Exploring recent technical, methodological, and organizational perspectives on green software practices

Explorando Perspectivas Técnicas, Metodológicas y Organizativas Recientes Sobre Prácticas de *Green Software*

Henríquez-Miranda, C.[1]; Sánchez-Torres, G. [2]

*[1]**Ph. D. Carlos Henríquez Miranda. Profesor Asociado.** Programa de Ingeniería de Sistemas. Facultad de Ingenierías. Universidad del Magdalena. e-mail: chenriquezm@unimagdalena.edu.co. Enlace ORCID https://orcid.org/0000-0001-8252-1413.*

*[2]**PhD. German Sánchez Torres. Profesor Titular.** Programa de Ingeniería de Sistemas. Facultad de Ingenierías. Universidad del Magdalena. gsanchez@unimagdalena.edu.co. Enlace ORCID: https://orcid.org/0000-0002-9069-0732*

Universidad del Magdalena
Carrera 32 #22-08. Santa Marta, Magdalena, Colombia.
Tel: 57-605-4381000 Ext. 3140
E-mail: chenriquezm@unimagdalena.edu.co

Recibido: 20/11/2024 / Aceptado: 12/02/2025

Resumen

La ejecución de algoritmos sofisticados, especialmente en aplicaciones de inteligencia artificial y en la nube, ha acelerado el consumo energético y las emisiones de CO_2 asociadas a su generación. Este estudio realiza una revisión sistemática de la literatura reciente sobre prácticas de software ecológico y su potencial para minimizar la huella de carbono en el sector de las tecnologías de la información y las comunicaciones (TIC). Los resultados ilustran la creciente adopción de técnicas de optimización energética, como el paradigma RMVRVM (*Remote-Model View Remote-View-Model*), estrategias eficientes de ajuste de modelos de lenguaje de gran tamaño y el desarrollo de métricas y metodologías para cuantificar el impacto de las aplicaciones. Sin embargo, se observa la falta de una adopción unificada de estándares y enfoques sostenibles a nivel organizacional. Se concluye que la integración temprana de criterios de eficiencia energética y responsabilidad social a lo largo del ciclo de vida del software es crucial para reducir significativamente el impacto ambiental de la tecnología, y que es fundamental mejorar la formación de los profesionales.

Palabras clave: Software ecológico, optimización energética, huella de carbono.

Abstract

Execution of sophisticated algorithms, particularly in Cloud and artificial intelligence applications has accelerated energy consumption and CO_2 emissions related with energy generation. This study conducts a systematic review of recent literature regarding green software practices and their potential towards minimizing the carbon footprint in the Information and Communications Technology sector. The outcomes illustrate the increasing adoption of energy optimization techniques, such as the Remote-Model View Remote-View-Model (RMVRVM) paradigm, efficient large language model tuning strategies, and the development of metrics and methodology to quantify the application impact. However, the lack of unified adoption of standards and sustainable approaches at an organizational level are observed. It is concluded that the early integration of energy efficiency and social responsibility criteria throughout the software life cycle is crucial to significantly reduce the environmental impact of the technology, and that enhancing the education of practitioners is fundamental.

Keywords: Green Software, energy optimization, carbon footprint.

1. INTRODUCTION

The software, by executing complex algorithms that need massive data processing, requires computational resources that translate into high energy consumption, which, in effect, increases CO_2 emissions associated with energy generation. Energy consumption and carbon emissions have increased noticeably, driven by the evolution of computing technologies. This has increased environmental concerns, necessitating the definition of short-term action frameworks. Projections suggest that the Information and Communications Technology (ICT) sector could account for 14% of global carbon emissions by 2040, indicating a need to develop sustainable practices (Tiwari *et al.,* 2023), in line with the fundamental green computing principles outlined in (Murugesan, 2008), where early adoption of responsible technology practices is described.

As software applications become increasingly resource-intensive, especially in the context of cloud computing and machine learning, the training of advanced artificial intelligence models contributes significantly to the increase in emissions (Xu *et al.,* 2023). In this context, green software practices seek to minimize environmental impact throughout their lifecycles, while sustainable software development integrates ecological considerations with conventional performance metrics. Consequently, incorporating green software practices into the Software Development Lifecycle (SDLC) is presented as a strategy for reducing the carbon footprint in computing (Gupta & Gupta, 2025). Although the GREENSOFT model, introduced in 2011 (Naumann *et al.,* 2011), was pioneering in integrating sustainability into software development, the rise of resource-intensive technologies like Cloud and AI suggest the need to expand its guidelines to meet the current challenges.

Recent work explores multiple ways to improve energy efficiency and reduce emissions. The Remote-Model View Remote-View-Model (RMVRVM) paradigm, for example, optimizes energy usage by offloading data processing to server resources (Singh *et al.,* 2024). Efficient parameter fine-tuning (PEFT) for large language models (LLMs) also helps developers select energy-efficient libraries (Gupta & Gupta, 2025). Furthermore, the substantial emissions of LLMs suggest the need for standardized metrics to measure their environmental impact (Everman *et al.,* 2023). Despite these advances, many organizations limit sustainability to isolated phases of development, lacking a cohesive approach (Jayanthi *et al.,* 2021). Furthermore, the absence of universal sustainability metrics hinders broader adoption (Tiwari *et al.,* 2023).

In parallel, research on machine learning (AutoML) reveals that resource-intensive processes can be optimized to improve environmental outcomes (Castellanos-Nieves & García-Forte, 2023). However, frameworks often fail to integrate sustainability, delaying the adoption of green practices (Ahmad Ibrahim *et al.,* 2022; Jayanthi *et al.,* 2021). Despite advances in understanding the environmental impact of software, a gap persists in the comprehensive integration of this knowledge into practical, industry-wide frameworks.

This work examines the current literature on green software practices, emphasizing a comprehensive sustainability framework. Its primary objective is to explore how integrating green software practices throughout the software development lifecycle can reduce the environmental impact of computing. By establishing a holistic analysis, we hope to contribute a comprehensive overview of existing practices and their effectiveness in mitigating carbon emissions.

The following sections detail the methodology and literature review, outline the clustering and trend analysis, and synthesize the findings into a comprehensive sustainability framework.

2. METHODOLOGY

The research methodology for this project was carried out in various stages, from identifying problems to analyzing industry market trends.

a. Definition of Problem and Scope

The primary goal of this work was established as: to review existing literature in the field of these practices in lowering the carbon footprint of ICTs. Since the topic of discussion is recent regarding new technologies effects, the scope of the study was restricted to publications from 2019 to 2025 to choose applicable works. The research question we are interested in is: *What recent green software practices have been proposed to reduce the environmental impact associated with energy consumption and carbon emissions in computing technologies?*

b. Literature Review

We applied the PRISMA approach to this step. We chose Scopus meta-database, as it indexes peer-reviewed computer science and environmental science papers. The search query is constructed iteratively to span the domain's terminological diversity. The latest available version utilized is: "*TITLE-ABS-KEY(('green software' OR 'sustainable software') AND ('carbon footprint' OR 'carbon emissions' OR 'CO2 emissions' OR 'emissions reduction')) AND PUBLICATION > 2018*".

The inclusion (topic relevance, peer-reviewed journal article publication, empirical or modeling research) and exclusion (books, conference proceedings, articles without institutional access) criteria are utilized to ascertain the quality and relevance of literature analyzed.

It is important to note, that while this study primarily focuses on works published between 2019 and 2025, some earlier references have been included due to their pioneering and foundational nature in the field of sustainable software.

c. Clustering and Trend Analysis

After the literature is gathered and compiled, data are drawn into a standard form so that we can recognize important variables and trends within. Out of this analysis, the most critical strategies and knowledge base deficiencies are identified.

d. Synthesis and Integration of Findings

From the extracted information, a thematic synthesis of studies is selected. This stage integrates the various approaches and methodologies identified into a conceptual framework for understanding green software practices. The synthesis aims to articulate a holistic vision to propose comprehensive guidelines for IT sustainability.

3. RESULTS

a. Literature Review

Following the PRISMA methodology, 29 studies were initially identified through the Scopus database, of which 27 were found to be accessible and 4 were excluded for lack of relevance to the topic, lack of peer review, or inability to fully access. This process resulted in 23 articles that meet the inclusion criteria, focusing on the carbon footprint of green software, and that address empirical or modeling studies published between 2019 and 2025, consistent with the objectives of this review (see Figure 1).

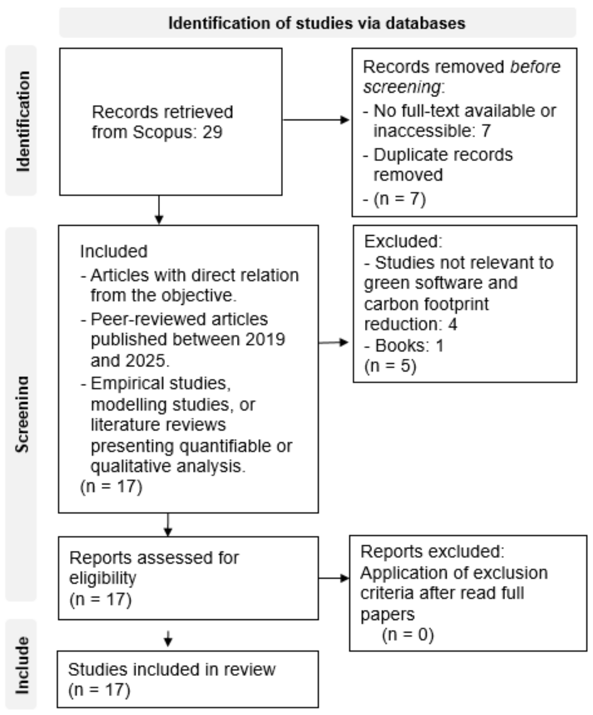
Identification of studies via databases
Identification
Records retrieved from Scopus: 29
Records removed *before screening*:
- No full-text available or inaccessible: 7
- Duplicate records removed
- (n = 7)
Screening
Included
- Articles with direct relation from the objective.
- Peer-reviewed articles published between 2019 and 2025.
- Empirical studies, modelling studies, or literature reviews presenting quantifiable or qualitative analysis.
(n = 17)
Excluded:
- Studies not relevant to green software and carbon footprint reduction: 4
- Books: 1
(n = 5)
Reports assessed for eligibility
(n = 17)
Reports excluded: Application of exclusion criteria after read full papers
(n = 0)
Include
Studies included in review
(n = 17)

Figure 1. *PRISMA methodology application results.*

b. Clustering and Trend Analysis

After gathering and compiling the literature through the PRISMA methodology, we standardized the data by extracting key variables—methodology, key findings, limitations, and sustainability impact—from each study. This process resulted in a detailed table. To make the analysis accessible, we derived a concise taxonomy (see Table 1) that groups the studies into four clusters, into three categories.

The trend analysis across these clusters shows several recurring themes:

- Generalizability Issues: Many studies are context-specific (e.g., limited to particular platforms, datasets, or regions), restricting broader applicability.

- Need for adoption: There is a strong call for unified, real-time metrics to accurately assess energy consumption and carbon emissions.

- Integration into the SDLC: There is a growing trend toward embedding sustainability practices—both technical (energy-efficient coding, measurement tools) and organizational (CSR policies)—throughout the software development lifecycle.

Table 1. *General taxonomy of the analyzed works.*

Category	Subcategory	Definition	works
Sustainable Software Development Practices	Energy Efficiency Techniques	Methodologies that improve energy efficiency in software development through clean code practices and energy testing, reducing carbon footprint.	(Ahmad Ibrahim *et al.*, 2022; Gupta & Gupta, 2025; Inayatullah *et al.*, 2024; Rios, 2024; Singh *et al.*, 2024; Vergallo *et al.*, 2024; Wedyan *et al.*, 2023)
	Carbon Footprint Measurement and Assessment	Tools and frameworks for evaluating the carbon impact of software, supporting sustainable decision-making.	(Castano *et al.*, 2023; Everman *et al.*, 2023; Mehra *et al.*, 2022; Nayak *et al.*, 2024; Tiwari *et al.*, 2023)
Green AI and Machine Learning Practices	Green AI and Machine Learning Practices	Sustainable AI practices that promote efficient model training and assess their carbon impact.	(Castellanos-Nieves & García-Forte, 2023; Vergallo & Mainetti, 2024; Xu *et al.*, 2023)
Sustainable Organizational Practices	Corporate Social Responsibility in Software Development	Strategies for software companies to integrate sustainability into their corporate policies, aligning business goals with environmental concerns.	(Calero *et al.*, 2019; Jayanthi *et al.*, 2021)

c. Synthesis and Integration of Findings

i. Sustainable Software Development Practices

There are different methodologies and strategies aimed at minimizing the environmental impact of software development throughout its lifecycle. This category is pivotal in analyzing recent research focused on green software practices, particularly their potential to significantly reduce the carbon footprint associated with computing technologies.

Energy Efficiency Techniques

The reviewed studies are organized into three large conceptual categories: architectural models, specific energy efficiency techniques, and general methodological factors.

- *Architectural Models*: The RMVRVM paradigm delegates processing to the server, thereby reducing energy consumption on client devices and improving overall performance (Singh *et al.,* 2024). This model also recognizes limitations related to the generalization of results across diverse scenarios and technological contexts.

- *Specific Energy Efficiency Techniques*: The Batch Operations (BO) pattern effectively mitigates the energy consumption of communication peripherals by grouping operations. However, it presents significant limitations due to its limited validation on specific devices (Vergallo *et al.,* 2024). On the other hand, advanced techniques such as the Parameter-Efficient Fine-Tuning (PEFT) approach promote energy sustainability in Large Language Models (LLM) by consciously selecting libraries based on energy and functional criteria (Gupta & Gupta, 2025). Furthermore, from the programming languages perspective, comparative research shows that Java is significantly more energy-efficient than Python, emphasizing the importance of these factors in the early stages of development (Rios, 2024).

Likewise, (Inayatullah *et al.,* 2024) demonstrates the importance of applying clean code practices as a strategy to optimize software energy efficiency, empirically demonstrating its positive impact on reducing energy consumption and facilitating code maintenance and understanding.

- *General Methodological Factors*: In (Wedyan *et al.,* 2023), its described a methodology that integrates energy consumption testing within existing software testing frameworks, allowing energy inefficiencies to be identified from an early stage without incurring significant additional costs. However, it faces challenges in generalization due to the variability in energy consumption across different hardware environments. The effective incorporation of green software practices also entails integrating sustainability into the Software Development Life Cycle (SDLC). However, agile methodologies promote rapid feedback cycles, measuring sustainability indicators in each sprint can expose energy inefficiencies in good time. In (Dick *et al.,* 2013) propose adapting such methods to introduce sustainability metrics in each iteration of development, enabling the identification and systematic correction of high energy consumption practices and maintaining a continuous improvement cycle in reducing the carbon footprint.

  The literature supports the adoption of comprehensive frameworks that facilitate this integration (Ahmad Ibrahim *et al.,* 2022). In particular, (*Green in Software Engineering*, s. f.) raised this need more than a decade ago. Despite their comprehensive approach and early proposal, widespread adoption of these guidelines remains limited, highlighting the persistence of technical and organizational barriers, like a specific organizational structure dedicated to sustainability that can overcome existing barriers related to limited specialized knowledge and lack of institutional support (Inayatullah *et al.,* 2024; Jayanthi *et al.,* 2021).

Carbon Footprint Measurement and Assessment

For the measurement and assessment of carbon footprints in software development, the reviewed studies are organized into two key categories: specific measurement approaches and methodological frameworks.

- *Specific approaches to carbon footprint measurement*: In (Tiwari *et al.,* 2023), it describes the need to develop effective methodologies that integrate environmental considerations throughout the SDLC, considering specific contexts and coding practices. In (Mehra *et al.,* 2022), it introduces the notion of the *green quotient* in the Project Green Quotient (PGQ), which allows developers to assess the sustainability of their projects at different phases of the SDLC, facilitating informed decision-making that can significantly reduce the carbon footprint and improve operational performance.

In specific software domains such as Large Language Models (LLMs) (Everman *et al.,* 2023). The Software Carbon Intensity (SCI) specification is used to measure and compare the environmental impact of different models, challenging the idea that larger and more energy-intensive models are inherently better in terms of performance, opening the door to more sustainable models without penalizing accuracy (Everman *et al.,* 2023). Another study, focused on the Hugging Face community, describes gaps in transparency and carbon emissions awareness in Machine Learning (ML) models, proposing classification systems based on emissions reporting and energy efficiency practices, incentivizing greater transparency and sustainability (Castano *et al.,* 2023).

- *Innovative Methodological Frameworks:* The introduction of the Green Continuous Testing (Green CT) approach proposes optimizing software testing practices by focusing on executing only essential cases, thereby reducing automated executions without compromising software quality, shifting toward the effective integration of sustainability metrics into software development and testing processes (Nayak *et al.,* 2024).

The community's effort has been significant in this field; however, the main weaknesses are focused on the generalization due to the specificity of the tools, technological contexts, and scenarios analyzed. Furthermore, challenges such as the reliance on self-reported data potentially affect the accuracy and consistency of the assessments (Castano *et al.,* 2023; Mehra *et al.,* 2022). In Table 2, we described the most relevant aspect the works in this category.

Table 2. *Critical evaluation of sustainable energy optimization and carbon footprint assessment techniques in software development.*

Strategy/Technique	Valuable Attribute	Critical Vulnerability	Impact and Implications	Qualitative Impact Assessment
Paradigm RMVRVM (Singh *et al.*, 2024)	Effective proposal to reduce energy consumption on client devices by offloading processing.	High dependency on network infrastructure and remote server quality, limiting applicability in regions with constrained infrastructure.	The strategy could exacerbate the technological gap between developed and developing regions.	Medium-High
Batch Operations (BO) (Vergallo *et al.*, 2024)	Effectively reduces consumption in communication peripherals by grouping operations, representing a simple and cost-effective best practice.	Validation limited to specific devices and restricted capacity for dynamic adaptation to changes in usage patterns.	Lacks adaptive flexibility and may lose relevance over time due to rapid changes in hardware and usage patterns.	Medium

PEFT (Parameter-Efficient Fine-Tuning) (Gupta & Gupta, 2025)	Promising technique for reducing energy consumption by selecting specific libraries, particularly relevant for large language models (LLMs).	Requires a high level of technical expertise and additional resources for continuous library selection based on updated energy criteria.	May be limited to organizations with sufficient technical resources, potentially marginalizing SMEs and community projects.	High
Java vs. Python (Rios, 2024)	Clearly identifies critical differences in energy efficiency by programming language, offering clear criteria for early technical decision-making.	Risks oversimplifying technical decisions by focusing solely on energy consumption without considering complexity, performance, or maintainability.	A language choice based solely on energy efficiency may incur unforeseen technical debt.	High
Clean Code Approaches (Inayatullah *et al.*, 2024)	Empirically demonstrates that adherence to good coding practices positively affects energy efficiency and software maintainability.	Effective adoption is highly dependent on organizational commitment and intensive team training.	Insufficient investment in training could significantly diminish its effectiveness in real-world scenarios.	High
Integrated Energy Consumption Testing (Wedyan *et al.*, 2023)	Enables early detection of energy issues integrated within regular testing processes.	Practical challenges due to hardware variability and the complexity of real operational environments may hinder generalization.	Without robust standardization methods, its application might remain confined to highly controlled and specific contexts.	Medium-High
Agile with Sustainability Metrics (Dick *et al.*, 2013)	Innovatively incorporates environmental metrics into agile methodologies, facilitating rapid corrective actions.	Possible organizational resistance due to the perceived increase in administrative and operational burden.	Success will critically depend on a profound cultural transformation within organizations, extending beyond mere technical adjustments.	High

ii. Green AI and Machine Learning Practices

The integration of energy efficiency into the evaluation of deep learning models is transforming how these systems are analyzed, adding environmental indicators to traditional metrics like accuracy and F1 score. In (Xu *et al.*, 2023), the impact of architectural complexity on energy consumption and CO_2 emissions is studied using metrics that quantify these effects. Some energy efficiency metrics are defined as:

- Total Energy Consumption:

$$E_{total} = \sum_{i=1}^{n} P_i \cdot \Delta t_i$$

where P_i is the power consumed during each time interval Δt_i. Expressed in kilowatt-hours (kWh), this measure estimates the energy cost of training the model.

- CO_2 Emissions: Derived from the total energy consumption using an emission factor, the calculation is as follows:

$$CO_2 = E_{total} \times Emission\ Factor$$

where the emission factor (in $kgCO_2e/kWh$) depends on the energy source.

- Energy Efficiency Ratio (EER): This metric relates the model's accuracy (Acc) to its energy consumption, providing an indication of how many accuracy points are achieved per unit of energy consumed.

$$EER = \frac{Acc}{E_{total}}$$

- Energy-Delay Product (EDP): Adapted for deep learning by considering the training time (T). This metric balances energy efficiency with the temporal performance of the training process:

$$EDP = E_{total} \times T$$

Integration in Development and AutoML: In the field of AutoML, (Castellanos-Nieves & García-Forte, 2023) demonstrates that hyperparameter tuning can optimize not only performance but also energy consumption. The use of AI in environmental management has also shown potential to optimize the monitoring and conservation of natural resources (Miranda *et al.,* 2024). Metrics such as E_{total}, CO_2 emissions, and EER are used alongside derived indicators that assess energy efficiency per iteration or as a percentage improvement in accuracy. Implementation through the Scikit-learn library facilitates the practical integration of these parameters into the optimization process, contributing to more sustainable modeling systems.

Carbon-aware strategies and their evaluation: In (Vergallo & Mainetti, 2024), it introduces innovative strategies so-called *Flexible Start and Pause & Resume,* that rely on monitoring carbon intensity and setting energy consumption thresholds. These techniques enable training to begin or pause based on favorable environmental conditions by defining thresholds I_{min} and I_{max}, such that:

$$Carbon\ condition\ I(t) \leq I_{min}\ or\ I(t) \geq I_{max}$$

Applicable in areas like natural language processing and computer vision, these strategies have shown promising results. However, their effectiveness varies based on geographical regions and computational environments, suggesting the need for broader validation across diverse contexts.

Some limitations are noted, the findings in (Xu *et al.,* 2023) depend on specific datasets and architectures; (Castellanos-Nieves & García-Forte, 2023) focuses on a limited number of algorithms; and (Vergallo & Mainetti, 2024) evaluates only a few workload types. These aspects show the importance of continued research to adopt more comprehensive metrics and strategies that are applicable to a wider range of scenarios and domains. Table 3 provide an overview of the different strategies for integrating energy efficiency into AI and machine learning.

Table 3. *Critical evaluation of green ai and machine learning practices.*

Strategy/Technique	Valuable Attribute	Critical Vulnerability	Impact and Implications	Qualitative Impact Assessment
Project Green Quotient (PGQ) (Mehra *et al.*, 2022)	Enables evaluation of sustainability across different SDLC phases, facilitating informed decision-making.	Significant reliance on self-reported data, which could affect the accuracy and reliability of the metrics.	May face adoption barriers due to a lack of transparency and consistency in internal reporting.	High
Software Carbon Intensity (SCI) (Everman *et al.*, 2023)	Valuable tool for measuring and comparing the environmental impact of specific models such as LLMs.	Lack of consensus on universally applicable emission factors, limiting valid comparisons across different contexts.	Standardization challenges could slow its mass and comparative adoption.	High
Green Continuous Testing (Green CT) (Nayak *et al.*, 2024)	Optimizes testing practices by reducing unnecessary executions while maintaining quality.	Limitations in dynamic and complex scenarios where frequent code changes might require additional execution.	Without robust dynamic adaptation, its efficiency could be significantly reduced in agile enterprise contexts.	Medium-High

iii. Sustainable Organizational Practices

As part of this taxonomy, we decided to include the Sustainable Organizational Practices category after observing that, in the reviewed literature, studies were found specifically focused on corporate strategies to reduce the environmental impact of software development. Approaches that integrate sustainability into software activities, enabling companies to adopt more environmentally responsible policies and processes.

Corporate Social Responsibility in Software Development

Corporate Social Responsibility (CSR) provides an essential framework for incorporating environmental objectives into the planning and execution of software projects. However, in (Calero *et al.*, 2019) suggests that most technology companies focus their sustainability efforts on hardware, leaving the adoption of Green Software practices in the background. According to the analysis of several companies' CSR policies and their alignment with the UN Sustainable Development Goals, the development of energy-efficient software is still not given the same priority as hardware initiatives. It demonstrates the need to intensify attention to software sustainability throughout its lifecycle, both in design and in implementation and maintenance. Similarly, in (Schott & Ovtcharova, 2024) demonstrate that to successfully integrate sustainability into software development, explicit support from top management and an organizational culture focused on long-term environmental goals is needed. By integrating these considerations into the scope of CSR, organizations can reduce their carbon footprint and strengthen their commitment to environmental protection. Furthermore, the adoption of best practices for more efficient and less polluting software not only responds to growing social demands but also provides competitive advantages in a market that is increasingly sensitive to sustainability.

Table 4, offers a critical assessment of strategies for integrating sustainability into corporate software development practices.

Table 4. *Critical evaluation of sustainable organizational practices in software development.*

Strategy/Technique	Valuable Attribute	Critical Vulnerability	Impact and Implications	Qualitative Impact Assessment
Integration of CSR in Software Development (Calero *et al.*, 2019)	Clarity in highlighting the importance of integrating sustainability into corporate policies.	Low prioritization compared to hardware initiatives, revealing an internal strategic weakness in companies.	Without strong strategic change from top management, initiatives may remain mere declarations without real impact.	High
Explicit Managerial Support (Schott & Ovtcharova, 2024)	Emphasizes that success depends on clear and explicit commitment from top management.	Often difficult to achieve due to resistance to cultural change and short-term economic priorities.	Could lead to stagnation of sustainable initiatives in early phases without continuous, explicit support.	High

4. DISCUSSION

The reported advancements reflect, in the software context, some level of attention to energy efficiency and carbon footprint measurement; however, the heterogeneity of software, the languages, architectures, execution contexts, restricts the proposal of universal methodologies. Most studies outline very specific approaches, such as optimizations in a specific language or techniques focused solely on AI models, questioning their applicability in complex enterprise environments. Even emission measurement metrics lack consensus as they depend on local emission factors and self-declarations that are difficult to validate, hindering reliable comparisons between projects.

Furthermore, there is a notable lag in incorporating sustainability into the corporate social responsibility of software development companies. The limited focus on the design and development phase, compared to hardware initiatives, reflects the low priority assigned to green practices in product creation. Scenario diversity will continue to hinder the widespread adoption of sustainable strategies.

The deficiency in the adoption of green practices indicates the need to train professionals with sustainability skills from the early stages of their training. In (Moreira *et al.,* 2024), this gap is precisely addressed by proposing a roadmap to integrate sustainability across software engineering curricula, emphasizing that education can play a key role for future generations of developers to implement and enhance current strategies for reducing environmental impact.

5. CONCLUSIONS

The literature shows efforts in utilizing green software practices that contribute to reducing CO_2 emissions and saving energy. However, the absence of transversal methodologies capable of adapting to the diversity of languages, frameworks, and platforms keeps the field in a low maturity stage. In the case of AI, training optimization lays the foundations to the way for more efficient models, but its adoption is still limited by the lack of unified metrics and standardized procedures.

On the other hand, although there are tools and methods for measuring the carbon footprint, such as SCI or the green quotient, a clear consensus regarding their implementation and validation is still needed, making it difficult to compare results between projects. Combined with this, the integration of sustainability into corporate policy is still limited, with corporate social responsibility strategies focused almost exclusively on hardware.

For the industry, these gaps point to the need for clearer guidelines that include both the definition of homogeneous emission indicators and development guides that can be implemented in diverse scenarios. Advancing awareness and specific training of teams, along with establishing corporate structures dedicated to sustainability, appears to be an important step.

6. REFERENCES

Ahmad Ibrahim, S. R., Yahaya, J., & Sallehudin, H. (2022). Green Software Process Factors: A Qualitative Study. En Sustainability (Switzerland) (Vol. 14, Número 18). https://doi.org/10.3390/su141811180

Calero, C., García-Rodríguez De Guzmán, I., Moraga, M. A., & García, F. (2019). Is software sustainability considered in the CSR of software industry? En International Journal of Sustainable Development and World Ecology (Vol. 26, Número 5, pp. 439-459). https://doi.org/10.1080/13504509.2019.1590746

Castano, J., Martinez-Fernandez, S., Franch, X., & Bogner, J. (2023). Exploring the Carbon Footprint of Hugging Face's ML Models: A Repository Mining Study. En International Symposium on Empirical Software Engineering and Measurement. https://doi.org/10.1109/ESEM56168.2023.10304801

Castellanos-Nieves, D., & García-Forte, L. (2023). Improving Automated Machine-Learning Systems through Green AI. En Applied Sciences (Switzerland) (Vol. 13, Número 20). https://doi.org/10.3390/app132011583

Dick, M., Drangmeister, J., Kern, E., & Naumann, S. (2013). Green software engineering with agile methods. 2013 2nd International Workshop on Green and Sustainable Software (GREENS), 78-85. https://doi.org/10.1109/GREENS.2013.6606425

Everman, B., Villwock, T., Chen, D., Soto, N., Zhang, O., & Zong, Z. (2023). Evaluating the Carbon Impact of Large Language Models at the Inference Stage. En Conference Proceedings of the IEEE International Performance, Computing, and Communications Conference (pp. 150-157). https://doi.org/10.1109/IPCCC59175.2023.10253886

Green in Software Engineering. (s. f.). Recuperado 15 de marzo de 2025, de https://springerlink.unalproxy.elogim.com/book/10.1007/978-3-319-08581-4

Gupta, S., & Gupta, A. (2025). Advancing Carbon-Efficient Software Development: A Sustainable Path Forward. En ICDCN 2025—Proceedings of the 26th International Conference on Distributed Computing and Networking (pp. 325-330). https://doi.org/10.1145/3700838.3703670.

Henriquez, C., Pérez, J. D. R., & Sanchez-Torres, . (2024). Aplicaciones de la inteligencia artificial en el monitoreo y conservación ambiental: Una revisión exploratoria. REVISTA AMBIENTAL AGUA, AIRE Y SUELO, 15(2), Article 2. https://doi.org/10.24054/raaas.v15i2.3189

Inayatullah, Widhiarso, W., & Stiawan, D. (2024). Energy Efficiency in Sustainable Software Development: Clean Code Approaches. En ICECOS 2024—4th International Conference on Electrical Engineering and Computer Science, Proceeding (pp. 101-105). https://doi.org/10.1109/ICECOS63900.2024.10791178

Jayanthi, H., Kulkarni, A., Patil, A. E., & S V, S. (2021). An Organizational Structure for Sustainable Software Development. En Proceedings—2021 3rd International Conference on Advances in Computing, Communication Control and Networking, ICAC3N 2021 (pp. 1539-1542). https://doi.org/10.1109/ICAC3N53548.2021.9725722

Mehra, R., Sharma, V. S., Kaulgud, V., Podder, S., & Burden, A. P. (2022). Towards a Green Quotient for Software Projects. En Proceedings—International Conference on Software Engineering (pp. 295-296). https://doi.org/10.1109/ICSE-SEIP55303.2022.9793921

Moreira, A., Lago, P., Heldal, R., Betz, S., Brooks, I., Capilla, R., Coroamă, V. C., Duboc, L., Fernandes, J. P., Leifler, O., Nguyen, N.-T., Oyedeji, S., Penzenstadler, B., Peters, A.-K., Porras, J., & Venters, C. C. (2024). A Roadmap for Integrating Sustainability into Software Engineering Education. ACM Trans. Softw. Eng. Methodol. https://doi.org/10.1145/3708526

Murugesan, S. (2008). Harnessing Green IT: Principles and Practices. IT Professional, 10(1), 24-33. IT Professional. https://doi.org/10.1109/MITP.2008.10

Naumann, S., Dick, M., Kern, E., & Johann, T. (2011). The GREENSOFT Model: A reference model for green and sustainable software and its engineering. Sustainable Computing: Informatics and Systems, 1(4), 294-304. https://doi.org/10.1016/j.suscom.2011.06.004

Nayak, K., Route, S., Sundararajan, M., Jain, A., & Shashidhar, R. (2024). Sustainable continuous testing in DevOps pipeline. En 2024 1st International Conference on Communications and Computer Science, InCCCS 2024. https://doi.org/10.1109/INCCCS60947.2024.10593566

Rios, R. A. B. (2024). Best practices for green (sustainable) software development using artificial intelligence; [Mejores prácticas para el desarrollo de software verde (sostenible) utilizando inteligencia artificial]. En European Public and Social Innovation Review (Vol. 9). https://doi.org/10.31637/epsir-2024-436

Schott, V., & Ovtcharova, J. (2024). Integrating Sustainability into Software Development: A Global Categorization. En Finance and Law in the Metaverse World (pp. 59-69). Springer, Cham. https://doi.org/10.1007/978-3-031-67547-8_6

Singh, L., Tiwari, S., & Srivastava, S. (2024). A study on creating energy efficient cloud-connected user applications using the RMVRVM paradigm. En Journal of Systems and Software (Vol. 213).

https://doi.org/10.1016/j.jss.2024.112033

Tiwari, R., Thombre, S., Patel, D., Tilokani, K., & Amritkar, S. (2023). Evaluating Code Sustainability: A Comprehensive Study of Metrics and Tools. En 2023 IEEE Engineering Informatics, EI 2023. https://doi.org/10.1109/IEEECONF58110.2023.10520542

Vergallo, R., Cagnazzo, A., Mele, E., & Casciaro, S. (2024). Measuring the Effectiveness of the 'Batch Operations' Energy Design Pattern to Mitigate the Carbon Footprint of Communication Peripherals on Mobile Devices. En Sensors (Vol. 24, Número 22). https://doi.org/10.3390/s24227246

Vergallo, R., & Mainetti, L. (2024). Measuring the Effectiveness of Carbon-Aware AI Training Strategies in Cloud Instances: A Confirmation Study. En Future Internet (Vol. 16, Número 9). https://doi.org/10.3390/fi16090334

Wedyan, F., Morrison, R., & Abuomar, O. S. (2023). Integration and Unit Testing of Software Energy Consumption. En 2023 10th International Conference on Software Defined Systems, SDS 2023 (pp. 60-64). https://doi.org/10.1109/SDS59856.2023.10329262

Xu, Y., Martínez-Fernández, S., Martinez, M., & Franch, X. (2023). Energy Efficiency of Training Neural Network Architectures: An Empirical Study. En Proceedings of the Annual Hawaii International Conference on System Sciences (Vols. 2023-January, pp. 781-790). https://scopus.unalproxy.elogim.com/inward/record.uri?eid=2-s2.0-85152121850&partnerID=40&md5=713c8b9d56460af43262e0ea53dc1cb3