmentally friendly solutions to processes that previously contributed to the carbon footprint and climate change [1]. However, it also has the potential to worsen the climate crisis if proper steps are not taken to manage energy use and the carbon footprint of the software itself. Green software development is becoming a discipline of its own: some have even suggested a new green software engineering (hereafter referred to as “Green SE”) process that is a change on the traditional software development life cycle with a focus on lowering

resource use [2]. With this new discipline comes new implications for software developers, new practices, new challenges, and new ways of evaluating if the software is effective in respect of sustainability objectives.

Green IT has been described as a discipline that considers and optimizes the resources consumed by the life cycle of Information and Communication Technology [1]. This idea can also be applied to Green SE. However, no single universally accepted definition of Green SE has been identified in the literature. This research sets out to explain Green SE by examining its practices, concerns, challenges, and analysis.

The objectives of this paper are to:
1. Provide a contemporary understanding of Green SE in the context of the growing urgency for sustainability.
2. Examine the environmental implications of traditional software engineering practices and their alignment with sustainable objectives.
3. Review the evolution and current state of Green SE research, addressing its various concerns and practices.
4. Elucidate the principles, challenges, and potential methods for measuring and evaluating the effectiveness of Green SE.
5. Contribute to the development of a universally accepted definition and framework for Green SE, paving the way for future sustainable software design practices.

This paper is organized as follows: Section 2 details the research methodology, elucidating our inclusion/exclusion criteria and delineating the related research questions. Section 3 encompasses the analysis, featuring subsections addressing each of the four specific research questions. Section 4 contemplates the limitations of our study, while Section 5 highlights avenues for future research. Finally, Section 6 offers the research conclusions.

2 Research Methodology

2.1 Search Strings

A multivocal literature review (MLR) [49] was conducted for this research. While most of the sources cited are peer reviewed (published academic literature), grey literature is also included. The grey literature included is from platforms which ostensibly incorporate robust oversight and moderation.

Google Scholar was used to identify the literature sources. Search strings included ‘green software engineering’, ‘energy efficiency software’, ‘green software engineering practices’, ‘green software engineering sustainable design’, ‘disadvantages of green software engineering’, ‘cost of green software engineering’, ‘green software engineering universal framework’, ‘life cycle assessment’, ‘green metrics’.

2.2 Inclusion/Exclusion

When searching on Google Scholar, we defined our inclusion/exclusion criteria to be the first 20 articles from 2019 or later. Software engineering, and especially Green SE, is an evolving discipline, so we wanted to ensure our research was current. We also took a brief look into the topics of the papers that came up to decide if they were relevant to our research questions. Although we limited our results on Google Scholar to papers since 2019, we did include some sources that were older. This was because they were cited by a paper published in 2019 or later, and they provided useful background information on the topic. 180 papers were identified as potential sources. We read the titles, key words, and/or the abstracts to evaluate the relevance and credibility. We also looked into sources that were cited by the original sources to find more information. Ultimately, we included a total of 48 sources.

To address the central research question of this paper, "What is Green Software Engineering?", we have identified the following four subsidiary research questions:

- What are the fundamental principles and practices of green software engineering?
- In what ways can software developers decrease the energy consumption of software?
- What are the primary challenges confronting green software engineering?
- How can we assess and quantify the environmental impact of green software engineering?

3 Analysis

3.1 What are the principles and practices of Green Software Engineering?

The growing demand for software products and services has led to an increase in the energy consumption and carbon footprint of the IT industry. As a result, it has become critical to adopt sustainable practices in software engineering to mitigate the environmental impact of the software development lifecycle [3]. The adoption of a green mindset is essential for all stakeholders, including software developers, software development organisations, end-users, and society as a whole. According to Professor San Murugesan [4], the IT sector and users must develop a positive attitude toward addressing environmental concerns and adopt forward-looking, green-friendly policies and practices.

Green SE consists of a set of principles and practices aimed at reducing the environmental impact of software development. The principles are high-level guidelines that describe the core values and concepts of Green SE, while the practices are specific actions that can be taken to implement those principles. The principles provide a conceptual framework for Green SE, while the practices offer practical guidance on how to achieve the goals set out by the principles. By adopting Green SE principles and practices, software developers can aspire to reduce the carbon footprint of software

development and contribute to a more sustainable future. The principles help to guide decision-making and provide a sense of direction, while the practices ensure that actions are aligned with the principles and contribute to the overall goal of environmental sustainability.

This section explores the key principles and practices of Green SE. The key principles include energy efficiency, sustainable design, life cycle assessment, and renewable energy sources. For each principle, we discuss the key practices that can be implemented to achieve the goals set out by the principles.

Energy efficiency is one of the key principles of Green SE. It involves optimizing energy usage at every stage of the software development lifecycle. This principle can be achieved by employing best practices such as reducing computational complexity, power management, and using cloud-based services. Minimizing computational complexity involves optimizing the algorithms and data structures used in the software to reduce the amount of processing power required. Power management involves optimizing the hardware and software settings to minimize energy consumption [5]. Finally, using cloud-based services is another way to achieve energy efficiency. Cloud-based services enable software to be run on remote servers, which are optimized for energy efficiency [6]. This means that the energy required to power the software is not consumed on the user's hardware, which can be less energy-efficient.

Sustainable design is a key principle focused on designing software systems that are environmentally friendly and sustainable. Essential practices for sustainable design include minimizing energy consumption, reducing waste, and designing for the future [7]. Minimizing energy consumption involves designing software systems to be energy-efficient at every stage of the software development lifecycle, while reducing waste involves designing software systems that minimize the amount of waste generated at every stage of the software development lifecycle. Designing for the future involves designing software systems that are flexible, scalable, and adaptable to future changes in technology and user needs [8].

Life cycle assessment is the principle of evaluating the environmental impact of software throughout its entire lifecycle. A comprehensive life cycle assessment involves practices such as carbon footprint analysis, energy consumption analysis, and waste generation analysis. Carbon footprint analysis involves measuring the amount of carbon emissions generated by the software at every stage of the software development lifecycle. This can be achieved by using tools such as the Software Sustainability Assessment Framework, which helps to measure the environmental impact of software [9]. Energy consumption analysis involves measuring the amount of energy consumed by the software at every stage of the software development lifecycle. This can be achieved by using tools such as the Energy Consumption Analysis Tool, which helps to measure the energy consumption of software.

Renewable energy sources are a key principle of Green SE that can help to reduce the environmental impact of software development. By using renewable energy sources,

such as solar or wind power, software development can be made more sustainable, reducing the reliance on non-renewable energy sources [10]. The use of renewable energy sources can be achieved through practices such as the use of green hosting services, the adoption of green data centres, and the implementation of energy-efficient hardware. One practice is the use of green hosting services. These services provide data centres that are powered by renewable energy sources such as solar or wind power. By using green hosting services, the environmental impact of software development can be minimized. Another practice involves the adoption of green data centres. Green data centres are data centres that are designed to be energy-efficient and are powered by renewable energy sources [11]. Green data centres can help to reduce the carbon footprint of software development by minimizing energy consumption.

In summary, Green SE is essential for mitigating the environmental impact of the software development lifecycle. The key principles of energy efficiency, sustainable design, life cycle assessment, and renewable energy sources provide a framework for achieving environmental sustainability. The key practices associated with each principle offer practical guidance on how to achieve the goals set out by the principles, contributing to a more sustainable future.

3.2 How can software developers reduce the energy consumption of software?

Energy use tend to increase with larger populations and increased reliance on technology. Reducing energy consumption has become critical, as has concern with carbon emissions. Refactoring code is the practice of editing code to improve its design without changing its functionality. Refactoring may mean a number of different things: editing methods to make them more concise, remove duplicate code, or shorten a class [12]. Refactoring code has many potential benefits, including decreasing the use of energy while the program does the same job. For example, a triple refactoring combination on applications for portable devices written in C# and Java can considerably lower power consumption [13]. By refactoring code, software developers can not only make their code more readable and elegant, but also improve its energy efficiency.

The programming languages employed in a system may also influence the energy consumption of an application. Researchers utilized The Computer Languages Benchmark Game framework to collect 13 problems, finding that C used the least energy, memory, and was the fastest [14]. They also stated that “Compiled languages tend to be, as expected, the fastest and most energy efficient ones” (14) as opposed to interpreted and virtual machine languages. However, faster languages are not always more energy efficient [14]. This also does not mean that there is one best programming language to use for a piece of software if the developer is considering energy, time, and memory usage.

When searching for Green SE, one may find many resources about the usefulness of cloud computing. Cloud computing can enable customers to pay for software on demand instead of owning all of the hardware [15]. Before the cloud, companies and individuals often had more hardware and servers than were necessary. The cloud fixed

this problem since these services are delivered over the internet when they are needed. It can be cheaper and more energy efficient for companies [16]. However, simply because something uses the cloud does not necessarily mean it is energy efficient. Many large companies have cloud data centers, which consume massive amounts of energy to operate, take up space, and require significant energy to cool [15]. Power consumption of the cloud is on the rise, which inevitably contributes to carbon emissions. Decreased carbon emissions is a central focus of Green SE [2]. Virtualization (running multiple virtual computers on one piece of hardware), consolidation (consolidated the number of servers so there are less idle servers), a thermal aware approach to data centers, and static and dynamic power management are some cloud-based energy saving approaches [17].

Software defined networking (SDN) is a similar approach to cloud computing because it "is capable of providing the solutions without the knowledge of underground complex network architecture" [18]. SDN allows software developers to manage network operation through software. In a traditional network architecture where elements are not globally controlled [19]. In a traditional network, specific knowledge of the infrastructure may be a necessity [20]. Due to the similarities with cloud computing, there are similar concerns of energy efficiency in software defined networking. Using sleep state of end devices and managing traffic have been suggested as techniques to achieve this [18].

Beyond the energy efficiency of programs being difficult to measure, there has been uncertainty as to how aware software developers are of energy efficiency. When questions regarding energy efficiency on Stack Overflow were compared with other questions on the platform, it was discovered that questions about energy efficiency are popular and diverse in themes (code design, energy use, noise, general questions), but the answers are not always good quality [21]. Improving the answers to these questions would help software developers be prepared to write energy efficient programs.

Although energy efficiency is difficult to measure, some researchers sought out to establish commonly agreed upon terminology surrounding energy efficiency of software. The results noted that this terminology did not exist, leading to the creation of a Green Software Measurement Ontology (GSMO) that includes terms and definitions such as Test Case, Test Case Measurement, and a Measuring Instrument [22]. A hardware-based approach was used to measure energy use, with an Energy Efficiency Tester (EET) and software tool called ELLIOT used to process the data. The EET includes a power source, sensors, and a system microcontroller to gather the information gathered by the sensors and store them in memory. The Green Software Measurement process includes defining the requirements, configuring the measurement environment with the measuring instrument, performing the measurement, and doing data analysis on that test case [22]. Published in 2021, and the authors hoped that it would establish a practice on measuring energy use. Creating energy efficient software is easier if developers can gauge the energy use of software. A focus on energy efficiency can only be useful if there exists a way to measure energy use and gauge the impact.

3.3 What are the challenges facing Green Software Engineering?

As the topic of environmental sustainability becomes a widespread concern throughout many areas of computer science, there has been a recent spike in research into Green SE and sustainable computing. Although green software is a topic of increasing interest within the IT industry, it is not a technology that is commonly implemented within these industries. The reason for the lack of green software implementation can come down to three main challenges that a company may be faced with when implementing such technology. These challenges being the lack of awareness, lack of a universal framework and the cost of implementation.

Green software is still a relatively new concept. Results from a recent survey indicates that sustainable software is a new concern for software engineers and, despite a high interest in the subject, they have a low perception of the impacts of sustainability throughout the development cycle [23]. A study carried out in 2017 found that, while viewing software sustainability as important, software engineers are primarily concerned with the technical aspects of software sustainability rather than the environmental aspects [24]. When referring to software sustainability, they address organisational and economic issues but lack considerations for environmental issues. Through this study, there is a diverse understanding of what sustainable software is which suggests that a clear definition of sustainable software has yet to be refined and distributed within the IT industry. The results of this survey further reveal that software practitioners have a skewed understanding of sustainability concepts in the software development process [23]. This is due to their targeted perceptions on the reuse of code in regard to sustainable software. This perception is still relevant to Green SE as it does have a positive environmental impact. However, it prevents companies from becoming green as they have their focus only on one dimension of the four major dimensions of software sustainability as defined in a proposed sustainability framework [25].

Companies are yet to fully adapt to these dimensions and utilise any other green models, processes, methods and tools that can support the development of their software in a meaningful way. This can be further evidenced through a recent study where, out of nineteen software engineers interviewed, fourteen participants reported that they have worked on software that was not sustainable [26]. This shows that not only are companies not implementing all dimensions of green software, but they are also not prioritising the technical dimension that they put most emphasis on in the development process. Thus, it can be found that companies do not promote sustainable development within the company [23]. While companies tend to focus on the technical aspects of sustainable software, researchers are mainly interested in proposing frameworks, approaches and models [27]. This shows a gap between the level of interest between academia and the industry which needs to be aligned. It is important for companies in the IT industry to recognize the importance of Green SE and the benefits that it can provide in order to spread awareness within the industry and allow for higher rates of research and implementation of green software.

The lack of awareness of Green SE means there is an additional deficit in the amount of research going into developing a universal Green SE framework. While a number of

frameworks have been proposed, they are not being widely used in the computing industry. This is due to the lack of a universal framework that can be used by companies. Universal frameworks are an essential part of software engineering by acting as a common reference point for engineers to aid in developing software regardless of where in the world they are working. Without this framework, the companies that want to implement green software will need to research and develop their own framework to work from. Existing software engineering models such as ISO 25000 and ISO 25010 do not consider sustainability as a quality attribute [27]. As a result, over a third of organisations in Europe do not implement green IT practices and less than one fifth of the organisations monitor how their employees reduce their energy consumption [28]. The main reason given for this is that there is no official legislation in their countries enforcing green practices. The majority of Green SE research is happening in developed countries due to the peak in interest. However, in developing countries, there is little research carried out regarding this topic. The focus of these countries is mostly on developing software applications and ICT products for the developed countries [29]. As there is a lack of universal interest in the topic of Green SE, there is insufficient incentive for research to be invested towards a universal green software framework. The existence of a universal framework is essential to allow for better Green SE processes and will promote the implementation of green software into their IT businesses and organisations.

The initial implementation cost of green technology may be significant as there are many aspects to the implementation. Not only is it necessary to change the coding standards in accordance with a green framework, but there are also additional payments involved in research and buying new equipment [30]. Although green technology becomes more cost effective over time, this high upfront cost of the technology implementation process can deter businesses from making the switch [31]. Another additional payment concerns maintenance costs which may be increased due to the need to understand the new system and the changes that need to be analysed. Training may also be necessary to understand the original and new programming languages, systems and methods [32]. As there is no universal green software framework, there will be additional charges in order to develop a framework that can be used within the industry. The cost of designing, implementing, evaluating and deploying a framework may be of the order of c.USD$15-20 [33]. The total price of all the resources and maintenance needed for a company to convert from their current software to green software may deter them from making the switch to a more sustainable approach.

3.4 Green Software Engineering is Part of Green System Engineering?

The European Union supported the development of green and sustainable concepts and in the last 4 years also blue print projects [52] to develop skills and concepts to move towards a green economy. European studies about future skills [54] required to empower this new development have been performed for e.g. the automotive sector. When looking at solutions for green technologies in the automotive sector it is obvious that the product strategy, the system and the software life cycle are interlinked [51], e.g.

software implementing an electric powertrain is supporting the green economy. However, this requires a system design integrating high voltage batteries, electric motors, sensors, and software on electric control units. Moreover, if the mechanical design of the car has a high wind resistance the electric power in the battery is inefficiently consumed. So in fact all three layers (product, system, software life cycle) and their integration play a role for achieving a green solution [53].

A modern car, plane, ship, etc. may be largely controlled by functions that are implemented in software, so software is a key to change functional behavior of systems. For instance, software can switch to green mode and in this case reduce the consumption and at the same time would force the driver to an economy mode. Most cars have meanwhile an over the air update functionality. So one strategy could be an over the air update of a fleet to drive in green economy mode decided by e.g. a region or government. This in fact makes software the nearly most important turn key.

3.5 How can we measure and quantify the impact of green software engineering on the environment?

Green SE is a field that seeks to reduce the environmental impact of software development and operation. Measuring and quantifying the impact of Green SE on the environment is crucial for promoting sustainable development practices [34]. Quantitative and qualitative methods can be used to achieve a comprehensive understanding of the environmental impact of software development and operation [35].

Quantitative methods include life cycle assessment. In the context of Green SE, life cycle assessment can be applied to assess factors such as energy consumption, greenhouse gas emissions, and resource use [36]. Life cycle assessment involves identifying and quantifying the environmental impacts of each stage of a product or service's life cycle, from raw material extraction and processing to production, use, and disposal. By doing so, life cycle assessment provides a comprehensive understanding of the environmental impact of the entire life cycle of a product or service, enabling decision-makers to identify opportunities for improvement [37].

Energy efficiency metrics such as power usage effectiveness and data center infrastructure efficiency can also be used to measure the amount of energy used by software systems during development and operation [38]. Power usage effectiveness is a ratio that measures the amount of energy used by a data center facility compared to the amount of energy used by the IT equipment it houses [39]. Data center infrastructure efficiency is similar to power usage effectiveness but takes into account the energy efficiency of the IT equipment itself. These metrics can help software developers and data center operators to identify opportunities to improve energy efficiency and reduce energy consumption [40].

Carbon emissions reduction can also be quantitatively measured by calculating the reduction in greenhouse gas emissions achieved through measures such as server consolidation, virtualization, and energy-efficient hardware. Server consolidation involves re-

ducing the number of physical servers in a data center by consolidating multiple applications onto a single server [41]. Virtualization involves creating multiple virtual servers on a single physical server, allowing for more efficient use of hardware resources [42]. By reducing the number of physical servers in a data center and optimizing the use of IT equipment, carbon emissions can be reduced.

Overall, these quantitative methods provide a rigorous and systematic approach to measuring the environmental impact of software development and operation. By quantifying factors such as energy consumption, greenhouse gas emissions, and resource use, decision-makers can identify opportunities to improve the environmental sustainability of software systems [43]. These methods also provide a basis for comparing the environmental performance of different software systems and evaluating the effectiveness of Green SE practices. Qualitative methods can provide valuable insights into the impact of Green SE on the environment. Surveys, interviews, and case studies are commonly used qualitative methods [44].

Surveys can be used to gather data on the attitudes and behaviours of software developers regarding Green SE practices. For example, a survey might ask developers about their awareness of energy-efficient coding practices or their use of sustainable software development tools [45]. The data collected from surveys can be used to identify trends in Green SE practices, as well as barriers to the adoption of these practices.

Interviews with stakeholders such as customers, employees, and management can provide insights into the impact of Green SE on business operations and customer satisfaction. For example, an interview with a customer might reveal that they are more likely to purchase software products that are developed using sustainable practices [46]. An interview with an employee might reveal that they are more likely to stay with a company that prioritises environmental sustainability. Interviews with management can provide insights into the cost-effectiveness of Green SE practices, as well as the impact of these practices on the company's bottom line.

Case studies can provide detailed information on specific Green SE projects and their impact on the environment. For example, a case study might examine the development of an energy-efficient software application and the resulting reduction in greenhouse gas emissions [47]. Case studies can also provide insights into the challenges and opportunities associated with Green SE, as well as best practices for implementing sustainable software development practices [48].

Overall, qualitative methods provide a more nuanced and detailed understanding of the impact of Green SE on the environment. By gathering data on attitudes, behaviours, and specific projects, qualitative methods can provide valuable insights into the human and organisational factors that influence the adoption of Green SE practices [46]. These methods can also help to identify opportunities for collaboration and communication among stakeholders, as well as potential barriers to the adoption of sustainable software development practices.

Green SE is a vital field for promoting sustainable development practices. Measuring and quantifying the impact of Green SE on the environment requires the use of both quantitative and qualitative methods. Life cycle assessment, energy efficiency metrics, carbon emissions reduction, surveys, interviews, and case studies are all methods that can be used to achieve a more comprehensive understanding of the environmental impact of software development and operation. By using these methods, the benefits and challenges of implementing Green SE practices can be identified, and strategies for reducing the environmental impact of software development and operation can be developed.

4 Research Limitations

When discussing Green SE, it is important to consider several limitations that may impact research and implementation. Firstly, the limited literature available may not be well-established or robust enough to draw solid conclusions, which means that research may not provide a comprehensive analysis of the topic. Additionally, the availability and quality of data can pose limitations on research as there may not be enough reliable data available to support meaningful conclusions. Moreover, the context in which Green SE practices are implemented may vary depending on industry, organisational culture, and technology infrastructure, which means that research findings may not be generalizable across different contexts. Finally, potential biases in the research design and the lack of a standardized framework or set of practices may impact the validity and reliability of findings.

Another set of limitations pertains to the implementation of Green SE practices in real-world settings. Even if practices are identified and validated, implementation may be challenging due to technical, organisational, or financial constraints. Research may not fully address these challenges due to time and resource constraints, leading to a narrow focus on specific aspects of Green SE, such as energy efficiency or sustainable design. In conclusion, a nuanced understanding of the limitations of Green SE is crucial to interpret research findings with appropriate caution and to develop effective strategies for implementing these practices.

It is important to highlight that the initial research was undertaken by four final year undergraduate students over a 6-week period. Although preliminary training in academic research and writing was provided, the core primary researchers were essentially novices. To further mitigate this risk, a senior academic was available on a weekly basis to address any questions and to direct the work. Later, the work was reviewed and extended by a team of senior academics. Nevertheless, the experience of the core researchers and the limited time frame available for the review has reduced the academic completeness of the process, as such it might be considered research methodology light, therefore tending towards an experience report. An obvious area for improvement concerns the consistent treatment of quality characteristics in the included works, which is not well reported.

5 Directions for Future Research

These findings emerged from a six-week project. Future research efforts can try to answer the question with a broader scope. Ideally, a consensus regarding the definition of Green SE would be reached, along with mutually agreed-upon methods for implementing Green SE practices. One possible solution involves conducting further surveys of software developers to gauge their understanding of Green SE. Additionally, now that sustainability has been a concern in software engineering for at least several years, the effectiveness of these Green SE practices can be evaluated. Consequently, software practitioners can discern which practices yield the most significant environmental benefits and are worth incorporating into their work.

Inefficient code and associated algorithmic implementation can increase the hardware and indeed maintenance costs, and therefore it is perhaps an appropriate time to refocus efforts on efficiency in existing system. This can be as simple as unnecessary code included in systems but never actually used, it nevertheless requires hardware resources and associated electrical supply.

6 Conclusions

The urgency to address sustainability concerns has led to a growing interest in Green SE, which aims to create reliable, sustainable software that meets the needs of users while reducing environmental impacts. Despite the recent spike in research, there is still no universally accepted definition or framework for Green SE. Through a multivocal literature review, this paper examines the fundamental principles and practices of Green SE, the obstacles confronting the field, and methods for curbing the energy consumption of software systems. As software practitioners embrace the green agenda, they will need to take sustainability into account in the future of software design, and work towards creating software that not only meets the needs of users but also minimizes the environmental impact of their work.

To achieve this goal, software developers need to adopt practices such as code reuse, energy-efficient design, and sustainable software lifecycle management. Part of this task will inevitably involve embracing emerging cloud computing paradigms such as function-as-a-service (FaaS) [50]. However, implementing Green SE practices can pose significant challenges, including technical, economic, and social barriers. To overcome these challenges, developers can leverage green metrics, quantitative and qualitative methods, and life cycle assessment to evaluate the environmental impact of their software and make data-driven decisions.

In conclusion, Green SE signifies an essential stride towards forging a sustainable future, with software developers holding a pivotal role in this pursuit. By adopting Green SE practices, developers can reduce the carbon footprint of software systems and contribute to global efforts to mitigate climate change. However, achieving this goal will require ongoing research, collaboration, and innovation in the field of software engineering.

Acknowledgements. This research is supported in part by SFI, Science Foundation Ireland (https://www.sfi.ie/) grant No SFI 13/RC/2094 P2 to–Lero - the Science Foundation Ireland Research Centre for Software.

References

1. Kern, E., Dick, M., Naumann, S., Guldner, A., Johann, T.: Green software and green software engineering–definitions, measurements, and quality aspects. In First International Conference on Information and Communication Technologies for Sustainability, pp. 87-91. ETH Zurich, Zurich, Switzerland (2013).
2. Ray, S.: Green software engineering process: moving towards sustainable software product design. Journal of Global Research in Computer Science *4*(1), 25-29 (2013).
3. Raja, SP.: Green computing and carbon footprint management in the IT sectors. IEEE Transactions on Computational Social Systems (2021).
4. Murugesan, S.: Harnessing green it: Principles and practices. IT Professional, pp. 24-33 (2008).
5. Georgiou, S., Rizou, S., Spinellis, D.: Software development lifecycle for energy efficiency. ACM Computing Surveys (2019).
6. Chauhan, NS., Saxena, A.: A green software development life cycle for cloud computing. IT Professional (2013).
7. Saputri, TR., Lee, S-W.: Integrated Framework for incorporating sustainability design in software engineering life-cycle: An empirical study. Information and Software Technology (2021).
8. Moises, AC., Malucelli, A., Reinehr, S.: Practices of energy consumption for Sustainable Software Engineering. 2018 Ninth International Green and Sustainable Computing Conference (IGSC) (2018).
9. Erdélyi, K.: "Special factors of development of green software supporting eco sustainability," *2013 IEEE 11th International Symposium on Intelligent Systems and Informatics (SISY)*, Subotica, Serbia, pp. 337-340 (2013).
10. Verdecchia, R., Lago, P., Ebert, C., de Vries, C.: "Green IT and Green Software," in *IEEE Software*, Vol. 38, no. 6, pp. 7-15, Nov.-Dec (2021).
11. Yuan, H., Liu, H., Bi, J., Zhou, MC.: Revenue and energy cost-optimized Biobjective task scheduling for Green Cloud Data Centers. IEEE Transactions on Automation Science and Engineering, pp. 817-830 (2021).
12. Fowler, M.: Refactoring. Addison-Wesley Professional, Boston, MA, USA (1999).
13. Şanlıalp, İ., Öztürk, MM., Yiğit, T.: Energy Efficiency Analysis of Code Refactoring Techniques for Green and Sustainable Software in Portable Devices. Electronics 11(3), 442 (2013).
14. Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, JP., Saraiva, J.: Ranking programming languages by energy efficiency. Science of Computer Programming 205, 102609 (2021).
15. Jain, A., Mishra, M., Peddoju, SK., Jain, N.: Energy efficient computing-green cloud computing. In: 2013 international conference on energy efficient technologies for sustainability, pp. 978-982. IEEE, Nagercoil, India (2013).
16. What is cloud computing?, https://aws.amazon.com/what-is-cloud-computing/, last accessed 2023/22/12.

17. Bharany, S., Sharma, S., Khalaf, O.I., Abdulsahib, G.M., Al Humaimeedy, A.S., Aldhyani, T.H., Maashi, M., Alkahtani, H.: A systematic survey on energy-efficient techniques in sustainable cloud computing. Sustainability 14(10), 6256 (2022).
18. Rout, S., Sahoo, K.S., Patra, S.S., Sahoo, B., Puthal, D.: Energy efficiency in software defined networking: A survey. SN Computer Science 2(4), 308 (2021).
19. Singh, S., Jha, R.K.,: A survey on software defined networking: Architecture for next generation network. Journal of Network and Systems Management 25, 321-374 (2017).
20. What is Software-Defined Networking (SDN)?, https://www.vmware.com/topics/glossary/content/software-defined-networking.html, last accessed 2023/22/12.
21. Pinto, G., Castor, F., Liu, Y.D.: Mining questions about software energy consumption. In: *Proceedings of the 11th Working Conference on Mining Software Repositories*, pp. 22-31. Association for Computing Machinery, Hyderabad, India (2014).
22. Mancebo, J., Calero, C., García, F., Moraga, M.Á. and de Guzmán, I.G.R.,: FEETINGS: framework for energy efficiency testing to improve environmental goal of the software. Sustainable Computing: Informatics and Systems 30, 100558 (2021).
23. Karita, L., Mourão, B.C., Machado, I.C.,: Software industry awareness on green and sustainable software engineering: a state-of-the-practice survey, SBES (2019).
24. Groher, I., Weinreich, R.,:An Interview Study on Sustainability Concerns in Software Development Projects, 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (2017).
25. Lago, P., Aklini Kocak, S., Crnkovic, I., Penzensradler, B,: Framing Sustainability as a Property of Software Quality, Communications of the ACM 70-78. (2015).
26. Souza, M.R., Haines, R.,Vigo, M., Jay, C,:What Makes Research Software Sustainable? An Interview Study With Research Software Engineers., 2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE) (2019).
27. Mourão, B.C., Karita, L., Machado, I.C.,: Green and Sustainable Software Engineering - a Systematic Mapping Study, SBQS (2018).
28. Lago, P.; Gu, Q.; Bozzelli, P. ,: A systematic literature review of green software metrics, VU Technical Report (2014).
29. Kumar, A.,: An Empirical Study on Green and Sustainable Software Engineering, 14th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems (SEPADS '15), Vol. 27 (2015)
30. Iravani, A,., Hasan, M., Zohoori, M.,: Advantages and Disadvantages of Green Technology; Goals, Challenges and Strengths , International Journal of Science and Engineering Applications, Vol 6 Issue 09, ISSN-2319-7560 (2017).
31. Applover.com. Pros and cons of green computing – is it worth the cost?, https://applover.com/blog/pros-and-cons-of-green-computing-is-it-worth-the-cost/, last accessed 2023/02/23.
32. Ibrahim, S.R.A., Yahaya, J., Salehudin, H., Deraman, A.,: The Development of Green Software Process Model A Qualitative Design and Pilot Study, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 12, No. 8 (2021).
33. David, O., Ascough II, J.C., Lloyd, W, Green, T.R., Rojas, K.W., Leavesley, G.H.,Ahuja, L.R.,: A software engineering perspective on environmental modelling framework design: The Object Modeling System, Environmental Modelling and Software, Vol. 39, pp 201-213 (2013).
34. Calero, C., Piattini, M.: Introduction to green in software engineering. Green in Software Engineering, pp. 3–27 (2015).
35. Turkin, I., Vykhodets, Y.: Software engineering master's program and Green IT: The design of the software Engineering Sustainability course, pp. 662-666. Kyiv, UKraine (2018).

36. Mohankumar, M., Anand Kumar, M.: A GREEN IT STAR MODEL APPROACH FOR SOFTWARE DEVELOPMENT LIFE CYCLE. International Journal of Advanced Technology in Engineering and Science, Vol. 03, No. 01 (2015).
37. Wolfram, N., Lago, P., Osborne, F.: Sustainability in software engineering. Sustainable Internet and ICT for Sustainability, pp. 1-7. SustainIT, Funchal, Portugal (2017).
38. Kern, E., Guldner, A., Naumann, S.: Including software aspects in green it: How to create awareness for Green Software issues. Green IT Engineering: Social, Business and Industrial Applications, pp. 3–20 (2018).
39. Forti, S., Brogi, A.: Green application placement in the cloud-iot continuum. Practical Aspects of Declarative Languages, pp. 208–217 (2022).
40. Ganesan, M., Kor. A-L., Pattinson, C., Rondeau, E.: Green Cloud Software Engineering for big data processing. Sustainability 12:9255 (2020).
41. Almusawi, SMY., Khalefa, MS.: Study of knowledge management framework to enhance Enterprise Resource Planning system in Green software development process," *International Conference on Communication & Information Technology (ICICT),* pp. 1-6. Basrah, Ira (*2021*).
42. Kern, E., Silva, S., Guldner, A.: "Assessing the sustainability performance of Sustainability Management software", *Technologies*, 6(3), p. 88 (2018).
43. Almusawi, SMY., Khalefa, MS.: Study of knowledge management framework to enhance Enterprise Resource Planning system in Green software development process, pp. 1-6. Basrah, Iraq (2021).
44. Ahmad Ibrahim, SR., Yahaya, J., Sallehudin, H.: Green Software Process Factors: A qualitative study. Sustainability 14:11180 (2022).
45. Abdalkareem, R., Mujahid, S., Shihab, E., Rilling, J.: "Which Commits Can Be CI Skipped?," in *IEEE Transactions on Software Engineering*, Vol. 47, no. 3, pp. 448-463 (2021).
46. Raisian, K., Yahaya, J., Deraman, A.: Current Challenges And Conceptual Model Of Green And Sustainable Software Engineering, Journal of Theoretical and Applied Information Technology, Vol. 94; Issue 2;428-443 (2016).
47. Shahin, M.: "An empirical study of architecting for continuous delivery and deployment," *Empirical Software Engineering*, 24(3), pp. 1061–1108 (2018).
48. Turkin, I., Vykhodets, Y.: Software engineering sustainability education in compliance with industrial standards and green IT concept. Green IT Engineering: Social, Business and Industrial Applications, pp. 579–604 (2018).
49. Garousi, V, Felderer, M. and Mäntylä, M.V., Guidelines for including grey literature and conducting multivocal literature reviews in software engineering, Information and Software Technology, Volume 106, 2019, Pages 101-121. ISSN 0950-5849.
50. Grogan, J., Mulready, C., McDermott, J., Urbanavicius, M., Yilmaz, M., Abgaz, Y., McCarren, A., MacMahon, S.T., Garousi, V., Elger, P. and Clarke, P., 2020. A multivocal literature review of function-as-a-service (faas) infrastructures and implications for software developers. In Systems, Software and Services Process Improvement: 27th European Conference, EuroSPI 2020, Düsseldorf, Germany, September 9–11, 2020, Proceedings 27 (pp. 58-75). Springer International Publishing
51. Messnarz R., Much A., Kreiner C., Biro M., Gorner J. (2017) Need for the Continuous Evolution of Systems Engineering Practices for Modern Vehicle Engineering. In: Stolfa J., Stolfa S., O'Connor R., Messnarz R. (eds) Systems, Software and Services Process Improvement. EuroSPI 2017. Communications in Computer and Information Science, vol 748. Springer, Cham. https://doi.org/10.1007/978-3-319-64218-5_36

52. Jakub Stolfa, Svatopluk Stolfa, Christian Baio, Utimia Madaleno, Petr Dolejsi, Federico Brugnoli, Richard Messnarz, DRIVES—EU blueprint project for the automotive sector—A literature review of drivers of change in automotive industry, in: Journal of Software: Evolution and Process, Volume32, Issue3, Special Issue: Addressing Evolving Requirements Faced by the Software Industry, March 2020
53. Messnarz, R., Ekert, D., Grunert, F., Blume, A. (2019). Cross-Cutting Approach to Integrate Functional and Material Design in a System Architectural Design – Example of an Electric Powertrain. In: Walker, A., O'Connor, R., Messnarz, R. (eds) Systems, Software and Services Process Improvement. EuroSPI 2019. Communications in Computer and Information Science, vol 1060. Springer, Cham. https://doi.org/10.1007/978-3-030-28005-5_25
54. Samer Sameh Makkar, Selina Meza, Razvan Bogdan, Darius Barmayoun, Jakub Stolfa, Svatopluk Stolfa, Marek Spanyik, Richard Messnarz, Ana Toth. *et al.* (2022). Automotive Skills Alliance—From Idea to Example of Sys/SW International Standards Group Implementation. In: Yilmaz, M., Clarke, P., Messnarz, R., Wöran, B. (eds) Systems, Software and Services Process Improvement. EuroSPI 2022. Communications in Computer and Information Science, vol 1646. Springer, Cham. https://doi.org/10.1007/978-3-031-15559-8_9