The Impacts of Artificial Intelligence throughout the Software Development Life-cycle on Sustainability: A Mixed-method Study

Fazla Iqbal
LUT University
Lappeenranta, Finland
miffazl@gmail.com

Muhammad Asif Khan
LUT University
Lappeenranta, Finland
Asif.khan@lut.fi

Oshani Weerakoon
University of Turku
Turku, Finland
Osweer@utu.fi

Shola Oyedeji
LUT University
Lappeenranta, Finland
Shola.Oyedeji@lut.fi

Jari Porras
LUT University
Lappeenranta, Finland
Jari.porras@lut.fi

Abstract

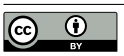
Artificial Intelligence (AI) is increasingly embedded in the software development life cycle (SDLC), reshaping the design, implementation, and maintenance of software. However, the impacts of sustainability remain insufficiently explored. This study investigates how AI affects sustainability across SDLC phases by integrating insights from academic research and industrial practice. We applied a mixed-method approach combining a systematic review of 64 peer-reviewed studies with 27 semi-structured interviews conducted across 12 countries. The analysis covers five dimensions of sustainability: environmental, economic, social, individual, and technical, highlighting both positive "handprints" and negative "footprints." The results indicated that AI is most widely adopted during implementation and testing, with limited use in the requirements, design, and deployment phases. AI improves productivity, code quality, and testing efficiency. However, major challenges persist, such as unmonitored energy consumption, vendor dependency, bias, skill degradation, and AI-related defects. Based on these findings, we propose actionable recommendations for integrating sustainability checkpoints, such as energy and cost monitoring, model size optimization, human oversight, and AI-aware quality reviews, into AI-assisted workflows. These measures aim to shift from incidental efficiency to intentional sustainability throughout the SDLC.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; • **Software and its engineering** → **Software development process management**; • **Social and professional topics**; • **General and reference** → *Empirical studies*;

Keywords

Artificial intelligence, Software development lifecycle, Software sustainability, AI in SDLC, Footprint, Handprint



GREENS '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2381-0/26/04
https://doi.org/10.1145/3786148.3788622

ACM Reference Format:
Fazla Iqbal, Muhammad Asif Khan, Oshani Weerakoon, Shola Oyedeji, and Jari Porras. 2026. The Impacts of Artificial Intelligence throughout the Software Development Life-cycle on Sustainability: A Mixed-method Study. In *10th International Workshop on Green and Sustainable Software (GREENS '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3786148.3788622

1 Introduction

The growing importance of software systems as essential drivers of economic growth and innovation has positioned the Software Development Lifecycle (SDLC) as the core framework [32] [51]. In this environment, Artificial Intelligence (AI), including machine learning based and Generative AI techniques (e.g., Chat-GPT [1], GitHub Copilot [2]), is profoundly transforming software engineering by automating processes and introducing novel concepts throughout all phases of the SDLC [46, 51]. However, these advancements introduce Major sustainability challenges that extend beyond technical performance. In this study, we consider Sustainability in software engineering as broadly framed across five dimensions: environmental, economic, social, individual, and technical. [14]. AI will impact all these dimensions. AI's environmental foot- print includes energy consumption and carbon emissions from model training and inference [14]. Economic sustainability concerns high infrastructure costs and Return On Investment (ROI) [32]. Social aspects involve inclusivity and bias mitigation [40], while individual well-being addresses maintainability and managing technical debt in AI-augmented systems [3]. Few studies have systematically explored AI's broader sustainability impacts across the entire SDLC. This study addresses this gap by mapping how AI is integrated across SDLC phases and by comparing academic perspectives with real-world industrial practices to identify sustainability opportunities and risks.

2 Background & Related Work

2.1 AI Integration in the Software Development LifeCycle (SDLC)

The Software Development Life Cycle encompasses all phases of software creation and maintenance. It is defined by standards such

[1] https://chatgpt.com
[2] https://copilot.microsoft.com/

as ISO/IEC 12207 [3], which lays out the processes and tasks required for software development and upkeep. Key phases typically include requirement analysis, design, implementation (coding), testing, deployment, and maintenance [9].

Artificial intelligence approaches are increasingly used in SDLC phases to automate or support processes. AI-powered tools, for example, can help with requirements engineering tasks such as requirements gathering, analysis [41, 52, 57], and code completion throughout implementation [36], as well as automating testing [8] and debugging. AI has been found to increase developer productivity, improve code quality, and accelerate development cycles. Automated code creation, intelligent debugging and bug prediction, predictive software maintenance, and data-driven decision-making during the development process are key areas of impact. Recent empirical studies have explored how pre-trained models on platforms like Hugging Face assist with software engineering tasks, including code generation, testing, and documentation [23]. Moreover, AI "co-pilot" systems, such as Microsoft's Copilot [4], ChatGPT's Codex [5], and Amazon Q Developer [6], can eliminate manual effort and errors, hence speeding up the development process.

2.2 Sustainability in Software Engineering & SDLC

Sustainability has become a fundamental topic in software engineering because systems have a greater impact on the environment, economy, and society. It is commonly described as addressing current needs without affecting the ability of future generations to meet their own [30]. According to the Karlskrona Manifesto for Sustainability Design, sustainability is generally viewed across five dimensions: *environmental, social, economic, individual*, and *technological* [11].

From an *environmental* perspective, software development contributes to energy consumption and carbon emissions [34, 37] through activities such as data center operations, resource-intensive computation, and hardware production [33]. For example, the ICT sector is estimated to contribute about 1.4% to 3.9% of global greenhouse gas emissions [21]. However, it also possesses the potential to reduce emissions in other sectors by up to 15% by enabling technologies [18], which is seven times greater than its impact. The *social* and *individual* dimensions of sustainability emphasize well-being, inclusivity, and long-term engagement of users and developers [38]. This includes fostering inclusive access, minimizing cognitive overload, and supporting positive work environments for development teams. *Economic sustainability* relates to cost efficiency and long-term value creation, recognizing that sustainable software can reduce operational expenses, extend product lifespans, and improve return on investment [28]. *Technical sustainability* focuses on maintainability, adaptability, and resilience of systems, ensuring they can evolve without excessive rework or degradation in quality [28].

Earlier work identified persistent challenges in integrating sustainability into the SDLC [35]. More recent research, including the introduction of the Green AI paradigm, highlights that sustainability considerations remain unevenly addressed in AI-intensive software development despite increased awareness [27]. According to Betz et al. [12], tools and methods to support sustainability-aware decisions in software engineering practice are lacking. Furthermore, explicit sustainability standards, insufficient industry knowledge, and the absence of defined measures for sustainability evaluation have contributed [38].

2.2.1 Footprint & Handprint. While sustainability has typically focused on eliminating harm, recent research has introduced the complementary concepts of *footprint* and *handprint* to reflect the negative and positive impacts, respectively. A *footprint* measures the negative impacts of software operations, including carbon emissions, water consumption, and resource depletion, throughout its lifecycle [44]. A *handprint* indicates the beneficial contributions of software or organizations toward environmental or social progress [5, 24]. Guillaume et al. [24] define the handprint as a constructive metric that "encourages actions with positive impacts," complementing the footprint's emphasis on harm mitigation. Kühnen et al. [29] link handprints to the UN Sustainable Development Goals (SDGs) [7], underlining the importance of organizations not only reducing negative impacts but also actively contributing to societal good.

Applying these concepts to the SDLC allows for a dual-lens assessment: minimizing the software footprint (e.g., through energy-efficient architectures and sustainable deployment strategies) while maximizing its handprint (e.g., by developing software that supports sustainability goals in other domains). For example, AI-based optimization tools may increase energy consumption during model training, but provide long-term benefits by decreasing operational inefficiencies.

3 Methodology

This study used a mixed-methods qualitative approach, combining a systematic literature review (SLR) [13] using semi-structured interviews with practitioners. The objective was to investigate how Artificial Intelligence (AI) is integrated into the Software Development Life Cycle (SDLC) and to examine the resulting sustainability implications. Both the SLR and the interviews were designed to address the same research questions throughout the SDLC phases and sustainability dimensions. The SLR captured the state of the art from academic research, while the interviews provided the state of practice from the industry. Integrating both sources offered a balanced understanding of AI adoption patterns, sustainability perceptions, and practical challenges across SDLC phases.

3.1 Research Questions

This study is guided by one main research question and two subquestions. These questions explore the integration of Artificial Intelligence (AI) into the Software Development Life Cycle (SDLC) and examine how this integration influences software sustainability. The aim is to bridge academic and industrial perspectives by analyzing both the theoretical understanding of sustainability in research and its practical application within organizations.

[3] https://www.iso.org/standard/63712.html
[4] https://copilot.microsoft.com/
[5] https://chatgpt.com/codex
[6] https://aws.amazon.com/q/developer/
[7] https://unric.org/en/united-nations-sustainable-development-goals/

- **RQ1:** How does current research conceptualize sustainability within the integration of Artificial Intelligence (AI) into the Software Development Life Cycle (SDLC)?
- **SRQ1:** What are the current use cases of Artificial Intelligence (AI) across the different phases of the Software Development Life Cycle (SDLC), and what benefits (handprints) and challenges (footprints) do they present?
- **SRQ2:** How is sustainability addressed in existing frameworks and industry strategies for AI integration, and what improvements or recommendations could enhance their effectiveness?

3.2 Systematic Literature Review

The systematic literature review (SLR) followed the guidelines established by Brereton et al. [13] and the snowballing protocol proposed by Wohlin [53]. The primary objective was to identify peer-reviewed studies that addressed sustainability within Artificial Intelligence (AI)-integrated Software Development Life Cycle (SDLC) frameworks.

The searches were conducted between **June and July 2025** across four major digital libraries: *IEEE Xplore*[8], *ACM Digital Library*[9], *Scopus*[10], and *ScienceDirect*[11]. The unified search string used across all databases was as follows:

```
("Artificial Intelligence" OR "AI") AND ("Software Development" OR "SDLC") AND ("Sustainability")
```

In total, **278 papers** were retrieved (IEEE, 73; ACM, 71; Scopus, 49; ScienceDirect, 85). Duplicate, non-English, and inaccessible papers were removed before the screening process. The screening process was independently performed by **Researchers 1 and 5** following the inclusion and exclusion criteria defined in Table 1. The review included studies from 2015 onward, reflecting the adoption of modern machine learning techniques in software engineering, with sustainability relevance as an inclusion criterion to align with the study's goals.

Table 1: Inclusion and Exclusion Criteria

Category	Criteria
Inclusion	Peer-reviewed journals or conference papers (2020–2025); addressed AI use in any SDLC phase. discusses sustainability (environmental, economic, social, technical, or individual). report the empirical or conceptual results.
Exclusion	Non–peer-reviewed; Not focused on AI or SDLC. No information on AI benefits, opportunities, or any use cases related to SDLC

After screening, **32 studies** met the inclusion criteria. Subsequently, one round of *backward snowballing* was conducted, which identified an additional **32 relevant studies**, resulting in a total of **64 primary studies** included in this review. Figure 1 illustrates the overall review process encompassing the database search, screening, inclusion, and data analysis phases. Data extraction and thematic synthesis were conducted collaboratively by **Researchers 1, 2, and 3** to ensure reliability and consistency throughout the analysis.

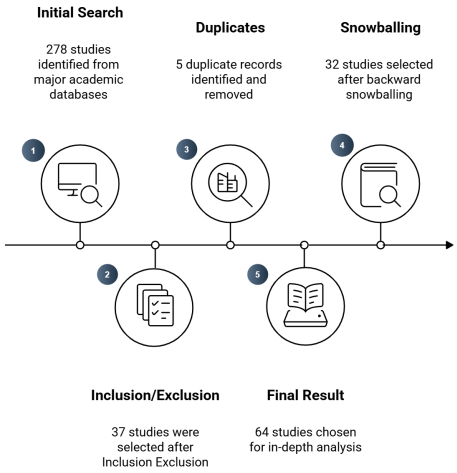


Figure 1: Literature Review Process

3.3 Practitioner Interviews

Semi-structured interviews were conducted with ten questions across three main themes: (1) AI use cases in practice, (2) impacts of AI, and (3) recommendations for sustainable integration. Data were collected between **August and September 2025** through virtual Microsoft Team sessions. The consent of participants was obtained regarding recording, information use, and their identities, and organization details were anonymized to ensure confidentiality. Participants were selected through a combination of purposive and snowball sampling, to ensure representation across industries and roles. Each interview involved two researchers: one moderating and one taking notes. A total of 27 practitioners from 12 countries participated, with Finland representing the largest share at (37%). Table 2 summarizes the distribution of organizations by country and industry domain. Participants included principal architects and software engineers with 1 to 20 years of experience. All participants were involved in at least one SDLC phase, and all organizations applied AI in at least one phase. One organization reported using an internally developed large language model for software development tasks.

All interview sessions were recorded, transcribed, and thematically analyzed by two researchers following Braun and Clarke's six-phase framework [12]. An inductive coding method was used in an Excel sheet to uncover patterns pertaining to AI adoption and sustainability within the software development lifecycle (SDLC). The coding methodology was repetitive and benefited from conversations, producing aligned themes concerning the roles of AI, its merits, obstacles, and implications for sustainability. These findings were compared with a literature review to highlight the similarities and differences between academic and industrial viewpoints on the sustainable integration of AI in the SDLC. This study provides a

[8]https://ieeexplore.ieee.org/
[9]https://dl.acm.org/
[10]https://www.scopus.com/
[11]https://www.sciencedirect.com/
[12]https://www.vappingo.com/word-blog/braun-clarkes-six-phase-framework/

Table 2: Distribution of Participating Organizations

R.ID	Country	Domain	R.ID	Country	Domain
R1	UK	Financial Services	R15	Japan	Software Development
R2	Finland	Cloud Services	R16	Finland	Research and Development
R3	Singapore	Banking	R17	Finland	Research and Development
R4	Finland	Industrial	R18	Canada	Gaming
R5	Bangladesh	Software Development	R19	Dubai	Software Development
R6	Sweden	Telecommunications	R20	Sweden	Software Development
R7	Finland	Aviation	R21	Sri Lanka	Software Development
R8	Finland	Research and Development	R22	Estonia	Software Development
R9	USA	IT Services & Consulting	R23	Sweden	Research and Development
R10	Singapore	Software Development	R24	Finland	Software Development
R11	Finland	Software Development	R25	New Zealand	Banking
R12	Finland	Energy	R26	Finland	Software Development
R13	Pakistan	Software Development	R27	Canada	Ecommerce
R14	Australia	Software Development			

Note: R.ID = Respondent ID (interview participant).

replication package [13] for transparency, including the SLR protocol, search strings, criteria for inclusion/exclusion, data extraction tools, and coding schema for interview analysis, along with an Excel coding sheet and privacy-preserved interview results.

4 Results

The results combine findings from a systematic literature review and practitioner interviews to provide an integrated understanding of sustainability in AI-integrated SDLCs. The comparison highlights how academic research (state-of-the-art) and industrial practice (state-of-the-art) align or diverge across the SDLC phases and sustainability dimensions.

4.1 AI Use Cases Across Software Development Phases (SRQ1)

The findings combine evidence from the systematic literature review (SLR) and practitioner interviews. Together, they reveal how Artificial Intelligence (AI) is being applied across different phases of the Software Development Lifecycle (SDLC). Both sources show a growing integration of AI, but with varying levels of maturity and focus across phases. This section establishes a baseline for the adoption of AI across the SDLC phases. This baseline was later used to analyze sustainability handprints and footprints. The analysis identified that most AI adoption concentrates on implementation and testing, while the requirements and deployment phases remain less explored in both research and practice [22, 46, 48].

4.1.1 State of the Art. Table 3 shows AI use cases reported in the literature across SDLC phases. Most studies propose AI techniques for implementation and testing tasks, such as code generation, defect prediction, and automated test generation [31, 46]. Design-related research focuses on model-driven approaches and pattern identification, whereas requirements engineering and deployment receive comparatively limited attention [22, 48]. Implementation and testing dominate in terms of the depth and maturity of AI integration rather than the number of publications per phase. Across the reviewed studies, AI adoption is primarily motivated by efficiency, automation, and software quality improvement, with sustainability considerations often implicit rather than explicitly addressed [48, 55]. Therefore, the table serves as a reference point for examining sustainability impacts in the subsequent sections.

4.1.2 State of the Practice. Table 4 summarizes the interviews conducted with practitioners. All 27 participants applied AI in at least one phase of the SDLC, primarily during implementation and testing. The main applications include code generation, refactoring, debugging, and automating tests. Some teams also utilize AI for clarifying requirements and prototyping designs through integrated development environment (IDE) assistants and conversational tools.

The emphasis is on speed, quality, and cost rather than sustainability indicators. Participants observed indirect benefits such as reduced rework, quicker testing, and improved resource utilization. A few large companies operate internal LLMs for domain-specific coding and documentation. Challenges continue to exist, including issues with accuracy, security, and integration into existing pipelines. The impacts on sustainability are largely unquantified, in contrast to academic research that assesses lifecycle effects[39, 46]. This discrepancy drives the analysis presented in Sections 4.2 and 4.3.

4.2 Handprints (positive impacts) and Footprints (negative impacts) (SRQ1)

Figure 2 illustrates the normalized distribution of AI-related sustainability handprints across different phases of the Software Development Life Cycle (SDLC). The handprints are most prominent during the implementation and testing phases, reflecting the widespread use of AI for tasks such as automation, productivity enhancements, and improvements in code quality and maintainability [46, 48]. Technical handprints (E5) peak during implementation and testing, which aligns with findings that highlight AI's role in enhancing readability and providing efficient automation in programming

[13] https://github.com/fazla/SLR-Interviews-AI-SDLC-Sustainability.git

Table 3: AI-Related Use Cases Across SDLC Phases Reported in Literature (State of the Art)

SDLC Phase (ID)	AI Use Cases Reported in Literature
S1	AI-based project management and effort estimation [22, 54, 55], requirement extraction and analysis from stakeholder documents [56], and improved requirement clarity using large language models [26].
S2	UML and architecture generation [22], model-driven design automation [14], pattern identification and optimization [48, 55], and GUI generation for user interface design [26].
S3	Code generation and completion [31, 43], automated code summarization and review, refactoring and defect prediction [31], and automated library migration by using deep learning [16].
S4	Automated test case generation and prioritization [46], assertion generation, bug localization and program repair [6], and anomaly detection via model-based testing [42].
S5	AI-driven DevOps and Continuous Integration/Continuous Delivery (CI/CD) pipeline optimization [22, 48], incident response and anomaly detection in cloud services [1, 19], defect and maintenance effort prediction [2, 55], technical debt reduction [4], and automated program repairs [25].

Legend: S1 Requirements, S2 Design, S3 Implementation, S4 Testing, S5 Deployment/Maintenance.

Table 4: AI Use Cases Across SDLC Phases (State of the Practice)

SDLC Phase (ID)	Observed AI Use Cases (R1–R27)
S1	• Brainstorming and idea generation (R17, R23) • Requirement extraction from documents and meetings (R1, R10) • Effort estimation and project planning (R1) • Identifying edge cases (R7)
S2	• UML and architecture generation (R22, R26, R5, R23, R14, R12) • Feature ideation and architecture design (R10, R11) • Process optimization and automation (R7)
S3	• Code generation, completion, and summarization (R3, R10, R22, R6, R17) • Automated review, triage, and refactoring (R3, R12, R22) • Library migration and optimization (R23)
S4	• Test generation, prioritization, and execution (R1–R27) • Assertions and repair automation (R14) • Test data creation (R7)
S5	• CI/CD pipeline automation and configuration (R6, R25, R27) • Root cause analysis (R7) • Anomaly detection and operational monitoring (Multiple)

Legend: S1 Requirements, S2 Design, S3 Implementation, S4 Testing, S5 Deployment/Maintenance. R = Interview Respondent ID (R1–R27).

and testing tasks [45, 56]. Economic handprints exhibit a similar pattern, supporting literature that emphasizes faster delivery and cost reductions achieved through AI-assisted development [14]. In contrast, the requirements and design phases show a limited presence of handprints. Individual handprints (E4), such as reduced developer stress and skill support, are noted only sporadically in the early phases, despite studies suggesting that AI has the potential to lower entry barriers and promote inclusive learning [19, 49, 55]. The deployment and maintenance phases exhibit moderate environmental and economic handprints related to operational efficiency [7, 14, 47]; however, technical handprints remain scarce, indicating a lack of focus on long-term maintainability after deployment. A detailed mapping of studies to the handprint categories and SDLC phases is available in the replication package.

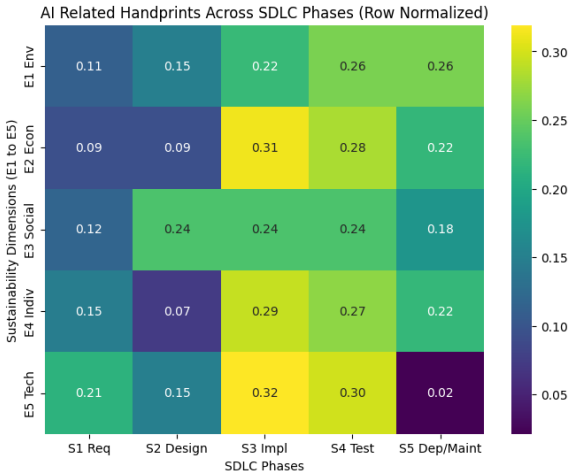


Figure 2: . Row-normalized distribution of AI related sustainability handprints across SDLC phases (State of the Art)

One senior developer noted, *"Energy consumption, I don't think that is relevant to us"* (R6). Energy and carbon impacts were largely unknown to participants.One senior developer stated. *"Who is developing this AI tool,they should consider when they develop these AI tools on energy consumption"* (R3) implying the energy consumption responsibility is with the relevant tool created party. In the Software Development Life Cycle (SDLC), the technical sustainability footprints (E5) are primarily linked to issues such as security vulnerabilities, AI hallucinations, technical debt, and an increasing reliance on unstable AI tools. During the requirements (req) and design (design) phases, research indicates early risks associated with unclear assumptions, insecure specifications, and an overreliance on AI-generated artifacts [14, 46]. These risks exacerbate during the implementation (impl) phase, where technical debt and hidden defects arising from AI-assisted code generation have been

reported [1]. The testing (test) phase is the most frequently discussed, with numerous reports detailing issues such as hallucinated test cases, unreliable oracles, and security blind spots introduced by AI-based testing tools [50]. In the deployment and maintenance (maint) phases, technical footprints persist due to model drift, tool dependency, and long-term maintainability challenges, especially when AI components are treated as black boxes [2, 46]. A complete mapping of studies to technical footprint categories and SDLC phases can be found in the replication package

4.3 Framework/strategies from academia and practice(SRQ2)

Literature offers multiple AI-integrated SDLC frameworks which highlighting environmental efficiency (Trinh et al., 2024; Akgül et al., 2025) as shown in Table 5, while practitioners focus on governance and policy-based control (e.g., internal LLMs, AI guardrails) as shown in Table 6.

Table 5: Frameworks Reported in the Literature

Frameworks
GREEN-CODE, RL-based framework for energy-efficient code generation [27].
Lifecycle-oriented framework covering requirements, design, coding, testing, and maintenance [46].
Template-based reporting with AI ID-cards [15].
FL-based governance framework for open-source AI models [31].
McLuhan's Tetrad for socio-technical impact analysis [1].
Integrate sustainability considerations from the requirements phase [48].

Table 6: Strategies Reported by Practitioners

Strategies
Apply Human-in-the-Loop practices and conduct peer reviews of AI outputs (R9, R15, R25, R23).
Adopt AI gradually, beginning with repetitive tasks and verifying generated outputs (R27, R1).
Use AI for design support instead of direct code generation to minimize risks (R19).
Leverage multiple AI tools and switch providers when needed (R24, R18, R15, R8).
Use AI with clear intent and rationale to maintain oversight and accountability (R14).
Establish company-level policies and guardrails for responsible AI use (R26, R9, R1).
Develop internal LLMs to ensure data security and control (R26).
Conduct regular training and awareness programs for developers (R2, R6, R7).
Adopt privacy-compliant AI tools and APIs for development workflows (R17, R24).
Perform controlled experiments and pilot testing before full-scale deployment (R26).
Create internal roadmaps for efficient and sustainable LLM adoption (R12, R20, R26).

R.ID denotes respondent identifiers See Table 2.

Practitioners highlighted increased productivity as the main benefit: *"If I wanna release a feature in one week with the agent support, I could do it in maybe probably 3 days or 3 1/2 days"* (R21). Yet, most noted the lack of organizational processes for validation: *"We should have a clear policies, a well defined written documentation about AI or how to utilize AI"* (R27).

5 Discussion

5.1 Bridging the State of the Art and the State of the Practice

The analysis shows alignment between research and practice regarding the application of AI, although there are differences in maturity and intent. Academic work predominantly focuses on implementation (S3) and testing (S4), highlighting areas such as automation, optimization, and quality improvement through methods like automated code generation and test prioritization [19, 46, 48]. These approaches are primarily assessed in controlled environments rather than in real-world production settings. Practitioners also report significant use of AI in S3 and S4, mainly for tasks such as code generation, debugging, refactoring, and test automation. However, the adoption of AI in earlier phases, such as requirements (S1) and design (S2), remains limited and reliant on tools. Additionally, the use of AI in deployment and maintenance (S5) is hindered by concerns related to security, reliability, and compliance. While research positions AI as a means to enable long-term optimization and sustainability [48, 55], practitioners tend to use it primarily to enhance delivery speed, reduce costs, and decrease manual effort. This reinforces the notion that sustainability tends to be a secondary outcome of an efficiency-driven approach to adoption.

5.2 Sustainability Footprints Across the SDLC

The adoption of AI in the Software Development Life Cycle (SDLC) brings both sustainability risks and efficiency gains. Research indicates that this adoption results in high energy consumption, increased computing demands, and lifecycle inefficiencies, especially with large models [14, 27, 55]. While automation enhances implementation and testing processes, these advantages are often counterbalanced by significant infrastructure and energy costs [20, 48]. Despite calls for measuring environmental impacts, there is a lack of standardized sustainability metrics. In practice, sustainability is rarely quantified. Practitioners often report indirect benefits, such as faster testing and reduced rework; however, decisions are primarily influenced by performance and delivery speed rather than sustainability objectives. This trend perpetuates a disconnect between awareness of sustainability issues and actual action taken [39, 46]. Research predominantly focuses on improving technical efficiency [27, 43], while social and individual impacts—such as skill erosion and cognitive strain—receive insufficient attention in both research and practical applications [1, 17, 19].

5.3 Suggestions for Reducing AI Impacts

The adoption of AI in the Software Development Life Cycle (SDLC) is increasing, but sustainability considerations are often neglected. While the benefits of efficiency and automation are widely acknowledged [46, 48], the negative impacts, such as higher energy consumption, growing technical debt, and skill degradation, are seldom addressed in practice (R14, R21, R25). To shift from incidental to intentional sustainability, we propose four targeted recommendations.

(1) **Align Model Size with Task Complexity**
Teams should adopt model tiering strategies that match model size to task complexity, as smaller models can achieve comparable performance for routine tasks with lower energy cost [14, 26].

(2) **Treat Technical Debt and AI Hallucination as Sustainability Risks**
AI hallucinations and generated defects increase rework and resource use, requiring AI specific quality checks and traceability to prevent hidden technical debt.

(3) **Address Skill Degradation in AI Assisted Development** To mitigate skill erosion from overreliance on AI tools, organizations should promote responsible use through targeted training and clear usage guidelines [10, 39].

5.4 Broader Implications

Researchers should shift their focus from isolated model evaluations to comprehensive lifecycle sustainability assessments. This can be achieved using tools validated in continuous integration environments rather than relying on external dashboards [22, 48]. In industry, sustainability checks should be integrated into code reviews and continuous integration (CI) pipelines. This integration should be supported by straightforward metrics that assess AI efficiency and data usage, addressing existing measurement gaps. For policymakers, establishing clearer reporting requirements, such as those outlined in the Corporate Sustainability Reporting Directive (CSRD), can enhance transparency and accountability regarding AI-related emissions. Educators should incorporate sustainability into software engineering curricula, emphasizing ethics, measurement, and the real-world energy and cost implications of AI systems.

5.5 Threats to Validity

This study has several limitations. First, the interview sample is small and geographically concentrated, which may limit the generalizability of the findings across different regions and industries. Additionally, interpretations of AI sustainability can vary depending on organizational and technical contexts. To address this ambiguity, we provided a common definition of sustainability and analyzed the responses in relation to the participants' roles. Moreover, the interviews relied on self-reported practices, which introduces the potential for recall and desirability bias. This was mitigated by involving two researchers in the process and employing cross-checked coding. The literature review encompasses studies up to mid-2025 and uses a single search string with one round of backward snowballing, which may result in the omission of some relevant work. To strengthen the validity of the findings, we applied triangulation between the literature and practitioner data.

6 Conclusion and Future Work

This study examined the adoption of artificial intelligence (AI) in the Software Development Life Cycle (SDLC) through a systematic literature review and practitioner interviews. Findings reveal that AI usage is mainly focused on the implementation and testing phases, aiming for automation, speed, and quality, rather than sustainability. The requirements and deployment phases are less explored, with sustainability often seen as a secondary consideration. Future research should focus on operationalizing sustainability metrics for AI within SDLC and CI/CD pipelines, measuring energy consumption with automated energy estimation, cost monitoring dashboards, quality, and bias across all SDLC phases, particularly in real-world Continuous Integration (CI) environments. Field studies should assess sustainability impacts using operational data like CI logs. Moreover, it's essential to validate lightweight sustainability indicators for Integrated Development Environments (IDEs) and examine AI-specific quality gates. Finally, more emphasis should be placed on human and organizational factors, including skill evolution, trust in AI, and governance practices.

Acknowledgments

This work has been supported by FAST, the Finnish Software Engineering Doctoral Research Network, funded by the Ministry of Education and Culture, Finland.

References

[1] Silvia Abrahão, John Grundy, Mauro Pezzè, Margaret-Anne Storey, and Damian A Tamburri. 2025. Software Engineering by and for Humans in an AI Era. *ACM Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–46.

[2] Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, Thomas Zimmermann, Xuchao Zhang, and Saravan Rajmohan. 2023. Recommending root-cause and mitigation steps for cloud incidents using large language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1737–1749.

[3] Nilay Yorgancılar Akgül, Tuğba Taşkaya Temizel, Özden Özcan Top, and Pelin Dayan Akman. 2025. Aligning Data Debt with AI-Integrated Software Project Lifecycle Processes: A Standard-Based Mapping Approach. In *Proceedings - 2025 IEEE/ACM International Conference on Technical Debt, TechDebt 2025*. Institute of Electrical and Electronics Engineers Inc., 1–11.

[4] Hadeel Alsolai and Marc Roper. 2020. A systematic literature review of machine learning techniques for software maintainability prediction. *Information and Software Technology* 119 (2020), 106214.

[5] R.A.F. Alvarenga, S. Huysveld, S.E. Taelman, S. Sfez, N. Préat, M. Cooreman-Algoed, D. Sanjuan-Delmás, and J. Dewulf. 2020. A framework for using the handprint concept in attributional life cycle (sustainability) assessment. *Journal of Cleaner Production* 265 (Aug. 2020), 121743. doi:10.1016/j.jclepro.2020.121743

[6] Domenico Amalfitano, Stefano Faralli, Jean Carlo Rossa Hauck, Santiago Matalonga, and Damiano Distante. 2023. Artificial intelligence applied to software testing: A tertiary study. *Comput. Surveys* 56, 3 (2023), 1–38.

[7] Ahmed El Cheikh Ammar, Sukru Eraslan, and Yeliz Yesilada. 2025. Predicting the truck factor in a software repository using machine learning. *Information and Software Technology* (2025), 107765.

[8] Yatin Bajaj and Manoj Kumar Samal. 2023. Accelerating software quality: Unleashing the power of generative ai for automated test-case generation and bug identification. *International Journal for Research in Applied Science and Engineering Technology* 11, 7 (2023), 345–350.

[9] S. Balaji and M.S. Murugaiyan. 2012. Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management* 2, 1 (2012), 26–30.

[10] Leonardo Banh, Florian Holldack, and Gero Strobel. 2025. Copiloting the future: How generative AI transforms Software Engineering. *Information and Software Technology* 183 (2025), 107751.

[11] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, and C.C. Venters. 2015. Sustainability Design and Software: The Karlskrona Manifesto. In *Proceedings of the 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* (2015-05-16/2015-05-24). Florence, Italy.

[12] Stefanie Betz, Birgit Penzenstadler, Leticia Duboc, Ruzanna Chitchyan, Sedef Akinli Kocak, Ian Brooks, Shola Oyedeji, Jari Porras, Norbert Seyff, and Colin C. Venters. 2024. Lessons Learned from Developing a Sustainability Awareness Framework for Software Engineering Using Design Science. *ACM Transactions on Software Engineering and Methodology* 33, 5 (june 2024), 1–39. doi:10.1145/3649597

[13] Pearl Brereton, Barbara A Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. 2007. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of systems and software* 80, 4 (2007), 571–583.

[14] Kouider Chadli, Goetz Botterweck, and Takfarinas Saber. 2024. The environmental cost of engineering machine learning-enabled systems: A mapping study. In *Proceedings of the 4th Workshop on Machine Learning and Systems*. 200–207.

[15] Luís Cruz, Xavier Franch, and Silverio Martínez-Fernández. 2025. Innovating for Tomorrow: The Convergence of Software Engineering and Green AI. *ACM Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–13.

[16] Juri Di Rocco, Phuong T Nguyen, Claudio Di Sipio, Riccardo Rubei, Davide Di Ruscio, and Massimiliano Di Penta. 2025. DeepMig: A transformer-based approach to support coupled library and code migrations. *Information and Software Technology* 177 (2025), 107588.

[17] Usman Khan Durrani, Mustafa Akpinar, Muhammed Fatih Adak, Abdullah Talha Kabakus, Muhammed Maruf Ozturk, and Mohammed Saleh. 2024. A decade of progress: A systematic literature review on the integration of AI in software engineering phases and activities (2013-2023). *IEEE Access* (2024).

[18] European Commission. 2020. *Supporting the Green Transition.* Technical Report. European Commission. https://ec.europa.eu/commission/presscorner/detail/en/fs_20_281 Technical report.

[19] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M Zhang. 2023. Large language models for software engineering: Survey and open problems. In *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. IEEE, 31–53.

[20] Alessio Ferrari and Paola Spoletini. 2025. Formal requirements engineering and large language models: A two-way roadmap. *Information and Software Technology* 181 (2025), 107697.

[21] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon S Blair, and Adrian Friday. 2021. The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations. *Patterns* 2, 9 (2021).

[22] Cuiyun Gao, Xing Hu, Shan Gao, Xin Xia, and Zhi Jin. 2025. The current challenges of software engineering in the era of large language models. *ACM Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–30.

[23] Alexandra González, Xavier Franch, David Lo, and Silverio Martínez-Fernández. 2025. How do Pre-Trained Models Support Software Engineering? An Empirical Study in Hugging Face. *arXiv preprint arXiv:2506.03013* (2025).

[24] Joseph H.A. Guillaume, Suvi Sojamo, Miina Porkka, Dieter Gerten, Mika Jalava, Leena Lankoski, Elina Lehikoinen, Michael Lettenmeier, Stephan Pfister, Kirsi Usva, Yoshihide Wada, and Matti Kummu. 2020. Giving Legs to Handprint Thinking: Foundations for Evaluating the Good We Do. *Earth's Future* 8, 6 (june 2020), e2019EF001422. doi:10.1029/2019EF001422

[25] Achim Guldner, Rabea Bender, Coral Calero, Giovanni S Fernando, Markus Funke, Jens Gröger, Lorenz M Hilty, Julian Hörnschemeyer, Geerd-Dietger Hoffmann, Dennis Junger, et al. 2024. 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* 155 (2024), 402–418.

[26] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology* 33, 8 (2024), 1–79.

[27] Shashikant Ilager, Lukas Florian Briem, and Ivona Brandic. 2025. Green-Code: Learning to Optimize Energy Efficiency in Llm-Based Code Generation. In *2025 IEEE 25th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 559–569.

[28] Asif Imran and Tevfik Kosar. 2019. Software Sustainability: A Systematic Literature Review and Comprehensive Analysis. *Information and Software Technology* (2019). Preprint available on arXiv:1910.06109.

[29] Michael Kühnen, Samanthi Silva, Janpeter Beckmann, Ulrike Eberle, Rüdiger Hahn, Christoph Hermann, Stefan Schaltegger, and Marianne Schmid. 2019. Contributions to the sustainable development goals in life cycle sustainability assessment: Insights from the Handprint research project. *NachhaltigkeitsManagementForum | Sustainability Management Forum* 27, 1 (March 2019), 65–82. doi:10.1007/s00550-019-00484-y

[30] Giuseppe Lami, Fabrizio Fabbrini, and Mario Fusani. 2013. A methodology to derive sustainability indicators for software development projects. In *Proceedings of the 2013 International Conference on Software and System Process*. ACM, San Francisco CA USA, 70–77. doi:10.1145/2486046.2486060

[31] Yihao Li, Pan Liu, Haiyang Wang, Jie Chu, and W Eric Wong. 2025. Evaluating large language models for software testing. *Computer Standards & Interfaces* 93 (2025), 103942.

[32] Antonio Mastropaolo, Camilo Escobar-Velásquez, and Mario Linares-Vásquez. 2025. From triumph to uncertainty: The journey of software engineering in the AI era. *ACM Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–34.

[33] M Mohankumar and M Anand Kumar. 2015. An Empirical Study on Green and Sustainable Software Engineering. In *Proceedings of the 11th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*. 95–104.

[34] Ichchha Moktan, Muhammad Asif Khan, Shola Oyedeji, Mikko Puonti, Mikhail Ola Adisa, and Jari Porras. 2025. Practical adoption of green coding: a comparative study across organizations in Finland and Globally. In *International Conference on Software Business*. Springer, 361–378.

[35] Brunna C Mourão, Leila Karita, and Ivan do Carmo Machado. 2018. Green and Sustainable Software Engineering - a Systematic Mapping Study. In *Proceedings of the 17th Brazilian Symposium on Software Quality (SBQS)*. 153–162.

[36] A. Nguyen-Duc et al. 2023. Generative Artificial Intelligence for Software Engineering – A Research Agenda. arXiv preprint arXiv:2304.07590.

[37] Shola Oyedeji, Muhammad Asif Khan, Panu Puhtila, Oshani Weerakoon, Tuomas Mäkilä, Mikhail Ola Adisa, Bilal Naqvi, and Santeri Auvinen. 2025. Green coding: state of practice. In *2025 11th International Conference on ICT for Sustainability (ICT4S)*. IEEE, 91–99.

[38] Shola Oyedeji, Ahmed Seffah, and Birgit Penzenstadler. 2018. A Catalogue Supporting Software Sustainability Design. *Sustainability* 10, 7 (2018), 2296.

[39] WPPM Pathirana and Yehemini Jayatissa. 2025. Towards Sustainable Software Testing Practices in IT Firms. In *2025 5th International Conference on Advanced Research in Computing (ICARC)*. IEEE, 1–6.

[40] Daniel Russo, Sebastian Baltes, Niels van Berkel, Paris Avgeriou, Fabio Calefato, Beatriz Cabrero-Daniel, Gemma Catolino, Jürgen Cito, Neil Ernst, Thomas Fritz, Hideaki Hata, Reid Holmes, Maliheh Izadi, Foutse Khomh, Mikkel Baun Kjærgaard, Grischa Liebel, Alberto Lluch Lafuente, Stefano Lambiase, Walid Maalej, Gail Murphy, Nils Brede Moe, Gabrielle O'Brien, Elda Paja, Mauro Pezzè, John Stouby Persson, Rafael Prikladnicki, Paul Ralph, Martin Robillard, Thiago Rocha Silva, Klaas Jan Stol, Margaret Anne Storey, Viktoria Stray, Paolo Tell, Christoph Treude, and Bogdan Vasilescu. 2024. Generative AI in Software Engineering Must Be Human-Centered: The Copenhagen Manifesto.

[41] M.A. Sami, M. Waseem, Z. Zhang, Z. Rasheed, K. Systä, and P. Abrahamsson. 2024. AI based Multiagent Approach for Requirements Elicitation and Analysis. arXiv:2409.00038.

[42] Max Schäfer, Sarah Nadi, Aryaz Eghbali, and Frank Tip. 2023. An empirical evaluation of using large language models for automated unit test generation. *IEEE Transactions on Software Engineering* 50, 1 (2023), 85–105.

[43] Jieke Shi, Zhou Yang, and David Lo. 2025. Efficient and Green Large Language Models for Software Engineering: Literature Review, Vision, and the Road Ahead. *ACM Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–22.

[44] Thibault Simon, Pierre Rust, Romain Rouvoy, and Joël Penhoat. 2023. Uncovering the Environmental Impact of Software Life Cycle. In *2023 International Conference on ICT for Sustainability (ICT4S)*. IEEE, Rennes, France, 176–187. doi:10.1109/ICT4S58814.2023.00026

[45] Trevor Stalnaker, Nathan Wintersgill, Oscar Chaparro, Laura A Heymann, Massimiliano Di Penta, Daniel M German, and Denys Poshyvanyk. 2024. Developer Perspectives on Licensing and Copyright Issues Arising from Generative AI for Software Development. *ACM Transactions on Software Engineering and Methodology* (2024).

[46] Valerio Terragni, Annie Vella, Partha Roop, and Kelly Blincoe. 2025. The future of AI-driven software engineering. *ACM Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–20.

[47] Valerio Terragni, Annie Vella, Partha Roop, and Kelly Blincoe. 2025. The Future of AI-Driven Software Engineering. *ACM Trans. Softw. Eng. Methodol.* 34, 5, Article 120 (May 2025), 20 pages. https://doi-org.ezproxy.cc.lut.fi/10.1145/3715003

[48] Eames Trinh, Markus Funke, Patricia Lago, and Justus Bogner. 2024. Sustainability integration of artificial intelligence into the software development life cycle. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 199–206.

[49] Sanidhya Vijayvargiya, Lov Kumar, Lalita Bhanu Murthy, and Sanjay Misra. 2022. Software requirements classification using deep-learning approach with various hidden layers. In *2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE, 895–904.

[50] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. 2024. Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering* 50, 4 (2024), 911–936.

[51] Oshani Weerakoon, Shola Oyedeji, Md Abdus Samad, Abdulkadir Abubakar, Ahsan Ishan, Muhammad Shayan, Tuomas Mäkilä, and Erkki Kaila. 2025. Enhancing Agile Workflows with AI-Driven, Sustainability-Aware Requirements Engineering: A Design Science Approach. In *International Conference on Software Business*. Springer, 212–229.

[52] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D.C. Schmidt. 2023. ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. arXiv preprint arXiv:2303.07839.

[53] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. 1–10.

[54] Lin Yang, Chen Yang, Shutao Gao, Weijing Wang, Bo Wang, Qihao Zhu, Xiao Chu, Jianyi Zhou, Guangtai Liang, Qianxiang Wang, et al. 2024. On the evaluation of large language models in unit test generation. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 1607–1619.

[55] Yanming Yang, Xin Xia, David Lo, and John Grundy. 2022. A survey on deep learning for software engineering. *ACM Computing Surveys (CSUR)* 54, 10s (2022), 1–73.

[56] Jie Sh'ng Yeow, Muhammad Ehsan Rana, and Nur Amira Abdul Majid. 2024. An automated model of software requirement engineering using gpt-3.5. In *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*. IEEE, 1746–1755.

[57] Z. Zhang, M. Rayhan, T. Herda, M. Goisauf, and P. Abrahamsson. 2024. LLM-Based Agents for Automating the Enhancement of User Story Quality: An Early Report. In *Agile Processes in Software Engineering and Extreme Programming*, D. Šmite, E. Guerra, X. Wang, M. Marchesi, and P. Gregory (Eds.). Springer, Cham, 117–126.