SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 8, No. 2, 273-285, 2025
DOI: 10.35377/saucis..1589506
RESEARCH ARTICLE

Assessing the Role of Software in Sustainability: A Survey of Industry Practices and Research Trends

Enes Bajrami[1*]
[1] Ss. Cyril and Methodius University in Skopje, Faculty of Computer Science and Engineering, Ruger Boskovik 16, 1000 Skopje, Republic of North Macedonia

Corresponding author:
Enes Bajrami, Ss. Cyril and Methodius University
enes.bajrami@students.finki.ukim.mk

Article History:
Received: 21.11.2024
Revised: 09.04.2025
Accepted: 09.04.2025
Published Online: 17.06.2025

ABSTRACT
The ever-increasing demand for complex software applications has turned the entire ICT resource and energy consumption into a significant environmental concern. Several research studies have concentrated on making hardware sustainable; however, the environmental aspects of software remain underexplored. This paper discusses the contribution of software to Green Computing, with a special emphasis on energy efficiency throughout the software development life cycle. The study has pointed out the guiding principles of Sustainable Software Engineering: the efficiency of energy and resources, sustainable lifecycle management, and design centered on the users. The analysis also reveals that environmental effects, such as carbon footprint and energy consumption, call for targeted software component improvements. The study also examines the hurdles in implementing green software engineering, such as legacy system challenges, regulatory issues, and economic viability. This research integrates insights from climate science, hardware optimization, and software engineering to contribute to developing eco-friendly software systems, thus charting future directions for sustainability in this field.
Keywords: Green Computing, Sustainable Software Engineering, Energy Efficiency, Resource Optimization

1. Introduction

The rising demand for increasingly complex software applications has led to a significant negative environmental impact from Information and Communication Technology (ICT), primarily due to its escalating consumption of resources and energy [1] [2]. Approximately 97% of climate scientists concur that the observed global warming trends over the past century are most likely attributable to human activities [3]. The impact of Information and Communication Technology (ICT) on sustainable development, particularly software, has become a prominent focus in Green Computing. Sustainable development involves utilizing resources to meet human needs while considering the ecological, economic, and societal consequences [4] [5]. Although recent efforts in ICT have sought to develop environmentally efficient solutions, it remains uncertain whether ICT's energy and resource savings will outweigh its overall resource consumption [6] [7]. A considerable body of research on Green ICT has primarily concentrated on environmental sustainability concerning computer hardware. However, addressing the energy consumption issues associated with software is crucial for advancing green computing. Software features contribute to CO_2 emissions just as much as hardware components [8] [9]. Software exerts an indirect environmental impact by managing and operating the underlying hardware. Certain software solutions can optimize resource utilization, while others are designed to be sustainable enough to reduce the need for additional hardware following updates [10]. Unfortunately, a notable lack of models and research focused on software and software development processes exists. Recently, significant efforts have been made to develop green software. Some initiatives aim to create sustainable software, while others design software development processes that guide stakeholders in producing environmentally friendly software products [11]. Additional efforts focus on developing tools that measure the environmental impact of software and the energy efficiency of application development environments [12]. There is also an emphasis on enhancing operating systems to better manage applications' power consumption [13]. The software dimensions of Green IT have not been extensively explored due to its intangible nature and its indirect impact on the environment [14]. However, researchers are increasingly recognizing software's direct and indirect environmental impacts. Energy efficiency and sustainability throughout the software life cycle are essential, though these factors have traditionally been overlooked in conventional software development processes [15]. The main challenges in integrating sustainability into software development include the unclear extent of software's contribution to overall hardware energy consumption, the uncertain role of software engineering in promoting sustainability, and the lack of a clear conceptual foundation [16]. Artificial intelligence (AI) and machine learning (ML) have gained attention for their potential to enhance sustainability efforts, particularly in optimizing energy consumption and improving software efficiency [17]. However, the increasing complexity of AI models, including large language models (LLMs), has

Cite as: E. Bajrami. (2025). Assessing the role of software in sustainability: A survey of industry practices and research trends. *Sakarya University Journal of Computer and Information Sciences*. vol. 8, no. 2, pp. 273-285, 2025 doi: 10.35377/saucis...1589506

also raised concerns about their energy consumption and carbon footprint [18]. While AI-powered solutions hold promise for green software development, this study focuses on existing software tools used in industrial settings for sustainability. Future work could explore how AI-driven optimization techniques contribute to improving energy efficiency in industrial applications [19] [20].

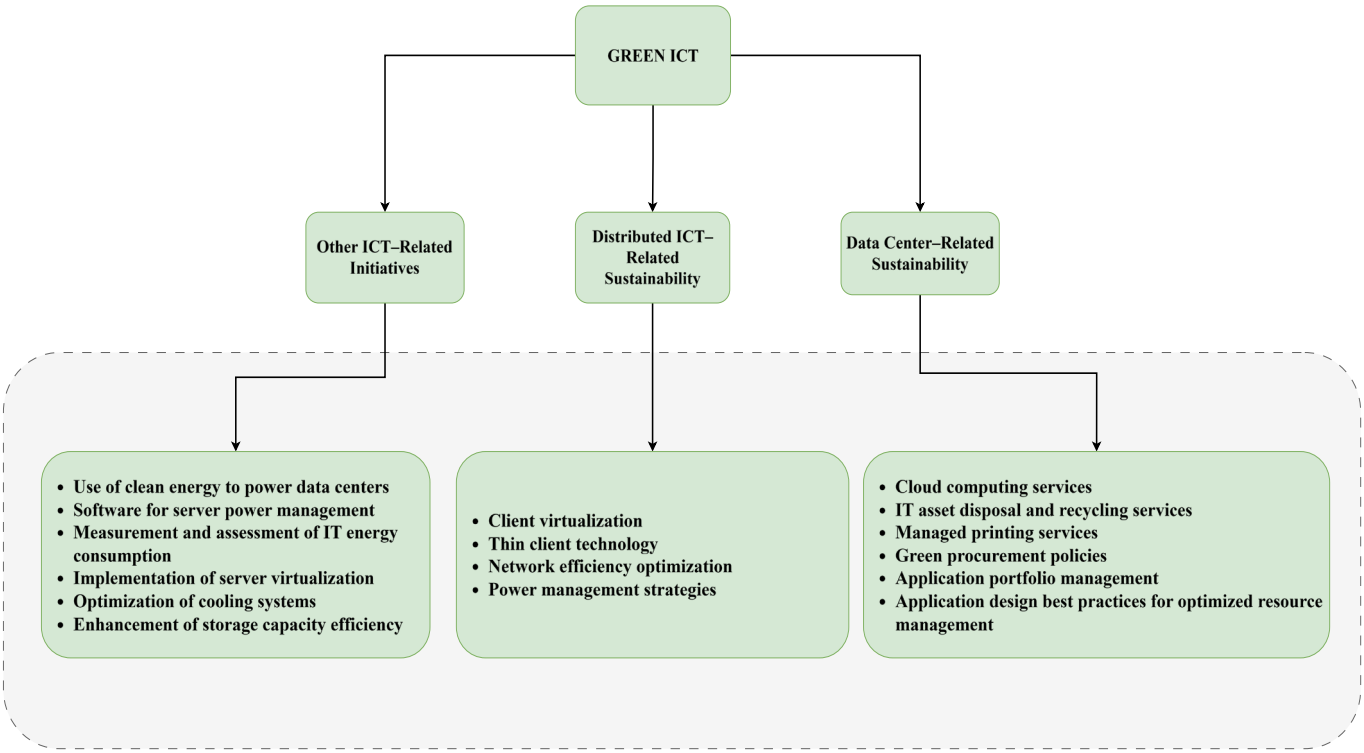

Figure 1: Taxonomy of Green ICT

Figure 1 presents a structured overview of Green ICT practices, categorized into three core domains: Data Center–Related Sustainability, Distributed ICT–Related Sustainability, and Other ICT–Related Initiatives. The author developed this taxonomy to conceptually organize key sustainability actions across different layers of ICT systems. In the first category, Data Center–Related Sustainability includes using clean energy, server power management, virtualization, improved cooling mechanisms, and enhanced storage capacity efficiency, all aimed at reducing the environmental impact of large-scale computing infrastructures. Distributed Sustainability focuses on client-side practices, such as client virtualization, thin client technology, network efficiency optimization, and power management strategies, which contribute to minimizing energy consumption at the end-user level. The final category, Other ICT–Related Initiatives, addresses broader organizational measures such as cloud computing services, IT asset recycling, managed printing, green procurement, application portfolio management, and sustainable application design. These elements provide a comprehensive foundation for understanding how sustainability is integrated into ICT environments.

2. Literature Review

Several studies have been carried out to collect, analyze, and assess the evidence available in Green Software Engineering. These investigations focus on identifying effective strategies and approaches that enhance software systems' sustainability, underscoring software's critical role in reducing environmental impact. In [21], the author explores the principles and practices of green software engineering, emphasizing its environmental impact and identifying best practices in the field. The study specifically focuses on energy efficiency, resource optimization, and eco-friendly development methodologies, all aimed at reducing the carbon footprint of software engineering. Additionally, the paper addresses the challenges and regulatory constraints associated with implementing sustainable practices. The research underscores the importance of a collective commitment to energy efficiency and sustainability within the software engineering community. In [22], a systematic literature review (SLR) was conducted to map the state-of-the-art in Sustainable Software Engineering (SE), focusing on existing models, guidelines, practices, and related proposals. However, this review was limited to 36 works published until 2010 and addressed only three research questions. The study explored the most frequently cited guidelines and models in Sustainable SE, tracked the evolution of interest in the field, and identified key authors and venues contributing to this area of research. In [23], a review study was conducted on green computing, focusing on five key areas: software

engineering, cloud computing, mobile computing, data centers, and the educational sector. The study performed a systematic literature review for each area, detailing current research trends and limitations. However, it lacked discussion on using datasets, their characteristics, testing mechanisms, and their contributions to various perspectives on current technology applications within these fields. Since green computing is closely linked with multiple domains, exploring these associations is crucial for developing environmentally friendly modern computing systems. In [24], a comprehensive review was conducted on energy management techniques to achieve green IoT. The study presents the current challenges in green IoT related to energy consumption and examines various approaches utilized by different studies for energy management. It offers a thorough overview of recent energy management systems within the IoT ecosystem. It outlines current research trends and future perspectives for energy management in the context of green IoT. Reviewing these implementation studies provides valuable insights, identifies existing challenges, and suggests potential directions for future research, not only within green computing but also across other related fields. In [25], the authors discuss the growing significance of software in the twenty-first century, emphasizing its indirect impact on hardware energy consumption and carbon emissions. While previous studies have predominantly focused on models and tools for measuring power consumption and energy efficiency from a hardware perspective, less attention has been given to the role of software development in energy optimization. The study highlights that energy consumption can be reduced by implementing green software practices throughout all phases of the development lifecycle. However, existing green software process models primarily concentrate on environmental and economic aspects without adequately integrating waste management in the development phase. To address this gap, a qualitative study was conducted involving interviews with eight informants from Malaysia's public and private sectors. The study aimed to (i) examine the current industry practices in green software processes, (ii) identify waste elements in software development, and (iii) determine key green factors influencing the software process. Thematic analysis was conducted using Atlas. ti 8 revealed three key themes: best practices in software processes, nine categories of software waste (e.g., building the wrong feature, rework, unnecessary complexity, cognitive overload, psychological distress, waiting, knowledge loss, ineffective communication, and delays), and six green factors (resources, people, organizational aspects, technical aspects, environmental concerns, and technology). The findings indicate that integrating best practices, green methodologies, and software technologies at every stage of development is crucial for achieving a sustainable and environmentally friendly software process. The study underscores the role of advancing computing technologies in maintaining a continuously updated and green software development framework. In [26], the authors explore the emerging research area of sustainability in Software Engineering, highlighting that while sustainability has been widely discussed in academia, it remains underrepresented in the software industry. The study emphasizes that incorporating sustainability into software design and development can provide significant societal benefits, but this requires increased awareness and knowledge among software professionals. To address this gap, the research examines sustainability knowledge, its perceived importance, and industry support from the perspective of South Asian software professionals. The study investigates key questions such as how professional software developers perceive sustainability, how the software industry identifies sustainability requirements, and how developers incorporate sustainability parameters during software development. A survey was conducted among 221 industry practitioners working on software projects in banking, finance, and management applications. The results indicate that while 91% of practitioners recognize the importance of sustainability, a substantial knowledge gap exists in its practical application. Notably, 48% of professionals misinterpret "Green software" as "sustainable software." Moreover, the technical aspect of sustainability is regarded as the most important factor by 67% of professionals and 77% of companies. A critical finding of the study is that 92% of software practitioners cannot identify sustainability requirements for software applications. The study contributes by proposing sustainability guidelines for specific software applications and a catalog to assist in identifying sustainability requirements. The findings provide an initial perspective on how sustainability is understood and addressed within the South Asian software industry. In [27], the authors highlight the critical role of the electronics sector in modern industry, emphasizing its contribution to clean technology, dry processes, and efficient design, which align with Industry 4.0 and sustainability principles. However, the rapid obsolescence of electronic devices has led to a significant increase in electronic waste. To mitigate this issue, the study proposes a novel edge computing structure, the AIFC, which operates independently of specific systems and leverages existing computing infrastructures. The AIFC is built on an enterprise service bus (ESB) and implemented using decentralized microservices, reducing reliance on conventional cloud computing models. The study adopts an action research approach involving collaboration between researchers and industry practitioners and tests the proposed structure in six different scenarios. These scenarios simulate small and medium-sized enterprise (SME) environments and encompass various stages, including proof of concept, prototyping, minimum viable product, scalability, and a roadmap for implementation. The developed microservices facilitate data filtering, processing, storage, querying, and sensor data acquisition while maintaining low latency and, in some cases, improving performance compared to traditional cloud-based architectures. Furthermore, the findings demonstrate that the approach eliminates the need for hardware or communication structure upgrades—key contributors to electronic waste and rapid obsolescence. Following the AIFC development process, the study presents a sustainable roadmap supporting Industry 4.0 initiatives and SME digital transformation efforts.

3. Methodology

3.1 Methodology of Research

This study employs a mixed-method approach, combining a literature-based analysis with a questionnaire-based survey to explore green software engineering practices and their adoption. While this study does not follow a Systematic Literature Review (SLR) methodology, a structured approach was applied to ensure a comprehensive and relevant literature search. The literature review was conducted by searching for peer-reviewed journal articles, industry reports, and case studies in IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect, and Google Scholar databases. Search terms included "Green Software," "Sustainable Computing," "Software Energy Efficiency," and "Eco-Friendly Software Engineering." Studies were selected based on their relevance to sustainability in software engineering, with priority given to recent publications from 2015 to 2024. This review provided the foundation for understanding key themes, methodologies, and trends in the field. The survey-based study was designed to complement the findings of the literature by assessing how green software practices are adopted in industrial settings. The structured questionnaire included 15 Yes/No questions focusing on energy optimization, predictive maintenance, resource management, and sustainability metrics. A pilot test was conducted with five factory managers to ensure clarity and relevance.

3.2 Data Collection

The study incorporates both qualitative and quantitative data sources. Qualitative data was gathered from peer-reviewed journals, conference proceedings, and industry reports better to understand green software engineering practices and their practical implications. The questionnaire-based survey was conducted to gather quantitative data on green software adoption across different industrial sectors. The survey targeted six types of factories in the Polog[1] region, including refrigerator, railing, door and window, candle, oil, and outdoor tile factories, chosen for their operational diversity. Since I live in the Polog region, this location was selected due to its accessibility. It allowed direct communication with factory managers and ensured a higher response rate and more reliable data collection. The survey responses were collected from 6 factories using stratified random sampling to ensure a balanced representation across industries. Data was collected through in-person interviews and online survey forms, ensuring broad participation. Python was used to analyze the responses and to process and visualize the data. Figures 3–6 were generated using Python-based tools such as Matplotlib and Seaborn, providing graphical insights into green software adoption trends, industry-wise variations, and key challenges in sustainability implementation. The combination of literature analysis and survey responses provides a holistic view of sustainability practices in software engineering. This mixed-method approach ensures that the study captures theoretical insights and practical realities, highlighting the factors influencing the adoption of green practices and offering actionable recommendations for future research and policy development.

4. The Principles of Sustainable Software Engineering

Sustainable Software Engineering is a developing field that blends climate science with software, hardware, energy markets, and data center design [28]. It encompasses a fundamental set of skills required to create, develop, and operate software applications in an environmentally responsible manner. A primary principle in green software engineering is energy efficiency [29]. This principle centers on optimizing software to minimize energy consumption during its operation. Electricity usage is decreased by making software more energy-efficient, and hardware longevity is enhanced due to reduced strain [30] [31]. Achieving energy efficiency involves employing techniques such as optimizing code, utilizing efficient data structures, and implementing algorithms that lower computational demands. Resource efficiency is another crucial principle, focusing on efficiently using computing resources like CPU and memory. Resource-efficient software operates effectively within the limitations of the hardware it runs on, avoiding unnecessary resource use [32]. This approach reduces the environmental impact and reduces companies' operational expenses. Load balancing, virtualization, and efficient resource management are key to enhancing resource efficiency [33]. Managing the software lifecycle sustainably is also a key principle. This entails considering the environmental impact of software from its design and development phases to deployment, maintenance, and eventual decommissioning. Sustainable lifecycle management advocates using renewable energy in data centers, adopting energy-efficient hardware, and recycling electronic waste [34]. Moreover, it promotes the creation of software that can be easily maintained and updated, prolonging its lifespan and lessening the need for frequent replacements. The principle of scalability is significant in ensuring that software can grow to handle more workload without a corresponding increase in resource use. Scalable software can efficiently manage more users or data without substantially increasing energy use [35]. Cloud computing and distributed systems are often utilized to achieve scalable, sustainable software. User-centered design is another important aspect. Software should be designed to focus on essential functionalities, ensuring it meets user needs without including superfluous features that might increase energy consumption [36]. By creating simple user interfaces and focusing on necessary features, developers can produce energy-efficient and user-friendly software, contributing to the software's overall sustainability [37]. Finally, sustainability metrics and reporting are vital for green software engineering. Developers and organizations must measure and report on the environmental impacts of their

[1] The Polog Statistical Region is one of eight statistical regions of the Republic of North Macedonia. Polog, located in the northwestern part of the country, borders Albania and Kosovo. Internally, it borders the Southwestern and Skopje statistical regions.

software [38]. Standard metrics like carbon footprint and energy usage help organizations assess and improve their software's sustainability. Regular reporting and transparency in these practices can bolster a company's reputation and commitment to sustainability [39].

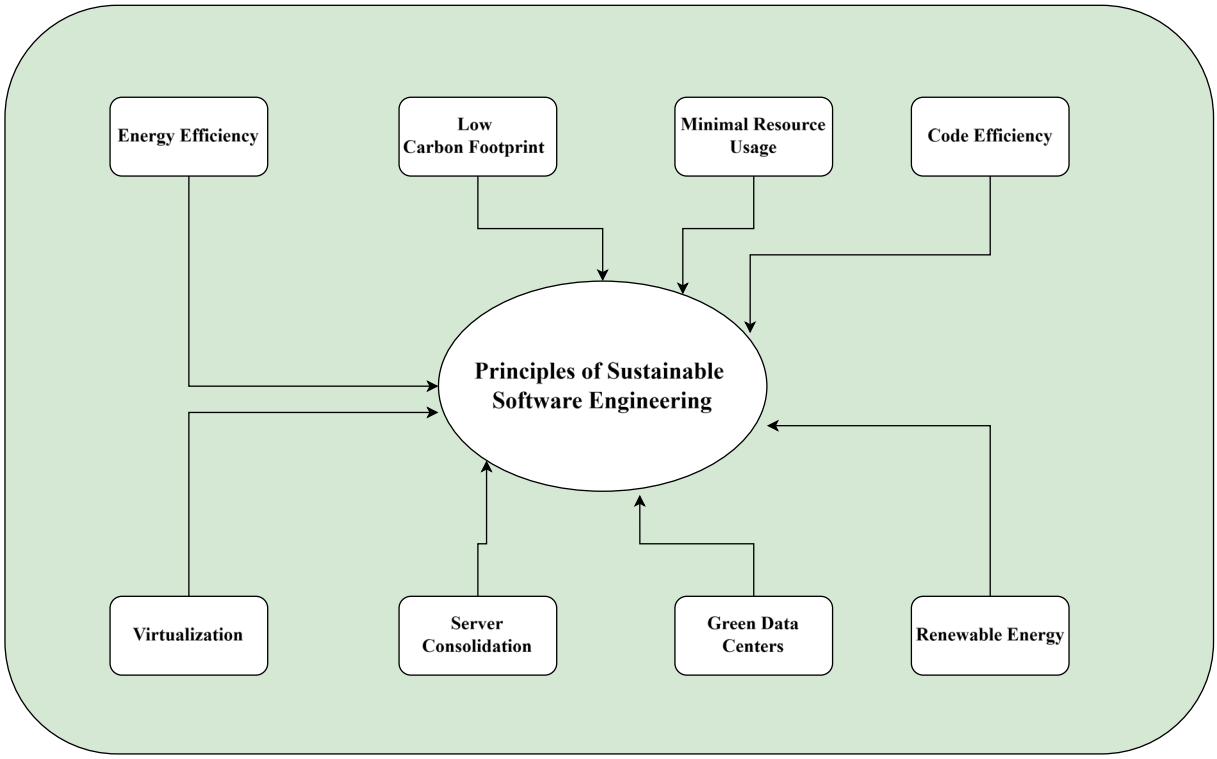

Figure 2: Green Software Engineering Principles

Figure 2 illustrates a visual summary of the fundamental principles of sustainable software engineering inspired by the work of Sivkumar Mishra and Namita Dehury [40] and redrawn by the author to align with the focus of this study. The diagram presents eight essential principles that guide the development of environmentally responsible software systems. These include energy efficiency, which involves reducing power consumption during software execution; low carbon footprint, which refers to minimizing greenhouse gas emissions linked to computing infrastructure; and minimal resource usage, which focuses on optimizing the consumption of processing power, memory, and storage. Code efficiency highlights the importance of writing optimized and maintainable software. Virtualization promotes scalability and resource sharing while reducing dependency on physical hardware. Server consolidation aims to streamline workloads onto fewer servers to enhance resource utilization. Green data centers support the transition to sustainable infrastructure through energy-efficient systems and clean power. Finally, renewable energy promotes the shift toward powering software systems using sustainable energy sources. These principles represent a comprehensive foundation for integrating sustainability into software engineering practices.

5. Evaluation of Environmental Effects

Environmental Impact Assessment (EIA) evaluates the potential effects of a project or policy on the environment. It aims to identify, predict, and mitigate adverse impacts, ensuring informed decision-making and promoting sustainable development by integrating environmental considerations into planning processes [41].

5.1 Measuring Carbon Footprints

This involves assessing the total carbon emissions produced during the Software Development Life Cycle [21]. Doing so helps evaluate the software's environmental impact and identify strategies to mitigate carbon footprints. Carbon footprints are typically quantified in CO2 equivalents, determined by energy usage, hardware, data centers, and user behaviors [42].

5.2 Analyzing Energy Consumption in Software Systems

This is the most crucial phase of the software development life cycle. Detailed observation and monitoring are conducted to reveal energy usage patterns and traits [21]. This process identifies which software components and processes consume the most energy, highlighting areas for optimization. Developers can then focus on creating more efficient algorithms, promoting environmentally friendly software development [43].

5.3 Identifying Hotspots for Improvement

The next step involves pinpointing the most energy-consuming components. After identifying these, we can focus on hotspots where environmental improvements will have the greatest impact. Hotspots vary depending on power consumption and the database resources they use. Once identified, developers can more effectively target these areas for improvement [21].

6. Methods and Utilities

The Methods and Utilities chapter delves into the foundational resources and strategies for sustainable software development. It begins with an overview of the key concepts and examines energy-efficient programming languages and frameworks. The chapter also explores sustainability-focused software development methodologies, comprehensively understanding the tools and techniques supporting eco-friendly software engineering practices.

6.1 Overview

Green software engineering tools are designed to assist developers in creating more energy-efficient and environmentally friendly software. These tools are specifically developed to measure, analyze, and optimize the environmental impacts of software. They enable efficient use of resources and come in various forms, including energy profiling tools, sustainability assessment platforms, resource optimization software, and green code analyzers [44]. A notable example is Google's commitment to sustainable software development, where they utilize machine learning for optimizing data center cooling and resources, leading to significant energy savings and reduced greenhouse gas emissions. Similarly, Meta's Open Compute Project (OCP) has achieved considerable energy efficiency and lowered its carbon footprint in data center operations [45].

6.2 Energy-Efficient Programming Languages and Framework

These frameworks are created to reduce energy and resource consumption, aiding developers in producing software that requires less energy [21]. Key features of energy-efficient languages include optimized use of resources, minimal runtime overhead, avoidance of unnecessary computations, efficient memory management, and simplified algorithmic complexity [46].

6.3 Sustainability-Focused Software Development Methodologies

A security-focused software development approach integrates environmental and optimization considerations, ensuring that software meets both functional requirements and environmental standards [21]. It encompasses the entire software lifecycle and emphasizes eco-design to reduce waste and energy use. Microsoft and Salesforce exemplify this methodology, with both companies achieving significant energy savings and reduced carbon emissions by prioritizing renewable energy, efficient coding, and sustainable data center practices [47].

7. Barriers and Challenges

7.1 Challenges in Implementing Green Software Engineering

The primary challenge in adopting green software engineering lies in the reliance on legacy systems, which are often difficult to adapt to eco-friendly practices. Additionally, successful implementation requires significant awareness and training within organizations [21].

7.2 Regulatory and Policy Challenges

Environmental regulations vary by region and impact software development by addressing issues like energy consumption, e-waste management, and emissions. Compliance with these regulations is essential for green software engineering [48]. This includes adhering to standards for energy efficiency, participating in eco-labeling programs, and aligning with sustainability initiatives. Furthermore, regulations may mandate using renewable energy and proper e-waste disposal, influencing how organizations manage data centers and infrastructure. Compliance costs must be considered, as well as potential tax incentives for adopting sustainable practices [49].

7.3 Economic Factors and Cost Efficiency

Economic viability is crucial for green software engineering practices. Organizations must evaluate the total cost of ownership, including development, operational expenses, and potential savings from energy efficiency. Optimizing resource utilization can reduce costs, and while compliance with environmental regulations may incur costs, non-compliance can result in fines. Transitioning to renewable energy may require upfront investment but can offer long-term savings [50]. Evaluating the return on investment (ROI) involves balancing these costs with benefits such as energy savings and enhanced brand reputation. Economic assessments should also factor in risks like fluctuating energy prices and the positive impact of sustainability on employee productivity. Achieving both sustainability and cost-effectiveness is essential for the long-term success of green software engineering initiatives [21] [51].

8. Questionnaire-Based Assessment of Sustainability Practices

The following section presents the findings of a comprehensive questionnaire distributed to six distinct types of factories in the Polog region. The survey aimed to evaluate the adoption of green software engineering practices and sustainability-focused solutions within these factories. Spanning 15 critical questions, the questionnaire explored various dimensions of sustainable software usage, such as energy optimization, predictive maintenance, resource management, and lifecycle analysis. The responses were analyzed and visualized using multiple charts to provide clear insights into adoption trends, factory-specific patterns, and overall engagement with sustainability practices. These visualizations highlight the strengths and gaps in the current implementation of green technologies, serving as a basis for further discussions and recommendations. This section includes four charts that collectively summarize the results, offering a mix of aggregated and detailed perspectives on the data. Each chart sheds light on a specific survey aspect, enabling a deeper understanding of how factories integrate (or fail to integrate) sustainable software practices in their operations. Figures 3, 4, 5, and 6 were independently developed by the author based on the quantitative analysis conducted in this study and serve as original visual representations of the survey findings.

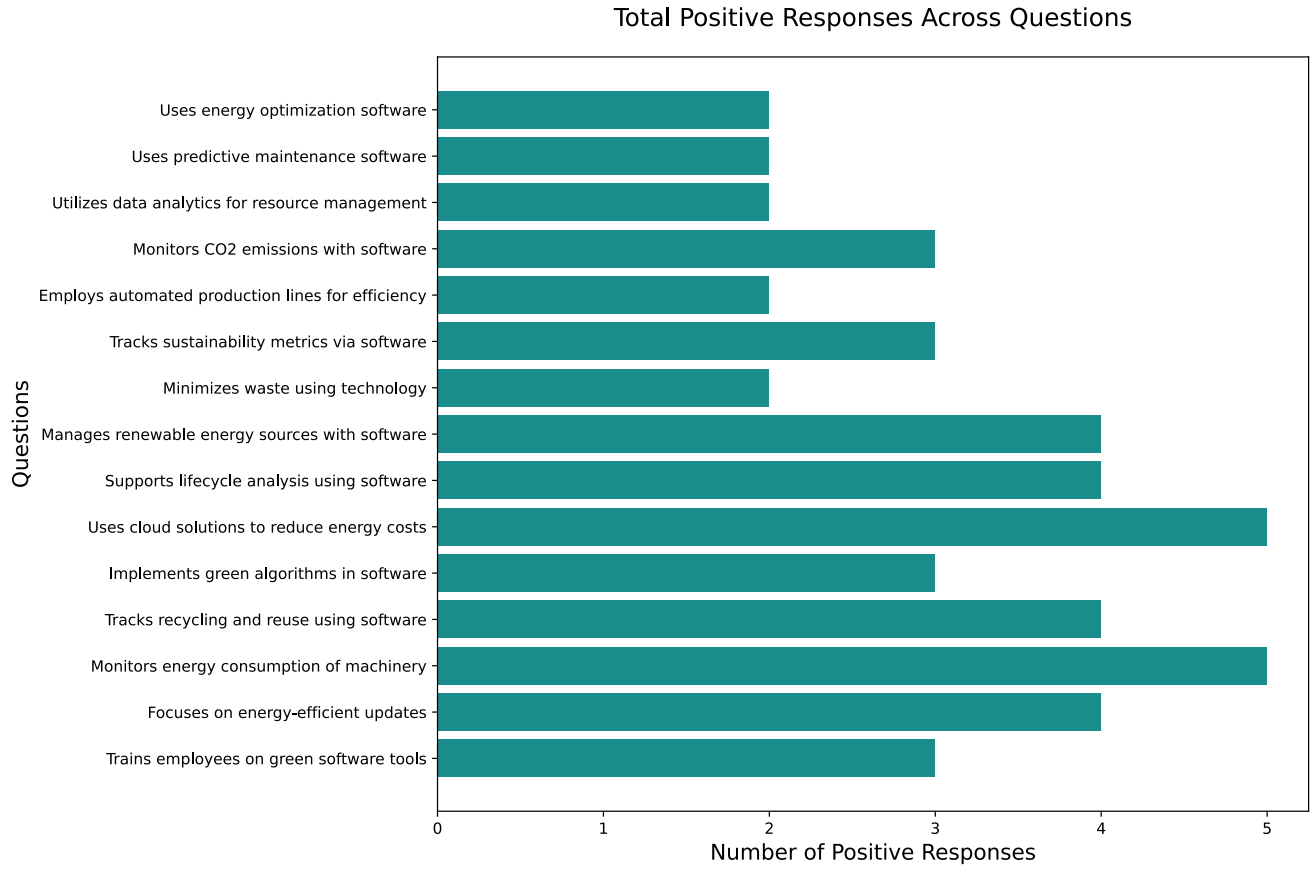

Figure 3: Total Positive Responses Across Questions

This chart visually represents the total number of positive responses received for each question across all factories. The questions are displayed along the y-axis, while the x-axis quantifies the number of positive responses. The chart highlights which areas of green software engineering practices (e.g., energy optimization or CO2 monitoring) are most commonly adopted by the factories overall.

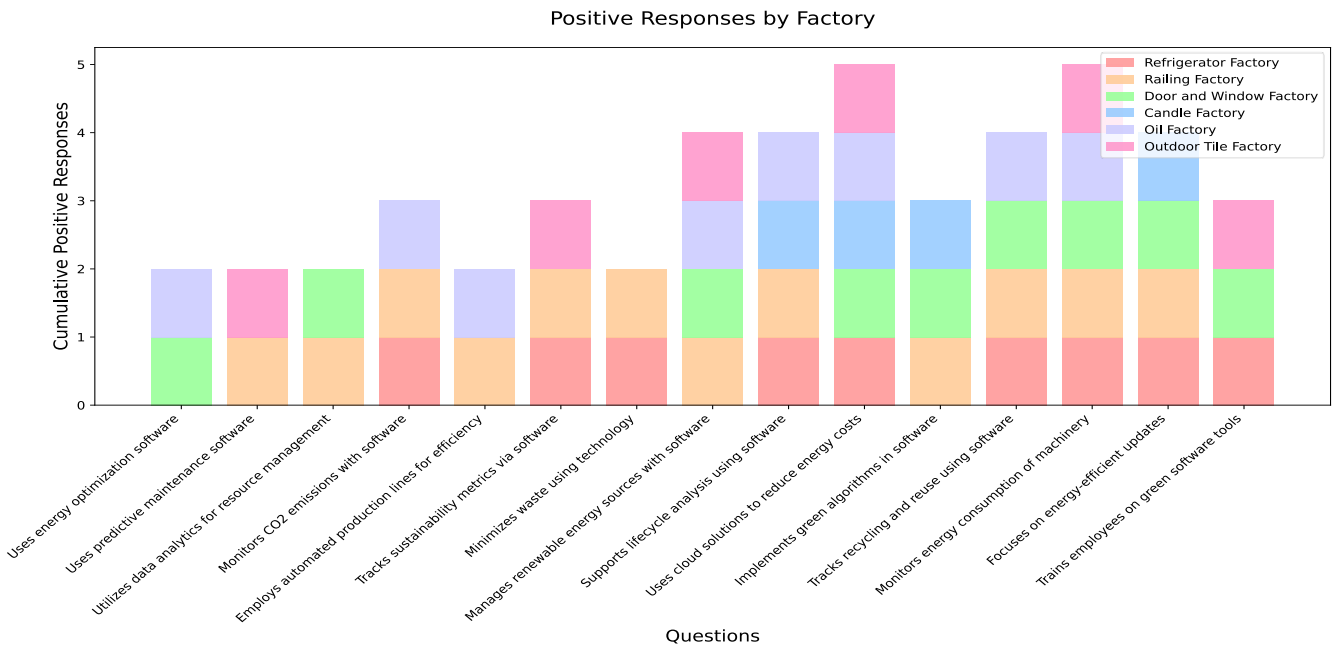

Figure 4: Positive Responses by Factory

This chart breaks down positive responses for each question by factory type. Each bar is segmented into colored sections, where each section represents the contribution of a specific factory type (e.g., Refrigerator Factory, Railing Factory). The chart allows a comparison of how different factory types adopt green software practices across various aspects of sustainability.

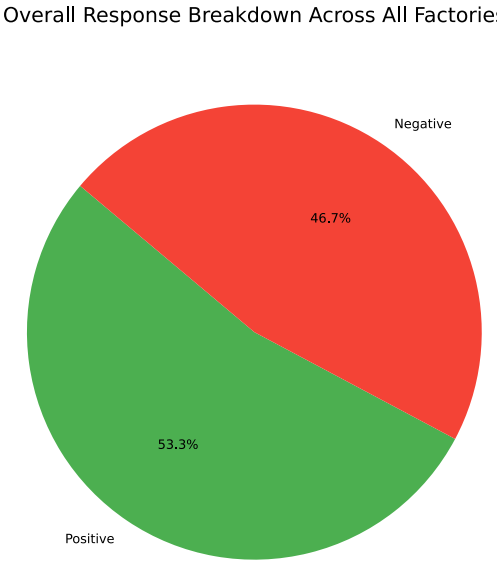

Figure 5: Overall Response Breakdown Across All Factories

This consolidated pie chart combines all responses across all factories and questions into positive and negative categories. The proportions of the two sections reveal the overall state of adoption of sustainability. Positive responses (green) reflect areas where factories have implemented green software engineering practices, while negative responses (red) indicate a lack of adoption.

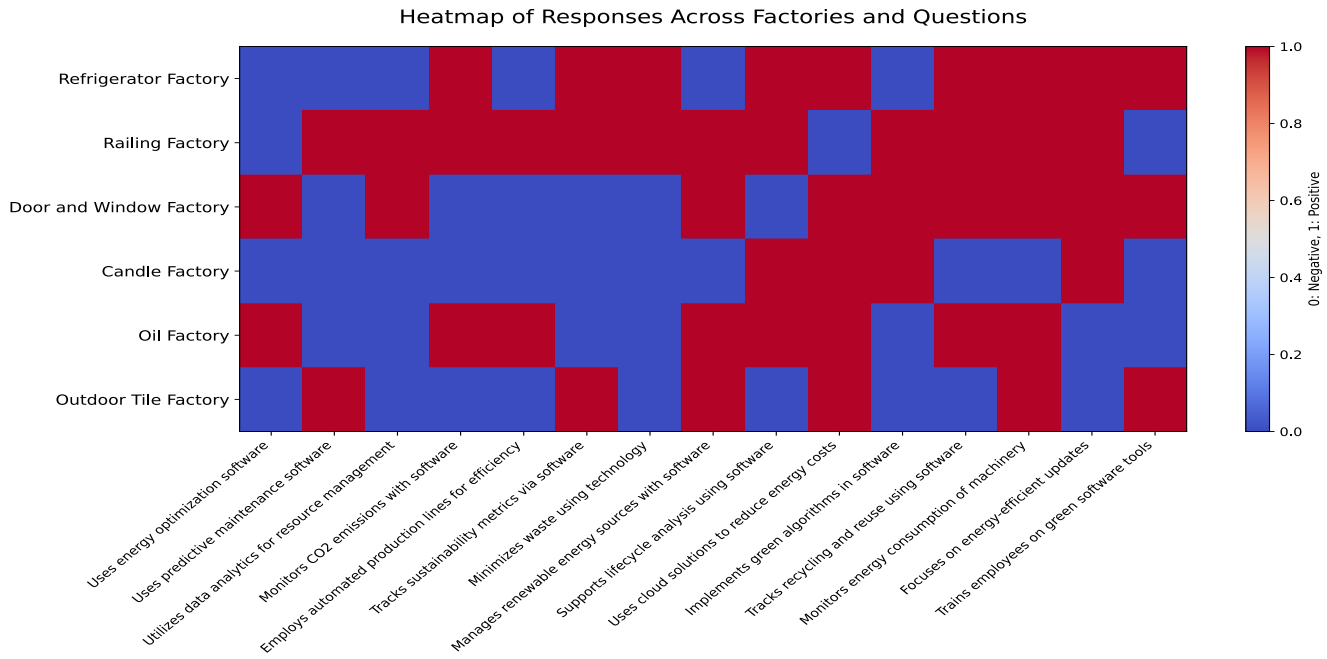

Figure 6: Responses Across Factories and Questions

This chart visualizes the response dataset in a heatmap format, where each cell represents a factory's response to a specific question. Darker shades represent negative responses (0), while lighter shades represent positive responses (1). Rows correspond to factory types, and columns correspond to questions, allowing readers to identify patterns or trends in the data at a glance.

9. Discussion

The study's findings demonstrate substantial deficiencies in implementing green software engineering methods among the examined manufacturers, indicating a widespread absence of sophisticated sustainability initiatives. Although fundamental techniques like predictive maintenance and resource management exhibited somewhat greater adoption rates, sophisticated technology such as CO2 monitoring, energy-efficient upgrades, and green algorithms were predominantly overlooked. Refrigerator, Door, and Window manufacturers showed somewhat superior engagement with specific green practices, possibly attributable to greater operational complexity and the necessity for efficiency. In contrast, Candle and Railing companies persistently fell behind in adopting sustainable software solutions. The findings highlight significant obstacles hindering advancement, such as insufficient awareness and training regarding green technologies, economic limitations that deter smaller factories from investing in these practices, and ineffective regulatory frameworks that do not enforce sustainability standards. These issues underscore the necessity for specific initiatives, including government subsidies, accessible training programs, and the creation of affordable, user-friendly solutions designed for small and medium enterprises (SMEs). The results underscore the necessity of cooperative initiatives among policymakers, industry stakeholders, and software developers to address existing deficiencies and facilitate the extensive implementation of sustainable software practices. The region may progress towards a more environmentally sustainable industrial framework by overcoming these obstacles while improving operational efficiency and cost-effectiveness. A significant finding from the survey indicates that factories employ various bespoke Enterprise Resource Planning (ERP) systems to facilitate their sustainability initiatives; however, these software solutions are tailored to individual factories, created either internally or by third-party vendors, and frequently do not possess a commercial designation. Although tailored to specific needs, these ERP-based systems possess shared features, including resource planning, inventory management, and production optimization, and display analogous user interfaces (UI) across many factories. These ERP solutions function as the principal digital framework for operational management, yet they are constrained in their ability to incorporate sophisticated green software functionalities. Moreover, companies utilize predictive maintenance technologies to enhance machinery performance and minimize energy waste, alongside monitoring systems to assess electricity consumption and carbon footprint. Nevertheless, none of the presently utilized software solutions are driven by artificial intelligence. Numerous industry participants indicated a desire to implement AI-based solutions imminently, especially for optimizing energy efficiency and automating sustainability reporting. This indicates that although custom ERP-based sustainability software is prevalent, there is an increasing push toward AI-driven optimization solutions to improve sustainable operations further. The findings suggest that current sustainability software utilization is operational yet stagnant, as most manufacturers employ rudimentary ERP and monitoring systems devoid of AI-driven analytics or automation capabilities. In light of the growing emphasis on energy efficiency and legal changes favoring environmentally sustainable industrial practices, implementing AI-enhanced sustainability software could represent a pivotal advancement for these sectors. Future research and development should

incorporate AI-driven optimization features into current ERP systems, facilitating the automation of sustainability reporting, improving energy efficiency, and optimizing resource utilization at a more sophisticated level.

10. Conclusion and Future Work

10.1 Conclusion

This study underscores the critical importance of incorporating sustainability into software engineering practices, highlighting that software's environmental impact can be as significant as that of hardware. This research provides a comprehensive understanding of green software engineering principles, adoption trends, and barriers by analyzing literature and real-world survey data. The findings reveal that while some basic green practices, such as predictive maintenance and energy optimization, show moderate adoption, advanced sustainability practices, including CO2 monitoring and green algorithms, remain largely neglected across industries. The questionnaire data provided insights into specific factory types, demonstrating variations in green software adoption. For example, Refrigerator and Door and Window factories displayed higher adoption rates than Candle and Railing factories, which lagged significantly. These results emphasize industries' critical challenges, including economic constraints, lack of awareness, and insufficient regulatory enforcement. Achieving sustainability in software engineering requires technical innovation and collaborative efforts from policymakers, industry stakeholders, and the software community to prioritize eco-friendly practices and develop actionable frameworks. This research highlights the need for a multi-faceted approach that includes education, accessible technologies, and supportive policies to drive the adoption of green practices across industries.

10.2 Future Work

Building on the insights gained from this study, future research should aim to bridge the gaps identified in the literature and the survey data. The next research phase will focus on developing specific research questions (RQs) that address the challenges and opportunities associated with sustainable software engineering. A systematic literature review will explore innovative methodologies and emerging technologies that enhance the sustainability of software systems. In particular, future studies should investigate the role of artificial intelligence (AI) and machine learning (ML) in optimizing energy and resource efficiency. These technologies hold immense potential for automating energy management, predicting resource consumption patterns, and improving the environmental impact of software systems. Additionally, longitudinal studies across diverse industry sectors could provide deeper insights into the evolving adoption of green software practices. Exploring the integration of green engineering standards into software development lifecycles and assessing the effectiveness of government incentives could further enhance our understanding of how to overcome barriers to adoption. Finally, creating open-source sustainability tools and frameworks tailored for small and medium enterprises (SMEs) could empower more organizations to adopt environmentally responsible software development practices, ensuring scalability and long-term impact.

References

[1] Stefan Naumann; Markus Dick; Eva Kern; Timo Johann, "The GREENSOFT Model: A reference model for green and sustainable software and its engineering," *Sustainable Computing: Informatics and Systems,* vol. 1, no. 4, pp. 294-304, 2011.

[2] M. Mahaux and C. Canon, "Integrating the Complexity of Sustainability in Requirements Engineering," *1st international workshop on Requirements for Sustainable Systems,* pp. 1-5, 2012.

[3] John Cook; Naomi Oreskes; Peter T Doran; William R L Anderegg; Bart Verheggen; Ed W Maibach; J Stuart Carlton; Stephan Lewandowsky; Andrew G Skuce; Sarah A Green; Dana Nuccitelli; Peter Jacobs; Mark Richardson; Bärbel Winkler; Rob Painting; Ken Rice, "Consensus on consensus: a synthesis of consensus estimates on human-caused global warming," *Environmental Research Letters,* vol. 11, no. 4, pp. 1-8, 2016.

[4] A. Govindasamy and S. Joseph, "Optimization of Operating Systems towards Green Computing," *International Journal of Combinatorial Optimization Problems and Informatics,* vol. 2, no. 3, pp. 39-51, 2011.

[5] Chia-Tien Dan Lo; Kai Qian, "Green Computing Methodology for Next Generation Computing Scientists," *IEEE Annual International Computer Software and Applications Conference (COMPSAC),* pp. 250-251, 2010.

[6] S. Wang, H. Chen and W. Shi, "SPAN: A software power analyzer for multicore computer systems," *Sustainable Computing: Informatics and Systems,* vol. 1, no. 1, pp. 23-34, 2011.

[7] A. Noureddine; A. Bourdon; R. Rouvoy and L. Seinturier, "A preliminary study of the impact of software engineering on GreenIT," *IEEE First International Workshop on Green and Sustainable Software,* pp. 21-27, 2012.

[8] S. Agarwal, N. Asoke and C. Dipayan, "Sustainable Approaches and Good Practices in Green Software Engineering," *International Journal of Research and Reviews in Computer Science (IJRRCS),* vol. 3, no. 1, pp. 1425-1428, 2012.

[9] S. Bhattacharya, K. Gopinath, K. Rajamani and M. Gupta, "Software Bloat and Wasted Joules: Is Modularity a Hurdle to Green Software?," *IEEE Computer,* vol. 44, no. 9, pp. 97-101, 2011.

[10] N. Amsel, Z. Ibrahim, A. Malik and B. Tomlinson, "Toward sustainable software engineering: NIER track," *IEEE 33rd International Conference on Software Engineering (ICSE),* pp. 976-979, 2011.

[11] F. Albertao, J. Xiao, C. Tian, Y. Lu, K. Q. Zhang and C. Liu, "Measuring the sustainability performance of software project," *IEEE 7th International Conference on e-Business Engineering (ICEBE),* pp. 369-373, 2010.

[12] Sara S. Mahmoud and Imtiaz Ahmad, "A Green Model for Sustainable Software Engineering," *International Journal of Software Engineering and Its Applications,* vol. 4, no. 4, pp. 55-75, 2013.

[13] E. Capra, C. Francalanci and S. A. Slaughter, "Is software "green"? Application development environments and energy efficiency in open source applications," *Information and Software Technology,* vol. 54, no. 1, pp. 60-71, 2012.

[14] H. S. Zhu, C. Lin, and Y. D. Liu, "A programming model for sustainable software," *In Proceedings of the 37th International Conference on Software Engineering - IEEE,* vol. 1, pp. 767-777, 2015.

[15] Y. Zhu and V. J. Reddi, "Greenweb: language extensions for energy-efficient mobile web computing," *In Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation (ACM),* pp. 145-160, 2016.

[16] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," *In Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, EASE,* pp. 68-77, 2008.

[17] Emma Strubell, Ananya Ganesh, Andrew McCallum, "Energy and Policy Considerations for Deep Learning in NLP," *57th Annual Meeting of the Association for Computational Linguistics (ACL). Florence, Italy,* vol. 34, no. 9, pp. 13693-13696, 2019.

[18] Patterson D, Gonzalez J, Le Q, Liang C, Munguia LM, Rothchild D, So D, Texier M, Dean J., "Carbon emissions and large neural network training," 2021.

[19] Anthony, Lasse & Kanding, Benjamin & Selvan, Raghavendra., "Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models," 2020.

[20] Rolnick, D., et al., "Tackling Climate Change with Machine Learning," *ACM Computing Surveys,* vol. 55, no. 2, 2022.

[21] Lakshin Pathak and Kiran Kher, "Green Software Engineering: A Comprehensive Study," *International Journal of Innovative Science and Research Technology,* vol. 9, no. 2, pp. 698-704, 2024.

[22] Berntsen, K.R., Olsen, M.R., Limbu, N., Tran, A.T., Colomo-Palacios, R., "Sustainability in Software Engineering - A Systematic Mapping," *Trends and Applications in Software Engineering. CIMPS 2016. Advances in Intelligent Systems and Computing - Springer,* pp. 23-32, 2017.

[23] M. Dhaini, M. Jaber, A. Fakhereldine, S. Hamdan, and R. A. Haraty, "Green computing approaches - A survey," *Informatica,* vol. 45, no. 1, pp. 1-12, 2021.

[24] S. Benhamaid, A. Bouabdallah, and H. Lakhlef, "Recent advances in energy management for green-IoT: An up-to-date and comprehensive survey," *J. Netw. Comput. Appl,* vol. 198, 2022.

[25] Ahmad Ibrahim, S. R., Yahaya, J., & Sallehudin, H., "Green Software Process Factors: A Qualitative Study," *Sustainability,* vol. 14, no. 18, p. 11180, 2022.

[26] Noman, H., Mahoto, N. A., Bhatti, S., Abosaq, H. A., Al Reshan, M. S., & Shaikh, A., "An Exploratory Study of Software Sustainability at Early Stages of Software Development," *Sustainability,* vol. 14, no. 14, p. 8596, 2022.

[27] Santos, L. C. d., da Silva, M. L. P., & dos Santos Filho, S. G., "Sustainability in Industry 4.0: Edge Computing Microservices as a New Approach," *Sustainability,* vol. 16, no. 24, p. 11052, 2024.

[28] B. Penzenstadler, A. Raturi, D. Richardson, and B. Tomlinson, "Safety, security, now sustainability: The non-functional

requirement for the 21st century," *IEEE Software,* vol. 31, no. 3, pp. 40-47, 2014.

[29] K. Naik and S. P. Mohanty, "Green Mobile Computing: Energy Saving Techniques," *CRC Press,* 2016.

[30] B. W. Boehm, "Software engineering economics," *IEEE Transactions on Software Engineering,* vol. 1, pp. 4-21, 1984.

[31] S. Murugesan, "Harnessing Green IT: Principles and practices," *IT Professional,* vol. 10, no. 1, pp. 24-33, 2008.

[32] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems,* vol. 29, no. 4, pp. 1012-1023, 2013.

[33] W. C. Dietrich, C. Görg, and A. Winter, "An empirical study on the influence of green software development on code quality," *Journal of Software: Evolution and Process,* vol. 30, no. 6, 2018.

[34] A. Hindle, "Green mining: A methodology of relating software change and configuration to power consumption," *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR 2012),* pp. 78-87, 2012.

[35] A. Shehabi, S. J. Smith, D. A. Sartor, R. Brown, M. Herrlin, J. G. Koomey, E. Masanet, N. Horner, I. Azevedo, and W. Lintner, "United States data center energy usage report," *Lawrence Berkeley National Laboratory,* 2016.

[36] J. H. Abawajy, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems,* vol. 28, no. 5, pp. 755-768, 2012.

[37] H. Alemzadeh, Z. Kalbarczyk, R. K. Iyer, and J. Raman, "Analysis of safety-critical computer failures in medical devices," *IEEE Security & Privacy,* vol. 11, no. 4, pp. 14-26, 2013.

[38] B. Le Calvar, F. Jouault, A. Krioukov, and A. L. Hee, "Greening software development: Energy usage of computational tools and software," *Journal of Systems and Software,* 2022.

[39] G. Procaccianti, P. Lago, and A. Vetrò, "Software sustainability from a software architecture perspective," *Proceedings of the 2016 IEEE/ACM International Conference on Software Engineering Companion,* pp. 423-431, 2016.

[40] Mishra, S., Dehury, N., "Big Data Analytics for Smart Grids, the Cyberphysical System in Energy—A Bibliographic Review," *Advances in Intelligent Computing and Communication. Lecture Notes in Networks and Systems - Springer,* vol. 202, 2021.

[41] Kurian Joseph; Saeid Eslamian; Kaveh Ostad-Ali-Askari; Mohsen Nekooei; Hossein Talebmorad; Ali Hasantabar Amiri, "Environmental Impact Assessment as a Tool for Sustainable Development," *Encyclopedia of Sustainability in Higher Education,* pp. 1-9, 2018.

[42] M. Goedkoop and R. Spriensma, "The Ecoindicator 99 - A damage oriented method for Life Cycle Impact Assessment," 2000.

[43] Chukka NDKR, Arivumangai A, Kumar S, et al., "Environmental Impact and Carbon Footprint Assessment of Sustainable Buildings: An Experimental Investigation," *Adsorption Science & Technology,* 2022.

[44] Javier Mancebo, Félix García, Coral Calero, "A process for analyzing the energy efficiency of software," *Information and Software Technology,* vol. 134, 2021.

[45] Mohankumar Muthu, K. Banuroopa and S. Arunadevi, "Green and Sustainability in Software Development Lifecycle Process," *Sustainability Assessment at the 21st Century,* 2019.

[46] Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, João Saraiva, "Ranking programming languages by energy efficiency," *Science of Computer Programming - Elsevier,* vol. 205, pp. 1-30, 2021.

[47] Ishtar Starxin, "Sustainable Software Development – Criteria from Theory and Their Use in Practice (Master Thesis)," 2020.

[48] Lago, P., Kazman, R., Meyer, N., Morisio, M., Muller, "Exploring initial challenges for green software engineering: summary of the first GREENS workshop," *Software Engineering Notes (ICSE),* vol. 31, no. 8, pp. 31-33, 2012.

[49] Raisian, K., Yahaya, J., & Deraman, A., "Green Measurements for Software Product Based on Sustainability Dimensions," 2021.

[50] Brixio, "A step on the way to a Greener Software Engineering," 2022.

[51] Macris, J, "Exploring the Latest Emerging Trends in Software Engineering: Technologies, Tools, and Techniques.," 2023.

Author Contributions

Enes Bajrami: Developed the study framework, designed and implemented the methodology, conducted the data analysis, and authored the manuscript. Ensured the reliability and accuracy of the findings and their alignment with the research objectives.

Acknowledgments

The author would like to express sincere gratitude to all the factories that participated in the questionnaire. Their generous allocation of time and provision of invaluable insights were crucial in advancing the understanding of green software engineering practices across various industries. The contributions made by the participants were key in ensuring a comprehensive analysis of the subject matter. The author deeply appreciates their willingness to share information and their essential support in facilitating this research.

Conflict of Interest Notice

The author declares no conflict of interest.

Artificial Intelligence Statement

The author used AI tools to improve readability and language, then reviewed and edited the content, taking full responsibility for the final publication.

Plagiarism Statement

This article has been scanned by iThenticate ™.