

Review Article

Green Software Metrics and Tools: A Systematic Literature Review

Özkan Yıldız¹, A.Selçuk Güceğlioğlu²

^{1,2}Informatics Institute, Middle East Technical University, Ankara, Türkiye.

¹Corresponding Author : ozkanyil1@yahoo.com

Received: 20 January 2026

Revised: 24 July 2026

Accepted: 29 July 2026

Published: 30 September 2026

Abstract - The increasing energy consumption of the Information and Communication Technologies (ICT) sector and the growing carbon emissions demonstrate the importance of sustainable software engineering practices. This study presents a Systematic Literature Review (SLR) covering the years 2020 to September 2025 to review and analyze research on green software metrics, tools, and validation methods. This study addresses five research questions, focusing on (i) green metrics, (ii) their place within the Software Development Life Cycle (SDLC), (iii) validation methodologies, (iv) measurement tools and models, and (v) the use of Artificial Intelligence (AI). The results obtained reveal that energy-focused metrics are widely discussed in the literature. Instruments and empirical experiments represent the most common methodological approaches used for validation. While emerging models demonstrate the place of machine learning in predictive energy forecasting, the use of AI remains limited, and standardized frameworks are lacking.

Keywords - Energy, Green, SLR, Measurement, SDLC, Software.

1. Introduction

The increasing use of technology worldwide has led to a corresponding growth in energy demand. A significant portion of this energy demand is met by fossil fuels. This increases the risk of lessening the limited resources and causes environmental pollution. Therefore, developing environmentally friendly systems that respond to the growing energy demand is becoming important. This critical reality has made the rigorous examination of energy consumption in Information Technology (IT) processes a priority.

As energy consumption increases faster than expected, the ICT sector has become a significant contributor to this problem. This sector is responsible for approximately 2% of the global carbon emissions. It is estimated to account for 6% to 9% of the world's total energy demand. This figure is expected to increase by 20% by 2030 [1, 2, 3].

One approach being developed to meet the IT sector's growing energy needs is Green Information Technologies (GIT) [4]. GIT is defined as the environmentally friendly design, implementation, and use of IT-related products. Initially, research on energy efficiency primarily focused on decreasing energy consumption in terms of hardware. However, it has become clear that software can also significantly impact the energy consumption of IT products. With ongoing technological advancements, considering the carbon footprint has become crucial for achieving

environmental sustainability. High-energy-consuming domains include data centers, AI model development, cryptocurrency mining, cloud-based applications, and mobile devices. Software is a critical component across all these domains.

To ensure energy efficiency, various studies have focused on improving software design, developing energy-efficient algorithms, and defining metrics and models. Such metrics are necessary to assess "greenness" from the earliest stages of the software lifecycle and ultimately improve energy efficiency. Current SLR studies on green metrics focus solely on determining green metrics and tend to concentrate on specific domains such as blockchain, data centers and the Internet of Things (IoT). This study differs from existing SLRs by focusing specifically on green metrics in software development. The study does not just focus on the identification of green metrics; it also examines the tools used for measurement, analyzes the distribution and frequency of metric usage across the SDLC, and identifies common methods used for metric validation. Moreover, it investigates the determination of green metrics in SDLC phases using AI techniques.

The next section provides a review of the related studies. The third and fourth sections describe the research methodology and data collection procedures of the SLR. The fifth section explains the results, while the sixth section



mentions the discussion and evaluation. The paper concludes with a section summarizing the findings and giving directions for future research.

2. Related Studies

A significant portion of related work includes systematically mapping and reviewing existing metrics and measurements. Calero et al. [5] performed an SLR to identify software sustainability metrics suitable for integration into established quality models (e.g., extensions of the ISO/IEC 25010 standard). Their research identified 82 metrics and related energy consumption to environmental sustainability. It notes that two-thirds of these relate to performance efficiency aspects, particularly resource utilization.

Welter et al. [6] conducted a Systematic Mapping Study (SMS) on green metrics for software development organizations. They classified 222 metrics focusing on areas such as software, applications and algorithms, data centers, service centers, high-performance computing, and cloud computing. Their findings indicate that a large percentage of studies (42.85%) focused specifically on measuring electrical energy.

Bozzelli et al. [7] performed an SLR study on Performance for Sustainable Development focused entirely on green software metrics. The study identified 96 different metrics. They are categorized by the resource being measured and the application context (e.g., application, service center), and the type of metric obtained (e.g., energy, performance, usage). This review revealed trends in research on green software metrics, focusing on how metrics measure and which types of metrics are more prominent in the literature.

Widening this study, Ergasheva et al. [8] performed an SLR on 169 metrics related to energy consumption in embedded, mobile, and cloud systems. They classify the metrics into groups such as hard, code, run-time, indirect, and process metrics. This research found that research related to energy efficiency in mobile systems accounts for a larger proportion of studies (49.2%) compared to embedded (27.5%) and cloud (23.3%) systems.

In their SLR, Debbarma and Chandrasekaran [9] analyzed the evolution of green measurement metrics in sustainable software engineering. The authors examined the types of metrics, measurement methods, and domains applied in green software research. Their findings show that energy and power consumption metrics dominate the field. For instance, estimation-based approaches, often grounded in power models, serve as the most common measurement methods.

Georgiou et al. [10] performed a survey investigating existing techniques and tools for energy efficiency across all phases of the SDLC. Their survey concluded that related work

often concentrated especially on the implementation and verification phases. The study categorized parallel and approximate programming, code refactoring, and monitoring tools.

Danushi et al. [11] conducted an SLR focused on carbon-efficient software development. They classified 65 primary studies using a 5W1H taxonomy. This review distinguished its scope from previous studies that classify metrics [5, 7] or analyze sustainability. They highlighted that while most proposals target software developers, more support is needed for software architects and sustainability engineers.

3. Research Method

This study used the SLR method [12]. SLR is designed to provide a comprehensive and structured synthesis of primary studies in a specific research area. The research questions and their rationales are given below.

Q1: What green metrics are clearly presented in the literature?

R1: The goal of this research question is to identify the green metrics relevant to the software development process. Therefore, it aims to minimize the environmental impact of software development processes.

Q2: In which phases of the SDLC are the identified metrics applied, and how they are utilized across these phases?

R2: This question aims to determine the usage of green metrics in the SDLC phases (e.g., requirements, design, implementation, testing, and maintenance). Therefore, it reveals where the measurement efforts are concentrated. It also identifies the underrepresented phases in the literature.

Q3: What methods are used for the validation of the identified green software metrics?

R3: This research question aims to determine the validation techniques applied in the selected studies. It identifies the common approaches (e.g., empirical evaluation, case study, comparative analysis, expert opinion) to investigate the effectiveness and accuracy of the green software metrics.

Q4: Which existing or new tools/models are used or proposed for measuring the metrics?

R4: This research question aims to identify tools and models used or proposed for green software metrics.

Q5: To what extent are AI techniques integrated into the studies for measuring green software metrics?

R5: This research question aims to analyze the current AI techniques used in green software measurement.

4. Data Collection

The subsequent section explains the activities undertaken in this SLR. It includes the search strategy, data sources, inclusion/exclusion criteria, and study selection procedure.

4.1. Search Strategy

Various key search terms, such as “green,” “software,” “measurement,” “metric,” and “SDLC,” were identified within the context of green software measurement. However, to ensure a comprehensive search and capture a broader range of studies, multiple combined search strings were tested, and related terms were also added. Consequently, the following keywords were identified: (green, sustainable, environmental, energy efficient), (metrics, measurement, indicator, KPI), and (software engineering, software development, SDLC). By combining these keywords, the search string was formed as follows:

(green OR sustainable OR environmental OR “energy efficient”) AND (metrics OR measure* OR indicator* OR KPI) AND (“software engineering” OR “software development” OR SDLC)

The search strategy was also enhanced by adding advanced query operators, such as “intitle: green,” to increase the relevance of the retrieved results.

4.2. Data Sources

The period from 2020 to September 2025 was chosen to capture the most recent studies in the field. The systematic literature search was applied in IEEE XPLORE, SCOPUS, WEB OF SCIENCE, and GOOGLE SCHOLAR. These databases are widely used in academic research. Preliminary analysis displayed that a significant portion of studies on green software measurement were indexed in these sources.

4.3. Studies Selection

In defining the inclusion and exclusion criteria, factors such as the language of publication, year of release, relevance to green software measurement, and publication in peer-reviewed journals were taken into consideration. The inclusion/exclusion criteria practiced in the article selection process are outlined below.

Inclusion Criteria:

- Studies related to green software measurement,
- Papers published between 2020 and September 2025,
- Publication in peer-reviewed journals or conference proceedings,
- Availability of full-text access, and
- Publication in the English language.

Exclusion Criteria:

- Articles focusing exclusively on hardware and lacking any discussion of green software,
- Duplicated papers,
- Gray literature (reports, reports of experiences, short papers, books, magazines).

Initially, queries were conducted in the digital libraries. The search criteria were tailored to the specific characteristics of each database. In addition to the default search parameters, “intitle” and “source title” operators were used in Google Scholar and Scopus to increase sensitivity. In Scopus, also retrieved articles were filtered to include only those in Computer Science.

Additionally, conference proceedings and journal articles indexed by the IEEE were selected. Targeted areas included Web of Science (WOS) categories such as Computer Science Theory and Methods, Computer Science Software Engineering, Computer Science Interdisciplinary Applications, and Computer Science Information Systems. The automated search process yielded a total of 6,543 primary studies. All searches were conducted using the “Sort by Relevance” criterion.

In the second stage, the top 200 articles were selected from each database based on their relevance ranking. Initially, it was determined that relevant articles were among the top 50 results. Within the next 50 articles, very few related articles were identified. In addition, to avoid missing keywords, the upper limit for the number of articles to be selected from each database was set at 200. As a result, a total of 800 articles from the four databases were included in the study. These 800 studies were manually screened, and after applying the predefined inclusion and exclusion criteria, 103 articles were selected. Duplicate records were removed, and a total of 76 articles were identified. Subsequently, the reference lists of the 76 articles were examined using the snowball method. With the inclusion of eight additional studies, the total number of articles reached 84. This approach allows the identification of related publications referenced by the initially selected studies.

The selected articles were assessed using the Likert scale (strongly agree, agree, neutral, disagree, strongly disagree). During the assessment, the existence of metrics, tools or models used or developed, association of metrics with the SDLC phases, and the application of the validation methods were examined. Based on the results of the Likert scale evaluation, 14 articles were identified for the final analysis. Two of these articles were found by using the snowballing method. The number of articles obtained through the study selection process is listed in Table 1, and the selected studies are detailed in Appendix 1.

Table 1. The number of articles obtained through the study selection process

Database	Initial Results	Selected After Screening	After Removing Duplicated Records	Final Included Studies
IEEE	4088	21	21	5
WOS	376	16	12	1
SCOPUS	909	41	28	4
GOOGLE SCHOLAR	1170	25	15	4

To ensure the reliability of the selection process, each article was independently reviewed by two reviewers. By reviewing the title, abstract, keywords, and introduction of each study, the reviewers applied the predetermined inclusion and exclusion criteria. Articles rated “strongly agree” on the Likert scale were accepted. Articles rated as “strongly disagree” were excluded. Articles rated as “agree” or “neutral” were reviewed jointly by two reviewers to determine their inclusion in the final dataset. In cases where consensus was not reached, the relevant article was re-evaluated. Cohen's Kappa statistic [13] was applied to further increase inter-rater reliability during the selection process. The percentage of agreement was found to be 75%, corresponding to the “strongly agree” level. Selection process of primary studies are given in Figure 1.

4.4. Data Extraction

The following data was extracted from each study: Metrics, SDLC Phase, Validation Method, Frameworks/Tools, and AI Usage. Using the prepared dataset, supplementary tables were developed, including Metric/Article, Tool/Article, SDLC/Article, Validation Method/Article, and AI Usage/Article to carry out a more detailed and structured analysis.

4.5. Search Strategy

The validation threats identified in this study are as follows. The first threat is keyword bias. This threat arises from the search terms not being selected to encompass all research in the field of green software measurement. Some relevant studies may have been unintentionally excluded due to conceptual ambiguity. The second threat is about quality assessment bias. Subjective assessments may have been made during the evaluation of inclusion and exclusion criteria. To mitigate this threat, the evaluation process was conducted independently by two reviewers, and disagreements were resolved through consensus discussions. In this way, it was aimed to increase methodological transparency, minimize reviewer subjectivity, and enhance the overall reliability of the review findings.

5. Results

In this section, the responses to the research questions are analyzed and discussed in detail.

Q1: What green metrics are clearly presented in the literature?

The SLR study identified a total of 33 different metrics. The metrics used in multiple studies with their corresponding article numbers are presented in Table 2, and the metric, tool and validation methodology lists are given in Appendix 2. The important key metrics in measuring green software are Energy Consumption, referring to the amount of electrical energy consumed in joules, and the overall Carbon Footprint, specified as the total amount of greenhouse gas emissions produced.

Another central metric used for optimization is Energy Efficiency. It is calculated as the ratio of useful work per joule, and it represents the energy-performance relationship of software. Metrics such as CPU Usage and Memory Usage are critical metrics that indicate the consumption of core computing resources. For graphic operations, instantaneous Power Consumption by the processor is measured during the execution through GPU Usage and GPU Energy Consumption

Operational factors are measured through hardware interaction metrics such as Disk Activity (read/write operations) and Network Activity (data transmitted and received). To accurately measure performance, measurement methodologies establish baselines using Idle Power. In that way, the overhead is quantified through lifecycle stages such as Boot Energy Usage, Installation Energy, and Removal Energy.

Software quality metrics, such as Maintainability Index, are highly correlated with energy impact. For instance, metrics such as Lines of Code are used to analyze the code structure, and by counting Code Smells, potential anti-patterns are identified. Measures such as Execution Time and resource allocation concepts, including Energy Proportion (%) and Memory Proportion (%), are used for comparative performance analysis. Finally, advanced approaches aim to provide practical implementation feedback and provide transparency through precise identification of Energy-Hungry Code Portions.

The most substantiated green metrics explicitly mentioned are Energy Consumption, CPU Usage and Memory Usage. These metrics are frequently referenced in most relevant articles, indicating their broad acceptance and trust. Power Consumption and Energy Efficiency are among the highly substantiated metrics. These are the core measures used in numerous articles to define and measure the green features of software.

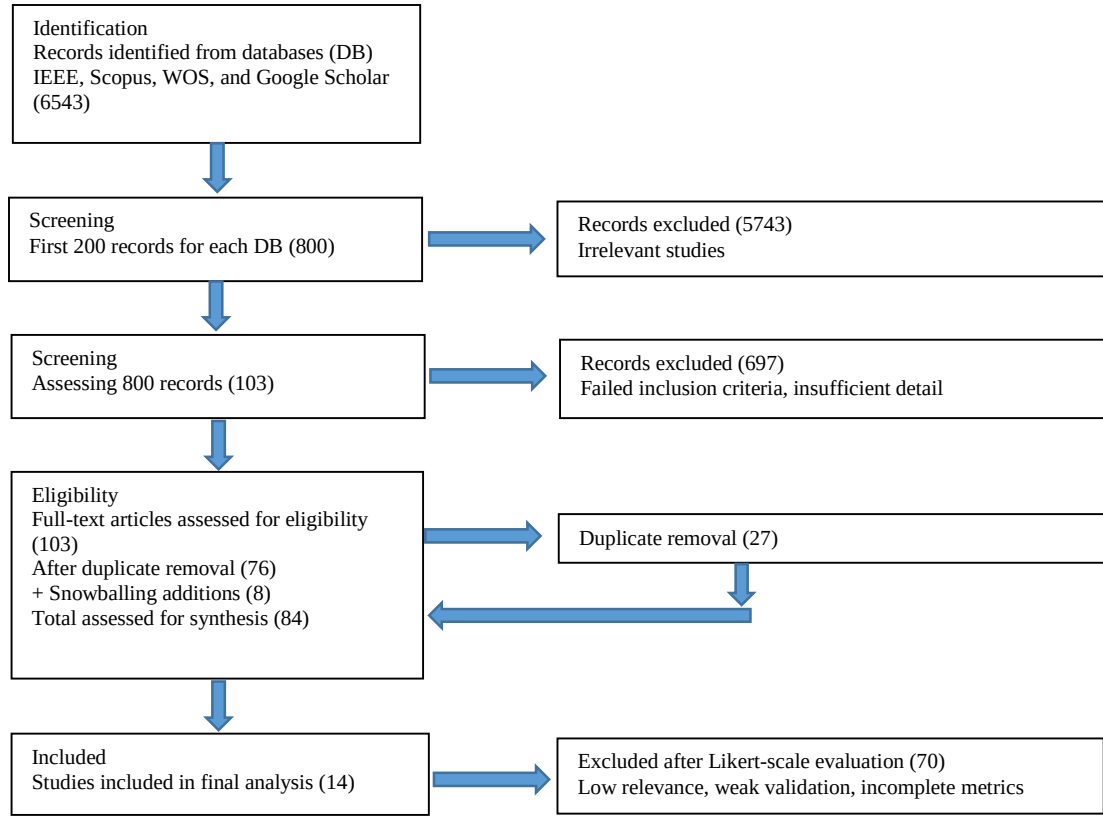


Fig. 1 Selection process of primary studies

Table 2. Metrics referenced in multiple studies

Metric Name	Article Number
Energy Consumption	1,2,3,4,6,8,9,10,11,14
CPU Usage	1,5,6,7,8,13
Memory Usage	1,5,7,8,10,13
Power Consumption	3,8,9,10,11
Energy Efficiency	3,8,11,13,14
Execution Time	5,8,9,11
Carbon Footprint	1,6,13
Disk Activity	5,7,8
Network Activity	5,7,8
Maintainability Index	4,7,12
Resource Utilization/Efficiency	1,14
Idle Power	5,13
Memory Proportion	10,13
Lines of code	11,12

Q2: In which phases of the SDLC are the identified metrics applied, and how they are utilized across these phases?

The SLR reveals that green metrics are integrated into nearly all phases of the SDLC. The usage rates are largely concentrated in the implementation-dependent and post-implementation phases. The frequencies of studies using the SDLC phases are given in Table 3. The implementation phase displays the highest frequency. It is explicitly referenced or implicitly targeted in eleven articles.

The Maintenance phase is addressed in nine sources. It focuses on continuous monitoring of operational energy consumption, hardware reuse, and optimizing idle behavior throughout the software's lifespan. The Testing phase is cited in eight articles. It uses metrics such as Energy Consumption, CPU Usage, and Memory Utilization to validate environmental requirements. These metrics are used to determine energy-hungry code segments and benchmark test script efficiency. Upstream in the SDLC, the Design phase appears in four articles and includes metrics such as Code Smells and Complexity for guiding architectural efficiency. Design metrics can also be predictive. Models such as Model-View-Controller Complexity and Code Smells Energy Model (MVC-CCsEM) uses static metrics to estimate energy consumption before application execution.

The Deployment phase shows moderate metric usage (cited in four sources). It concerns focusing on Carbon Footprint. In contrast, the Requirement phase is mentioned least frequently (in only one source). It focuses on defining sustainability goals and target metrics, such as energy limits and efficiency. Additionally, it emphasizes the relevant stakeholders and environmental impacts that need to be considered throughout the software lifecycle (e.g., energy consumption, resource efficiency, and carbon footprint). The Inference phase is cited in only one source. It aims to evaluate the sustainability impact of Large Language Model (LLM)

usage in real-world developer workflows, not during training or model creation phases. Generally, metrics such as Energy Consumption and CPU Utilization are continuously monitored throughout the Implementation, Testing, and Maintenance phases. While structural quality metrics, such as Maintainability Index, correlate code structure with efficiency throughout the lifecycle. These metrics largely supports operational activities where performance and resource consumption can be directly measured.

Table 3. The frequency of studies utilized across the SDLC phases

SDLC Phase	Frequency
Requirements	1
Design	4
Implementation	11
Deployment	4
Testing	8
Maintenance	9
Inference/Operation Phase	1

Q3: What methods are used for the validation of the identified green software metrics?

The validation methodologies employed in green software measurement studies rely on empirical data collection. In the SLR study, it is observed that the most frequently used validation method is Empirical Experimentation (Table 4). This method is used in eight studies. It involves running real-world execution tests under controlled environments and verifies the energy impact of new programming choices. The second most frequently used method is the Use of a Standard Measurement Tool, referenced in six studies. This method standardizes data collection using tools such as Intel Power Gadget, Power Stat, or Motorola Assessment Tool.

Comparative Analysis is included in four studies. With this method, measurable efficiency gains are determined by comparing different software elements such as algorithms, language implementations, and refactoring. It validates the optimization potential of the model by comparing the energy consumption of optimized code with the baseline code. Case Studies are used in three studies. Effectiveness and applicability are evaluated by new models such as the Green-Agile Maturity Model (GAMM). The repeated Measurements for Statistical Validity method is used in two studies. It ensures the validity of findings by testing the statistical significance of energy differences between original and revised versions. Additionally, Hardware Benchmarking (used twice) provides metric reliability through experimental measurements by validating energy models against known hardware setups. Finally, Literature Review and Expert Interviews (both used twice) establish the theoretical basis for the metrics and experimentally validate key factors through questionnaires.

Table 4. Validation methodologies referenced in multiple studies

Validation Methodology	Article Number
Empirical Experimentation (Experimental Validation)	3,4,5,9,10,11,12,13
Use of a Standard Measurement Tool (Tool-Based Validation)	3,7,8,10,13,14
Comparative Analysis	1,10,11,13
Case Study	3,9,14
Repeated Measurements for Statistical Validity	3,4
Hardware Benchmarking	6,11
Literature Review	1,14
Expert Interviews	1,14

Q4: Which available or new tools/models are used or proposed for measuring the metrics?

Measurement tools such as Intel Running Average Power Limit (RAPL), Intel Power Gadget, and nvidia-smi are used for capturing low-level energy consumption, such as CPU, DRAM, and GPU power usage (Appendix 2). These tools collect data using interfaces provided by the manufacturer. For platform-specific measurement, while PowerJoular enables real-time, cross-platform hardware monitoring to facilitate integration with advanced application monitoring frameworks, Trepp Profiler targets Android devices based on Qualcomm Snapdragon. JoularJX serves as a specialized Java agent that uses data from PowerJoular to accurately estimate energy consumption. It provides information about power usage per method and per thread within Java applications.

Comprehensive environmental measurement frameworks, like the Green Metrics Tool (GMT) enable multidimensional monitoring of over fifteen metrics. It includes temperature, disk activity, and network utilization metrics. GMT uses containerization technologies such as Docker to provide repeatable, isolated, and consistent test environments. Tools such as Selenium (for web applications) and JMeter (for performance and load testing) are widely used for automation and performance testing. They are often complemented by Linux-based monitoring utilities such as the Energy Spot Measuring Tool (ESMT) during the testing phase.

Static analysis platforms such as SonarQube and SonarCloud are used to evaluate code sustainability. These tools assess code quality, complexity and code smells. JHawk and Microsoft Code Metrics Tool also evaluate the maintainability and efficiency. In addition, predictive modelling approaches are used as alternatives to computationally intensive dynamic measurements. For example, MVC-CCsEM estimates application-level energy consumption without execution. It uses software quality metrics such as Cyclomatic Complexity and Duplicated Lines of Code for reducing the energy rebound effect.

GreenCode-PowerMeter (GCPM) is a model-based prediction framework. It can be integrated into the Visual Studio Code (VS Code) Integrated Development Environments (IDE), providing accurate, real-time power analysis by combining machine learning techniques. Moreover, frameworks such as the Emission Estimation Framework and SpecPower XGBoost Model are designed for energy estimation in complex or resource-constrained environments. The first one relies on Thermal Design Power (TDP) assumptions to estimate CPU energy consumption, while the other one uses machine learning to predict power consumption in cloud-based and virtualized server infrastructures.

At the organizational and procedural levels, the Green Software Measurement Process (GSMP) and the GAMM provide systematic methodologies and maturity models for integrating sustainable practices throughout the SDLC.

The number of tools utilized and developed are given in Figure 2.

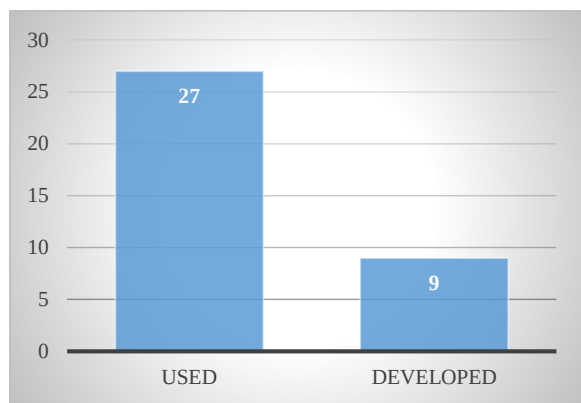


Fig. 2 The number of tools utilized and developed

Q5: To what extent are AI techniques integrated into the studies for measuring green software metrics?

The integration of AI techniques into solutions for evaluating and optimizing green software metrics is currently an area requiring expertise. The focus is primarily on energy prediction models. For this purpose, advanced language processing capabilities are used for code optimization. One of the areas of integration is LLMs. These models are actively used to explore energy accuracy trade-offs during software development tasks. For instance, GMT uses a configurable external LLM (Llama, ChatGPT, Mistral) to provide optimization advice on inefficient code segments. It rates and evaluates code segments and generates code improvements focused on reducing resource use. Code Green study evaluates existing AI models from a sustainability standpoint. This study heavily relies on AI, specifically LLMs, to investigate the trade-offs among code quality, energy efficiency, and environmental sustainability in software development.

Furthermore, GCPM can be directly integrated into the VS Code environment to enable real-time measurement. It uses machine learning techniques for predictive power estimation by including a decision tree model to achieve precise power consumption measurements.

AI is also integrated for resource estimation, where direct hardware measurement is not possible. As an example, the SPECPower XGBoost Model is a machine learning-based estimation tool that estimates the energy usage of cloud or virtualized server environments. Moreover, research also includes assessing the impact of AI itself on sustainability, such as measuring the energy and carbon footprint associated with LLM inference. However, many fundamental green software measurement models, such as MVC-CCsEM, GreenPy, and GAMM, do not integrate AI or machine learning into their scope (Figure 3).

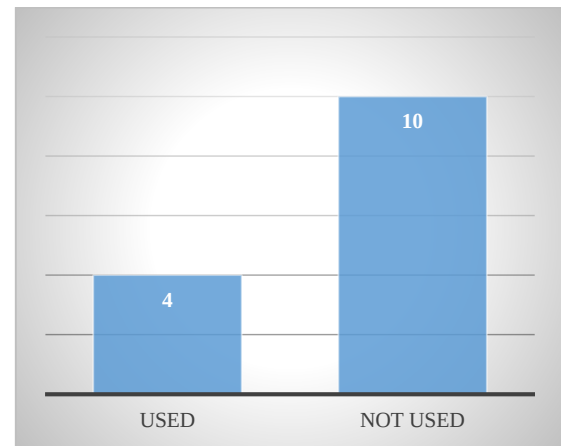


Fig. 3 AI usage

6. Discussion and Evaluations

The results of this SLR demonstrate that the Green Software Measurement research has made progress in defining and applying metrics. However, it also reveals that standardization in these studies is limited. While such metrics provide valuable insights into software efficiency, they represent a narrow green interpretation of broader environmental aspects. The review also shows that these metrics are primarily applied in the later stages of the SDLC, where measurement and optimization can be directly observed. On the other hand, the studies related to the early stages are still insufficient. This indicates that greenness issues are generally accepted as corrective factors rather than preventive ones.

The integration of AI is not equally distributed across the SDLC phases. Current AI-supported approaches are essentially used as predictive tools at the application level. Therefore, AI integration in requirements and design phases is not specified. This indicates a gap in the integration of AI throughout the SDLC phases.

Empirical experiments are one of the most frequently used validation methods. They typically use tool-based validation methods. These approaches increase measurement precision but limit generalizability due to hardware dependencies. To validate the software features, statistical methods are widely applied.

Tool development in the field of green software measurement is diverse. While most of the tools focus on detailed measurement of energy consumption and performance metrics, others use machine learning-based prediction methods to reduce measurement overhead.

This approach represents a shift from reactive measurement to proactive prediction. However, the integration of AI in these tools remains limited. It serves predictive energy modelling rather than providing comprehensive benefits.

When the frameworks discussed in the examined articles are compared, AI integration is observed only in GCPM. Green Software Life Cycle Model (GSLCM) and GAMB cover all SDLC phases, whereas the remaining frameworks focus on specific phases. Although the literature promotes lifecycle integration at a conceptual level, measurement practices remain largely execution-based.

Furthermore, GAMB operates at the organizational maturity level, whereas the other frameworks concentrate primarily on the process level. In addition, empirical evidence supporting the practical effectiveness of these frameworks remains limited. This highlights a need for systematic maturity assessments to evaluate the impact of these frameworks on carbon and energy reduction.

Another problem encountered in the literature is the lack of unified taxonomies and categorization of reference models. This situation leads to inconsistencies in interpreting what a “green” criterion or tool is. This inconsistency makes it difficult to compare, replicate, integrate, validate, and create standardized frameworks across the studies.

7. Conclusions and Future Works

Technological advancements continue to increase the number and scope of the systems in use. These developments will also increase the required energy demand in parallel. This situation increases the importance of studies focusing on green energy. In this context, the findings of the SLR indicate that the importance given to hardware-centric green metrics should be also extended to software-centric green metrics. Furthermore, it is estimated that the use of AI across the field, not only for predictive purposes but also for process improvement efforts, would yield significant energy savings. However, the use of AI should also be evaluated in terms of increased computational energy consumption. In addition, developing a general taxonomy that brings together existing models under a common structure would reduce fragmentation and improve consistency in the field. Adding green metric measurement interfaces to commonly used SDLC tools, such as IDEs, could encourage developers to consider sustainability during development activities. Finally, creating benchmark applications that compare different models and frameworks under similar conditions would help researchers and practitioners identify the most suitable approaches for applying green metrics.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

- [1] Peng Wang et al., “Trends in Energy Consumption Under the Multi-Stage Development of ICT: Evidence in China From 2001 to 2030,” *Energy Reports*, vol. 8, pp. 8981-8995, 2022, [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [2] Jukka Manner, “Black Software-The Energy Unsustainability of Software Systems in the 21st Century,” *Oxford Open Energy*, vol. 2, pp. 1-8, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [3] Simon Mingay, and Dennis Pamlin, “Assessment of Global Low-Carbon and Environmental Leadership in the ICT Sector,” Dennis Pamlin, 2008. [[Publisher Link](#)]
- [4] Roberto Verdecchia et al., “Green IT and Green Software,” *IEEE Software*, vol. 38, no. 6, pp. 7-15, 2021. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [5] Coral Calero, Manuel F. Bertoa, and Ma Angeles Moraga, “A Systematic Literature Review for Software Sustainability Measures,” *2013 2nd International Workshop on Green and Sustainable Software (GREENS)*, San Francisco, CA, USA, pp. 22-28, 2013. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [6] Marcio Welter, Fabiane Barreto Vavassori Benitti, and Marcello Thiry, “Green Metrics to Software Development Organizations: A Systematic Mapping,” *2014 XL Latin American Computing Conference (CLEI)*, Montevideo, Uruguay, pp. 1-7, 2014. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [7] Patricia Lago, Qing Gu, and Paolo Bozzelli, “A Systematic Literature Review of Green Software Metrics,” VU Technical Report, pp. 1-27, 2014. [[Google Scholar](#)] [[Publisher Link](#)]
- [8] S. Ergasheva et al., “Metrics of Energy Consumption in Software Systems: A Systematic Literature Review,” *IOP Conference Series: Earth and Environmental Science*, vol. 431, no. 1, pp. 1-11, 2020. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]

- [9] Tribid Debbarma, and K. Chandrasekaran, “Green Measurement Metrics Towards a Sustainable Software: A Systematic Literature Review,” *2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, Jaipur, India, pp. 1-6, 2016. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [10] Stefanos Georgiou, Stamatia Rizou, and Diomidis Spinellis, “Software Development Lifecycle for Energy Efficiency: Techniques and Tools,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1-33, 2019. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [11] Ornela Danushi, Stefano Forti, and Jacopo Soldani, “Carbon-Efficient Software Design and Development: A Systematic Literature Review,” *ACM Computing Surveys* vol. 57, no. 10, pp. 1-35, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [12] Barbara Kitchenham, and Stuart Charters, “Guidelines for Performing Systematic Literature Reviews in Software Engineering (EBSE 2007-001),” *Keele University, UK*, pp. 1-65, 2007. [[Google Scholar](#)]
- [13] Mary L. McHugh, “Interrater Reliability: The Kappa Statistic,” *Biochemia Medica*, vol. 22, no. 3, pp. 276-282, 2012. [[Google Scholar](#)] [[Publisher Link](#)]
- [14] Lauri Kivimäki et al., “Building up Green Software Life Cycle Model,” *2024 10th International Conference on ICT for Sustainability (ICT4S)*, Stockholm, Sweden, pp. 20-28, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [15] Boris Belchev, “Code Green: Evaluating the Carbon and Energy Implications of LLM Integration in Software Development,” *Master's Thesis, University of Twente*, Enschede, The Netherlands, 2025. [[Google Scholar](#)]
- [16] Nurzihan Fatema Reya et al., “GreenPy: Evaluating Application-Level Energy Efficiency in Python for Green Computing,” *Annals of Emerging Technologies in Computing (AETiC)*, vol. 7, no. 3, pp. 92-110, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [17] Ibrahim Şanlıalp, Muhammed Maruf Öztürk, and Tuncay Yiğit, “Energy Efficiency Analysis of code Refactoring Techniques for Green and Sustainable Software in Portable Devices,” *Electronics*, vol. 11, no. 3, pp. 1-17, 2022. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [18] Geerd-Dietger Hoffmann, and Verena Majuntke, “Green Metrics tool: Measuring for fun and Profit,” *arXiv preprint arXiv*, pp. 1-3, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [19] Adel Nouredine, “PowerJoular and JoularJX: Multi-Platform Software Power Monitoring Tools,” *2022 18th International Conference on Intelligent Environments (IE)*, Biarritz, France, pp. 1-4, 2022. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [20] Rishita Tiwari, “Evaluating Code Sustainability: A Comprehensive Study of Metrics and Tools,” *2023 IEEE Engineering Informatics*, Melbourne, Australia, pp. 1-7, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [21] Achim Guldner et al., “Development and Evaluation of a Reference Measurement Model for Assessing the Resource and Energy Efficiency of Software Products and Components-Green Software Measurement Model (GSMM),” *Future Generation Computer Systems*, vol. 155, pp. 402-418, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [22] Amina Nasrin, “Promoting Green Coding in VS Code with GCPM: A Power Model for Heterogeneous Processors,” *2024 IEEE 15th International Green and Sustainable Computing Conference (IGSC)*, Irvine, CA, USA, pp. 110-115, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [23] Shraddha Bhadane et al., “Sustainable Programming: A new Paradigm Shift Towards Sustainable Development,” *2024 IEEE Pune Section International Conference (PuneCon)*, Pune, India, pp. 1-6, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [24] P.S. Felix, and M. Mohankumar, “Energy Optimization in Software Development: A Comparative Study of Sorting Techniques,” *International Journal of Engineering Trends and Technology*, vol. 72, no. 7, pp. 339-349, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [25] Daniel Guamán, Jennifer Pérez, and Priscila Valdiviezo-Diaz, “Estimating the Energy Consumption of Model-View-Controller Applications,” *The Journal of Supercomputing*, vol. 79, no. 12, pp. 13766-13793, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [26] G. Anithakrishna, and M. Mohankumar, “SEFGAST: Step-up to Environment Friendly Green Automated Software Testing,” *International Journal of Engineering Trends and Technology*, vol. 70, no. 3, pp. 162-169, 2022. [[CrossRef](#)] [[Publisher Link](#)]
- [27] Nasir Rashid et al., “Green-Agile Maturity Model: An Evaluation Framework for Global Software Development Vendors,” *IEEE Access*, vol. 9, pp. 71868-71886, 2021. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]

Appendix 1 Selected Studies

Study ID	Ref. no.	Title	Year
1	[14]	Building up green software life cycle model	2024
2	[15]	Code Green: Evaluating the Carbon and Energy Implications of LLM Integration in Software Development	2025
3	[16]	GreenPy: Evaluating application-level energy efficiency in Python for green computing	2023
4	[17]	Energy efficiency analysis of code refactoring techniques for green and sustainable software in portable devices	2022
5	[18]	Green metrics tool: Measuring for fun and profit	2024
6	[19]	PowerJoular and JoularJX: Multi-platform software power monitoring tools	2022

7	[20]	Evaluating code sustainability: A comprehensive study of metrics and tools	2023
8	[21]	Development and evaluation of a reference measurement model for assessing the resource and energy efficiency of software products and components—Green Software Measurement Model (GSMM)	2024
9	[22]	Promoting green coding in VS Code with GCPM: A power model for heterogeneous processors	2024
10	[23]	Sustainable programming: A new paradigm shift towards sustainable development	2024
11	[24]	Energy optimization in software development: A comparative study of sorting techniques	2024
12	[25]	Estimating the energy consumption of model-view-controller applications	2023
13	[26]	SEFGAST: Step-up to environment friendly green automated software testing	2022
14	[27]	Green-agile maturity model: An evaluation framework for global software development vendors	2021

Appendix 2 Metrics, Tools and Validation Methodologies

Metric List			
Name	Study ID	Definition	Unit
Energy Consumption	1,2,3,4,6,8,9,10,11,14	Refers to the amount of electrical energy consumed, measured in joules	joules (J)
Carbon Footprint	1,6,13	The total greenhouse gas emissions produced	kg CO ₂ e
Resource Utilization / Efficiency	1,14	Refers to how effectively computing resources (CPU, memory, network, storage) are used	%
Hardware Utilization Efficiency	1	Measures the optimal use of hardware capabilities to avoid overprovisioning and idle energy waste	%
CPU Usage	1,5,6,7,8,13	Measures CPU usage	%
Memory Usage	1,5,7,8,10,13	Measures memory consumption	MB
Reusability Rate	1	Measures code reuse	%
Software Lifetime / Longevity	1	Refers to software life	years, months, or days
Power Consumption	3,8,9,10,11	Represents the amount of electrical power used by a processor during program execution, measured in watts	Watts
Maintainability Index	4,7,12	Reflects how easy it is to maintain or modify code	normalized to a 0–100 scale
Execution Time	5,8,9,11	Captures the time taken to complete (runtime performance)	seconds (s)
Thermal Characteristics	5	Refers to CPU temperature and how heat is generated and dissipated	°C
Disk Activity	5,7,8	Measures read/write operations to assess storage impact	B/s, KB/s, MB/s, GB/s
Network Activity	5,7,8	Quantifies data transmitted and received during execution	B/s, KB/s, MB/s, GB/s
Idle Power	5,13	Establishes a baseline for system resource use when the software is not active	W
Boot Energy Usage	5	Refers to resource consumption when starting up the system	joules (J)
Installation Energy	5	Refers to energy used during the installation and build process	joules (J)
Removal Energy	5	Measures resource impact during uninstallation	joules (J)
Page Faults	5	Indicates memory inefficiencies that can impact	count

		energy usage	
PIDs (Processes/Threads)	7	Refers to the number of active processes	count
Code Smells	7	Refers to quality flaws identified via tool	Smells/K LOC
Percentage Greenness	7	Refers to a score to rate the environmental friendliness of code	%
GPU usage	8	Refers to GPU memory usage	%
GPU Energy Consumption	8	Energy used by GPU operations	joules (J)
Energy Efficiency	3,8,11,13,14	Useful work per Joule Energy-to performance ratio	%
Energy Proportion	10	The relative share of total energy used for comparative efficiency analysis	%
Memory Proportion	10,13	Reflects efficiency in resource utilization	%
Lines of code	11,12	As a code efficiency indicator, correlated to energy impact	count
Complexity	12	Refers to cyclomatic complexity	count
Energy Transparency	13	Refers to visibility of energy consumption during the operation	kWh
Energy-Hungry Code Portions	13	Detect parts of test scripts that consume excessive energy	count
Paper Use Reduction	14	Tied to environmental performance in the SDLC; part of “greener” operations	%
Waste (electronic waste) reduction / hardware reuse	14	Cited as a way to reduce environmental footprint	%

Tool List

Name	Study ID	D/U	Definition
GSLCM	1	D	It serves as a conceptual framework to integrate environmental sustainability principles and green metrics (energy, carbon, resource use) into SDLC phases.
nvidia-smi	2	U	Used for GPU energy monitoring, provided by NVIDIA for monitoring, managing, and querying NVIDIA GPU devices on Linux, and Windows
Intel Power Gadget	2,3,4	U	Measuring processor energy, package power, CPU utilization, GPU utilization, and DRAM power
Green Algorithms Calculator	3	U	Estimate the carbon footprint and energy consumption
Trepn Profiler	4	U	Performance and power profiling tool for Android devices powered by Qualcomm Snapdragon processors
Microsoft Code Metrics Tool	4	U	Helps developers assess code quality, maintainability, complexity, and adherence to best practices
JHawk Tool	4	U	Static code analysis and metric tools are specifically designed for Java applications. They help developers assess code quality, complexity, maintainability, and design structure. They also evaluate refactoring impact on mobile devices.
GMT	5,8	D	Tracks 15+ metrics like CPU usage, memory, temperature, disk and network activity, and execution time
Intel RAPL	5,6,9	U	Used for measuring CPU and DRAM energy consumption on supported processors
Puppeteer	5	U	Used for UI benchmarking
Playwright	5	U	Used for UI benchmarking
Selenium	5,13	U	Open-source automated testing tool used to test web applications (“Tour Travo” and “Harmony Shop”) and measure energy consumption
JoularJX	6	D	Specialized tool for monitoring and estimating the energy consumption of Java applications at a fine-grained level—specifically, per thread or per method.
PowerJoular	6	D	Real-time hardware-level energy monitoring tool.
CPU Power Benchmark	6	U	To model and validate CPU energy consumption, especially for devices like the

Tool			Raspberry Pi
docker stats	7	U	Provides real-time monitoring of resource usage for running containers
SonarCloud	7	U	Cloud-based code quality and security analysis service; inspect bugs (functional errors), code smells (poor design, readability issues), security vulnerabilities (e.g., SQL injection, XSS), security hotspots (suspicious code needing manual review), and code duplication
NMON(for Linux)	7	U	Provides real-time and historical insights into CPU, memory, disk I/O, network, kernel, and file system usage
SERENA	8	U	Measures system-level energy/resource consumption using power meters and system resource monitors
GSMP	8	U	Provides structured process for conducting and replicating measurements
SAF Toolkit	8	U	Open-source software framework designed to measure, analyze, and reduce the energy consumption and carbon footprint of software applications—especially AI and machine learning workloads
Green Lab	8	U	Monitor real-time energy usage of applications (e.g., CPU, GPU, memory, and disk), estimate carbon emissions by using location-based electricity grid data, compare the sustainability of different algorithms, architectures, or deployment strategies, and support carbon-aware computing.
SpecPower XGBoost Model	8	U	ML-based energy estimation model, Used for Cloud or virtualized environments where direct measurement is difficult
Software Footprint Tool	8	U	Low-threshold tool to estimate energy use on local dev machines
Emission Estimation Framework	8	U	Estimate CPU energy consumption based on Thermal Design Power (TDP) and assumptions about hardware share
COMM (Container Overhead Measurement Methodology)	8	U	Assess resource overhead from containerization. Lightweight, practical tool for data centers or developers using Docker
GCPM	9	D	Model Design, power estimation technology for computer programs. Power estimation tool embedded in VS Code. Measuring software power consumption directly within the development environment
Power Stat Tool	10	U	Used to measure energy consumption (in Joules) and memory usage (in MB) for different programming languages while executing the same algorithm
Power Reading Unit (PRU)	11	D	Used as hardware measurement tool to measure real-time power and energy consumption during the execution of sorting algorithms
MVC-CCsEM	12	D	Measuring for complexity and code smells
SonarQube	12	U	Performs static analysis and extracts software quality metrics
Microsoft Joulemeter	12	U	Performs dynamic analysis and measures real energy consumption of the applications
ESMT	13	D	A Linux-based tool to estimate CPU and memory utilization during test execution, generate energy usage reports, and support green metric evaluation
JMeter	13	U	Java-based performance and load testing tool used for desktop application
GAMM	14	D	A framework for evaluating how well global software development (GSD) organizations integrate environmental sustainability practices into Agile software development processes
Motorola Assessment Instrument	14	U	Used as a quantitative measurement tool to evaluate each factor (metric) in the GAMM

Validation Methodology List

Methodology	Study ID	Explanation
Literature Review	1, 14	Theoretical validation, systematic analysis of research publications
Expert Interviews	1,14	Empirical validation, semi-structured interviews
Comparative Analysis	1,10,11,13	Compared with conventional models
Analytical (Phase-by-Phase Logical Validation)	1	Analyzing in relation to environmental performance indicators

Direct Energy Measurement via nvidia-smi	2	Measuring with using nvidia-smi
Pareto Frontier Analysis	2	Comparing models in terms of energy efficiency and accuracy
Spearman Correlation	2	Testing the relationship between model architecture (e.g., parameter count, transformer blocks) and energy/performance metrics
Empirical Experimentation (Experimental Validation)	3,4,5,9,10,11,12,13	With using consistent setups and controlled comparisons
Repeated Measurements for Statistical Validity	3,4	Repeated measurements.
Use of a Standard Measurement Tool (Tool Based Validation)	3,7,8,10,13,14	Using standard measurement tools such as Intel Power Gadget, Power Stat, Motorola Assessment Instrument
Case Study	3,9,14	Executing case studies to demonstrate practical validation
Statistical Hypothesis Testing	4	Determining whether the differences in energy consumption between original and refactored versions are statistically significant
Code-based Testing	6	Modify code, and re-measure energy to validate metric responsiveness
Hardware Benchmarking	6,11	Using benchmarks on known hardware setups (e.g., Raspberry Pi CPUs)
Empirical Benchmarking on Real Code	7	Observing and comparing resource utilization metrics (e.g., CPU, memory, network I/O, etc.)
Docker Stats Monitoring	7	Validating real-time metrics for each container
Cross-mapping with Established Measurement Methods	8	By mapping its components to established measurement methods
Benchmark-Based Validation	9	Comparing against well-known reference models such as RAPL
Quantitative Energy Calculation	11	Using the standard energy equation ($E=P \times t$)
Multiple Linear Regression Analysis	12	Determining how the independent variables (metrics: SIZE, SLOC, CC, CS, DL) influence energy consumption
Least Squares Method	12	Fitting the regression model and minimizing the residual error between observed and predicted energy consumption values
Statistical Significance Testing	12	Using p-values to determine the significance of each variable (e.g., , CC, DL, SIZE, SLOC)
Stepwise Regression	12	Selecting the most significant metrics
Model Accuracy Evaluation	12	Measuring the accuracy of energy predictions