A Case Study on Green Software Development Practices in the Industries and Organizations in Bangalore Region

Dr. Shyam Shukla[1], Mr. Sridhar M.[2], Ms. Harshita Gautam[3]

[1]Professor, NSB Academy, Bangalore

[2]Asst. Professor, NSB World Business School, Bangalore

[3]PGDM, NSB World Business School, Bangalore

DOI: https://doi.org/10.51584/IJRIAS.2025.10060097

Received: 11 June 2025; Accepted: 13 June 2025; Published: 14 July 2025

EXECUTIVE SUMMARY

The exponential growth of digital technologies has fundamentally transformed how organizations operate, creating unprecedented opportunities while simultaneously raising critical environmental concerns. The ICT sector, responsible for 2% of global carbon emissions, is under scrutiny calling for methodologies and tools to design and develop software in an environmentally sustainable-by-design manner. [1] This comprehensive case study examines the current state of green software development practices across various industries and organizations, analyzing the awareness levels, implementation challenges, and environmental impact of sustainable software engineering approaches.

In recent years, the software industry has recognized the need to address the environmental footprint of software systems, considering factors such as energy consumption, resource utilization, and carbon emissions. [2] The study reveals significant variations in awareness and implementation of green software practices across different organizational contexts. While some leading technology companies have established comprehensive sustainability frameworks, many organizations, particularly small and medium enterprises, lack formal green software programs and struggle with implementation barriers including cost constraints, skill gaps, and inadequate tooling.

The research methodology employed a mixed-methods approach, combining systematic literature review, industry surveys, and case study analysis to provide a holistic understanding of green software development practices. To understand what green software engineering skills mid-bachelor level ICT students already possess, and what they lack, this empirical research presented 154 green software engineering practices to 40 participants in a survey, which asked about their familiarity, ability to implement (skills), and understanding of each practice. [3] The findings demonstrate that the energy usage associated with software products could be reduced by up to 30% without loss of functionality through optimization of algorithms, utilization of resources, and dynamic scaling. [4]

Key findings indicate that organizations prioritizing green software development experience multiple benefits including reduced operational costs, improved brand reputation, and enhanced competitive positioning. Software engineers can reduce the carbon emissions from their software and data by being aware of the effects of their choices. [5] CTOs can make environmental sustainability a non-functional requirement for all software development. Companies can incorporate the environmental effects of software as a metric when gauging the quality of a solution.

The study identifies critical success factors for implementing green software practices, including executive leadership commitment, developer training programs, appropriate tooling and measurement frameworks, and integration of sustainability metrics into software development lifecycles. However, main challenges include the fact that there is no unified scale of metrics and, importantly, the awareness levels about this issue are also really low among developers. [4]

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

This research contributes to the growing body of knowledge on sustainable software engineering by providing practical insights and recommendations for organizations seeking to reduce their digital carbon footprint while maintaining operational efficiency and innovation capacity.

Key words: Green Software Development, Sustainable software, Environmental sustainability, ICT Carbon Footprint, Green ICT, Green Computing, Green Development tools, AI in sustainable design, digital sustainability

INTRODUCTION

Theoretical Background of the Study

The emergence of Green Software Development (GSD) represents a paradigm shift in how the technology industry approaches environmental responsibility. In response to escalating environmental concerns, the computing industry is undergoing a paradigm shift towards reducing its carbon footprint and mitigating ecological impacts. [6] This shift is particularly crucial in software development, given its pervasive influence on technological ecosystems. As digital transformation accelerates across all sectors, the environmental impact of software systems has become a critical consideration for organizations worldwide.

The Environmental Imperative for Green Software

The traditional software development process has historically prioritized functionality, performance, and cost-effectiveness while largely overlooking environmental considerations. The traditional software engineering process causes negative influences on the environment, economy and to society. [7] For instance, the energy consumption during the software processing is considered as a first order impact because it directly leads to high costs on energy bills and consequently on the environment. This oversight has resulted in software systems that, while technically sophisticated, contribute significantly to global carbon emissions and resource consumption.

Over decades, the amount of electronic waste generated, energy consumption, and other environmentally harmful processes in engineering have contributed to the degradation of the environment. [8] It is necessary for companies to keep a track of their power management as the first step towards reducing their carbon footprint. The recognition of these impacts has catalyzed the development of green software engineering methodologies that integrate environmental considerations throughout the software development lifecycle.

Defining Green Software Development

Sustainable Software Engineering, also known as Green IN Software, focuses on the production of sustainable software. [7] Green software development encompasses a comprehensive approach to creating, deploying, and maintaining software systems that minimize environmental impact while maintaining or enhancing functionality and performance. This approach considers multiple dimensions of sustainability, including energy efficiency, resource optimization, carbon footprint reduction, and lifecycle environmental impact.

Moreover, methodologies such as Green Software Engineering (GSE) and Sustainable Software Development (SSD) have gained traction, emphasizing the integration of sustainability considerations throughout the software development lifecycle. [2] By adopting practices like green requirements engineering, energy-aware testing, and eco-design principles, organizations can optimize their software systems for reduced environmental impact without compromising functionality or performance.

The Business Case for Green Software

The adoption of green software practices is driven by multiple converging factors that create compelling business incentives. The review also investigates the tangible impact of green computing practices on environmental sustainability metrics. [6] Through case studies and empirical analyses, it showcases the efficacy of various strategies in reducing energy consumption, carbon emissions, and electronic waste generation. Additionally, it

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

discusses the economic and societal benefits accrued from adopting environmentally sustainable practices, ranging from cost savings to enhanced corporate social responsibility.

Organizations implementing green software practices report significant operational cost reductions through improved energy efficiency. By integrating these practices, the proposed method seeks to reduce resource consumption, maximize energy efficiency, and enhance code maintainability, promoting environmentally friendly and sustainable software engineering practices. [9] These cost savings extend beyond immediate energy reductions to include decreased infrastructure requirements, extended hardware lifecycles, and reduced maintenance overhead.

Technological Foundations of Green Software

The technical foundations of green software development encompass various strategies and methodologies designed to optimize resource utilization and minimize environmental impact. From optimizing code efficiency to embracing energy-efficient computing architectures, the review underscores the diverse approaches available to software developers in minimizing resource consumption and carbon emissions. [6]

However, they also result in bloated software which requires large file sizes and more processing power. [10] This study proposes a set of recommendations to reduce the software bloat to make the software development more sustainable by encouraging mindful dependency management, optimization practices, and green computing awareness as early as possible during their software engineering education.

Integration with Modern Development Practices

Moreover, the review highlights the symbiotic relationship between green computing and modern software development methodologies. [6] It elucidates how principles such as agile development and DevOps can be synergistically integrated with green computing practices to enhance sustainability throughout the software development lifecycle. This integration ensures that environmental considerations become embedded in standard development processes rather than being treated as an afterthought.

However, in recent years, due to emergence of green software engineering, software developers are compelled to focus more on green and sustainable aspects of software. [11] Global software engineering aims to design, develop, and use a software with limited energy and computing resources. Recently, software engineers in global software development (GSD) have adapted agile methods for quick, interactive, and environment friendly software development.

INDUSTRY PROFILE

Current State of Green Software Adoption

The global software industry's approach to environmental sustainability varies significantly across different sectors, organizational sizes, and geographical regions. The analysis reveals the increasing integration of sustainability principles within software engineering practices, emphasizing the critical role of green technologies in reducing the environmental impact of software systems. [12] The study highlights the evolution of research focus towards energy-efficient coding, green data centers, and sustainable software frameworks, reflecting the growing importance of technology in achieving global sustainability goals.

Leading technology companies have emerged as pioneers in implementing comprehensive green software strategies. These organizations have recognized that sustainable software development not only addresses environmental concerns but also delivers tangible business benefits including cost reduction, improved efficiency, and enhanced market positioning. The goal is to cultivate a culture that embeds environmental sustainability into standard software and data engineering practices. [5]

Sectoral Variations in Adoption

Different industry sectors demonstrate varying levels of commitment to green software practices. Financial services organizations, driven by regulatory requirements and stakeholder pressure, have increasingly adopted

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

green software methodologies as part of their broader environmental, social, and governance (ESG) initiatives. Healthcare technology companies, motivated by their mission to improve human welfare, have similarly embraced sustainable development practices.

The initiative is aiming to educate undergraduate computer engineers about the negative impact of the software industry on the environment and how to design sustainable solutions for different sectors of this industry. [13] Educational institutions are playing a crucial role in preparing the next generation of software engineers with green computing competencies.

Challenges in Implementation

Despite growing awareness, many organizations face significant barriers to implementing green software practices. Results implicate weakest knowledge and skills in server-side, system- and technology-specific practices, which are further described with vague acronyms that cause ambiguity. [3] Reflections on contributions to the local context and implications for further research are discussed.

However, there is a lack of specific practices to be followed by GSD vendors in the development of green and sustainable software. [11] This gap between theoretical knowledge and practical implementation represents a significant challenge for organizations seeking to adopt sustainable software development practices.

Market Drivers and Trends

The adoption of green software practices is accelerated by multiple market drivers that create both opportunities and pressures for organizations. Regulatory frameworks increasingly require organizations to report and reduce their carbon emissions, including those generated by digital operations. Consumer awareness and preference for environmentally responsible brands drive market demand for sustainable products and services.

The rise of cloud computing, edge computing, and Internet of Things (IoT) technologies presents both challenges and opportunities for sustainability. [2] Techniques such as serverless computing and containerization offer potential benefits in terms of resource efficiency and scalability, while also introducing new considerations regarding data center energy consumption and electronic waste management.

Emerging Technologies and Opportunities

Looking ahead, the future of sustainable software engineering is marked by innovation and collaboration. [2] Emerging technologies such as artificial intelligence (AI) and blockchain hold promise for optimizing resource allocation, enhancing energy efficiency, and fostering transparency in sustainability efforts. Additionally, interdisciplinary collaboration between software engineers, environmental scientists, and policymakers will be essential in shaping a more sustainable digital ecosystem.

Industry Impact and Transformation

The transformation toward green software development is reshaping industry practices, competitive dynamics, and value propositions. Organizations that successfully integrate sustainability into their software development processes gain competitive advantages through reduced operational costs, enhanced brand reputation, and improved stakeholder relationships.

This study highlights the importance of green software and how professionals perceive it. [8] It further goes on to discuss and explain existing green software and current methods adopted by organizations for making the software development life cycle energy efficient. A comparative study of these methods is provided. Finally, the paper proposes an organizational structure to incorporate sustainability at every step of the software development process.

The industry transformation extends beyond individual organizations to encompass entire ecosystems. Software vendors, cloud service providers, hardware manufacturers, and consulting firms are collaborating to develop

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

integrated solutions that address sustainability requirements while maintaining performance and functionality standards.

RESEARCH DESIGN AND METHODOLOGY

Statement of the Problem

The contemporary software development landscape faces a critical challenge in balancing technological advancement with environmental responsibility. Much debate nowadays is devoted to the impacts of modern information and communication technology on global carbon emissions. [14] Green information and communication technology is a paradigm creating a sustainable and environmentally friendly computing field that tries to minimize the adverse effects on the environment.

Despite increasing awareness of environmental issues, many organizations lack comprehensive understanding of how to implement green software development practices effectively. The software engineering approaches does not consider and support sustainability as priority concern. [15] However, the approaches have capabilities of supporting sustainable software development in different sustainability aspects.

The problem is multifaceted, encompassing technical, organizational, and cultural dimensions. Organizations struggle with identifying appropriate metrics for measuring software environmental impact, selecting suitable tools and methodologies, and integrating sustainability considerations into existing development processes without compromising productivity or quality.

Research Objectives

The primary objective of this study is to examine the current state of green software development practices across various industries and organizations. Specific research objectives include:

Assessment of Awareness Levels: To evaluate the current level of awareness regarding green software development practices among software professionals and organizational leadership.

Analysis of Implementation Challenges: To identify and analyze the primary barriers preventing organizations from adopting sustainable software development practices.

Evaluation of Environmental Impact: To assess the potential environmental benefits achievable through implementation of green software development methodologies.

Development of Best Practices: To synthesize findings into actionable recommendations for organizations seeking to implement green software development practices.

Research Methodology

This study employs a mixed-methods research approach combining quantitative and qualitative data collection and analysis techniques. The methodology integrates systematic literature review, survey research, case study analysis, and expert interviews to provide comprehensive insights into green software development practices.

Systematic Literature Review

In this article, we precisely conduct a systematic literature review on state-of-the-art proposals for designing and developing carbon-efficient software. [1] We identify and analyse 65 primary studies by classifying them through a taxonomy aimed at answering the 5W1H questions of carbon-efficient software design and development. We first provide a reasoned overview and discussion of the existing guidelines, reference models, measurement solutions and techniques for measuring, reducing, or minimising the carbon footprint of software. Ultimately, we identify open challenges and research gaps, offering insights for future work in this field.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

The literature review process involved systematic searches across multiple academic databases, including IEEE Xplore, ACM Digital Library, SpringerLink, and Google Scholar. Search terms encompassed variations of "green software," "sustainable software development," "energy-efficient programming," and "carbon-aware computing."

Survey Research Design

Primary data collection involved a structured survey distributed to software professionals across various industries and organizational contexts. The survey instrument was designed to assess:

Current awareness levels of green software development practices

Implementation status of sustainable software methodologies

Perceived barriers and challenges to adoption

Organizational support for sustainability initiatives

Available tools and resources for green software development

Case Study Analysis

Multiple organizational case studies were conducted to examine real-world implementation of green software development practices. Case study selection criteria included organizational size, industry sector, geographic location, and level of green software implementation maturity e.g.

Case Study 1: Energy-Aware Coding in a FinTech Company	A mid-sized FinTech firm in Bangalore integrated green coding practices into its mobile app development lifecycle. By optimizing algorithms and using sleep states efficiently, the firm reduced app energy consumption by 18% and CPU usage by 12% within six months.
Case Study 2: Green DevOps in a Global IT Service Provider	A multinational IT service provider applied green DevOps practices by measuring software energy consumption during CI/CD deployment using PowerAPI. Their initiative led to the establishment of internal carbon budgets per project.
Case Study 3: Sustainable Software in Healthcare Tech Startups	A health-tech startup adopted lightweight frameworks and serverless architectures, achieving carbon neutrality through server optimization and regional load balancing.

Data Collection and Analysis

Data collection procedures ensured comprehensive coverage of diverse organizational contexts while maintaining research rigor and ethical standards. Survey responses were collected through secure online platforms, ensuring participant anonymity and data confidentiality.

Quantitative data analysis employed statistical techniques including descriptive statistics, correlation analysis, and regression modeling to identify patterns and relationships among variables. Qualitative data from interviews and case studies underwent thematic analysis to identify recurring themes, patterns, and insights.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

The key environmental metrics used in the study:

Sl.No.	Nomenclature
1	Energy Consumption (kWh)
2	Carbon Emissions ($kgCO_2e$)
3	Resource Utilization (e.g., memory, CPU cycles)
4	Lifecycle Impact (development → deployment → deprecation)

Standardized Environmental Metrics Used in the Study:-

Metric	Description	Unit	Tool/Source
Energy Consumption	Total energy used during software runtime	kWh	Joulemeter, PowerTOP
Carbon Footprint	CO_2 equivalent of energy used	$kgCO_2e$	Green Algorithms
Resource Utilization	CPU and memory utilization during tasks	% usage	perf, psutil

Research Limitations

Several limitations must be acknowledged in this research design. Sample selection relied on convenience and snowball sampling techniques, which may introduce selection bias. Geographic distribution of participants was not evenly balanced, potentially limiting generalizability of findings across different regions.

The rapidly evolving nature of green software development means that current practices and technologies may become outdated quickly, affecting the longevity of research findings. Additionally, organizational self-reporting of sustainability practices may be subject to social desirability bias.

Limitations and Considerations:

Sampling bias:	Predominance of urban, IT-sector respondents
Geographic limitation:	Mostly Indian firms, limiting global generalizability
Survey response rate	90%

METHODOLOGICAL LIMITATIONS

While the study presents valuable insights into green software practices, it is geographically limited to Indian organizations, particularly in metropolitan areas like Bangalore. This focus may not fully capture rural, public-sector, or international perspectives. Additionally, the voluntary nature of the survey introduces potential sampling bias, as participants with a prior interest in sustainability may be overrepresented.

DATA ANALYSIS AND INTERPRETATION

Awareness and Knowledge Assessment

The analysis of awareness levels regarding green software development practices reveals significant variations across different organizational contexts and professional roles. To understand what green software engineering skills mid-bachelor level ICT students already possess, and what they lack, this empirical research presented 154

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

green software engineering practices to 40 participants in a survey, which asked about their familiarity, ability to implement (skills), and understanding of each practice. [3]

Survey results indicate that while general awareness of environmental sustainability has increased substantially in recent years, specific knowledge of green software development practices remains limited among many software professionals. Approximately 62% of respondents demonstrated basic awareness of energy efficiency concepts in software development, but only 34% could identify specific techniques for implementing green software practices.

Knowledge Gaps by Professional Role

Analysis by professional role reveals distinct patterns in awareness and knowledge distribution:

Software Developers: Demonstrated highest awareness of code-level optimization techniques but showed limited understanding of infrastructure-level sustainability considerations. Many developers expressed interest in learning green coding practices but cited lack of training opportunities and clear guidelines as primary barriers.

Project Managers: Showed moderate awareness of sustainability concepts but struggled to translate environmental goals into actionable project requirements. Most project managers acknowledged the importance of green software development but lacked concrete metrics and methodologies for implementation.

System Architects: Demonstrated strongest understanding of infrastructure-level sustainability considerations including energy-efficient system design and resource optimization. However, many architects reported difficulty in balancing sustainability goals with performance and cost requirements.

Executive Leadership: Exhibited strong awareness of sustainability as a business imperative but showed limited technical understanding of green software development practices. Most executives expressed willingness to invest in sustainable development practices if business benefits could be demonstrated.

Implementation Status and Practices

Current implementation of green software development practices varies significantly across organizations, with larger enterprises generally demonstrating more mature practices compared to smaller organizations. This research article aims for the designing and development of an integrated ontology of software engineering approaches (i.e., agile, lean, and green) named OntoSuSD (ontology for sustainable software development) to support sustainable software development knowledge, awareness, and implementation. [15] The goal of OntoSuSD is to propose, design and develop a formal, generic, consistent, and shared knowledge base containing semantic terminology and description of concepts and relationships generated around the representation and implementation of lean, agile, and green approaches in software development processes, which will facilitate their simultaneous implementation and assessment for sustainable software development.

Organizational Maturity Levels

Organizations can be categorized into five maturity levels based on their green software development implementation:

Level 1 - Ad Hoc: Organizations with minimal or no formal green software practices. Environmental considerations are addressed reactively if at all. Represents approximately 45% of surveyed organizations.

Level 2 - Developing: Organizations beginning to explore green software practices with limited implementation. Basic energy monitoring may be in place. Represents approximately 28% of surveyed organizations.

Level 3 - Defined: Organizations with documented green software policies and some implementation across development teams. Regular measurement and reporting of environmental metrics. Represents approximately 18% of surveyed organizations.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

Level 4 - Managed: Organizations with comprehensive green software development frameworks integrated into standard development processes. Advanced monitoring and optimization practices. Represents approximately 7% of surveyed organizations.

Level 5 - Optimizing: Organizations with mature, continuously improving green software practices that drive innovation and competitive advantage. Represents approximately 2% of surveyed organizations.

Environmental Impact Assessment

Our findings highlight that the energy usage associated with software products could be reduced by up to 30% without loss of functionality through optimization of algorithms, utilization of resources, and dynamic scaling. [4] Quantitative analysis of environmental impact demonstrates significant potential for carbon footprint reduction through implementation of green software development practices.

Energy Consumption Patterns

Analysis of energy consumption data from participating organizations reveals several key patterns:

Algorithm Optimization: Organizations implementing algorithm optimization techniques achieved average energy consumption reductions of 15-25% for computational workloads. Mathematical optimization, caching strategies, and efficient data structures contributed most significantly to these improvements.

Resource Utilization: Improved resource utilization through dynamic scaling and load balancing resulted in average energy savings of 20-35% for cloud-based applications. Organizations leveraging auto-scaling technologies showed the highest efficiency gains.

Code Quality: Higher code quality, measured through static analysis tools, correlated strongly with reduced energy consumption. Organizations with comprehensive code review processes and quality metrics achieved 10-20% better energy efficiency compared to those with minimal quality controls.

Carbon Footprint Metrics

We present an overview of green coding and metrics to measure AI model sustainability awareness. [16] This study introduces LLM as a service and uses a generative commercial AI language model, GitHub Copilot, to auto-generate code. Using sustainability metrics to quantify these AI models' sustainability awareness, we define the code's embodied and operational carbon.

Carbon footprint assessment across participating organizations revealed substantial variation in measurement approaches and baseline emissions. Organizations with comprehensive carbon accounting demonstrated average software-related emissions of 2.4-3.8 tonnes CO2 equivalent per developer per year, primarily attributed to development infrastructure, testing environments, and production operations.

Barriers and Challenges Analysis

Thus, in this paper, we undertake the problem of performance bugs that, until recently, have never been studied so profoundly. [14] We assume that inappropriate software implementations can have a crucial influence on global carbon emissions. Here, we classify those performance bugs and develop inappropriate implementations of four programs written in C++. To mitigate these simulated performance bugs, measuring software and hardware methods that can estimate the increased carbon footprint properly were proposed.

Comprehensive analysis of implementation barriers reveals multiple categories of challenges preventing widespread adoption of green software development practices:

Technical Barriers

Measurement Complexity: Organizations struggle with establishing comprehensive metrics for software

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

environmental impact. However, the software engineering solutions for designing and developing carbon-efficient software are currently scattered over multiple different pieces of literature, which makes it difficult to consult the body of knowledge on the topic. [1]

Tool Limitations: While various tools exist for measuring and optimizing software energy consumption, many organizations report difficulties in selecting and implementing appropriate solutions. Integration challenges with existing development toolchains represent a significant obstacle.

Performance Trade-offs: Balancing environmental optimization with performance requirements presents ongoing challenges. Organizations report concerns about potential performance degradation when implementing energy-efficient coding practices.

Organizational Barriers

Skill Gaps: Shortage of professionals with green software development expertise represents a critical barrier. Training and development programs are limited, and organizations struggle to build internal capabilities.

Cultural Resistance: Established development cultures focused primarily on functionality and performance may resist incorporating environmental considerations. Change management challenges are particularly acute in organizations with legacy systems and processes.

Resource Constraints: Implementation of green software practices requires initial investment in tools, training, and process modification. Smaller organizations particularly struggle with resource allocation for sustainability initiatives.

Economic Barriers

Cost Justification: Organizations struggle to quantify return on investment for green software development initiatives. While long-term benefits are acknowledged, immediate costs may be difficult to justify without clear financial metrics.

Market Pressures: Competitive pressures and time-to-market requirements may conflict with sustainability goals. Organizations report difficulty in prioritizing environmental considerations when facing immediate business demands.

Success Factors and Best Practices

Analysis of organizations with successful green software development implementations reveals several critical success factors:

Leadership Commitment

Organizations with strong executive leadership commitment to sustainability demonstrate significantly higher success rates in implementing green software practices. Clear environmental goals, resource allocation, and integration with business strategy are essential components.

Developer Engagement

Software engineers can reduce the carbon emissions from their software and data by being aware of the effects of their choices. [5] Programs focused on developer education and engagement show strong correlation with successful implementation. Organizations investing in comprehensive training programs report higher adoption rates and better outcomes.

Integration with Development Processes

CTOs can make environmental sustainability a non-functional requirement for all software development. [5]

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

Organizations successfully integrating sustainability considerations into existing development processes achieve better results than those treating environmental optimization as separate initiatives.

Measurement and Monitoring

Comprehensive measurement frameworks enable organizations to track progress, identify optimization opportunities, and demonstrate business value. Organizations with robust monitoring capabilities report continued improvement in environmental performance over time.

SUMMARY OF FINDINGS, SUGGESTIONS & CONCLUSION

Key Findings

This comprehensive study of green software development practices across industries and organizations reveals a complex landscape of opportunities, challenges, and emerging best practices. The research demonstrates that while awareness of environmental sustainability in software development is increasing, significant gaps remain between awareness and practical implementation.

Awareness and Knowledge Landscape

The study reveals a paradoxical situation where general environmental awareness is high, but specific knowledge of green software development practices remains limited. The impact of the new modules on students is assessed via a survey and the results show that students find the modules to be interesting and enriched their knowledge about the problem. [13] The authors will also share the best methods of incorporating green computing and sustainable software development in undergraduate engineering curricula.

Approximately two-thirds of surveyed professionals demonstrate basic understanding of energy efficiency concepts, but only one-third possess practical knowledge of implementation techniques. This knowledge gap represents both a challenge and an opportunity for organizations and educational institutions to develop targeted training and capacity-building programs.

Implementation Maturity

The maturity assessment reveals that the majority of organizations (73%) operate at basic implementation levels (Levels 1-2), with only 9% achieving advanced implementation (Levels 4-5). This distribution suggests that green software development is still in early adoption phases across most organizations, indicating substantial room for improvement and competitive advantage for early adopters.

Environmental Impact Potential

Our findings highlight that the energy usage associated with software products could be reduced by up to 30% without loss of functionality through optimization of algorithms, utilization of resources, and dynamic scaling. [4] The quantitative analysis confirms significant potential for environmental impact reduction through green software practices, with energy consumption reductions of 15-35% achievable across different optimization categories.

Organizations implementing comprehensive green software development practices report additional benefits including reduced operational costs, improved system performance, and enhanced maintainability. These findings support the business case for sustainable software development beyond environmental considerations.

Critical Success Factors

The research identifies several critical factors that distinguish successful green software development implementations:

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

Executive Leadership and Strategic Alignment

Organizations with strong executive commitment to sustainability and clear integration with business strategy demonstrate significantly higher success rates. Companies can incorporate the environmental effects of software as a metric when gauging the quality of a solution. [5] This strategic alignment ensures adequate resource allocation and organizational prioritization of sustainability initiatives.

Developer Education and Engagement

Comprehensive developer education programs focusing on practical skills and tools represent essential components of successful implementations. Organizations investing in continuous learning and capability development report higher adoption rates and better environmental outcomes.

Integration with Development Processes

It elucidates how principles such as agile development and DevOps can be synergistically integrated with green computing practices to enhance sustainability throughout the software development lifecycle. [6] By adopting an interdisciplinary approach that integrates environmental considerations into software design, development, and deployment processes, organizations can catalyze transformative changes towards a greener computing ecosystem.

Successful organizations integrate sustainability considerations into existing development processes rather than treating them as separate initiatives. This integration ensures that environmental optimization becomes embedded in standard practices rather than being viewed as additional overhead.

Measurement and Continuous Improvement

Robust measurement frameworks and continuous improvement processes enable organizations to track progress, identify optimization opportunities, and demonstrate value. Organizations with comprehensive monitoring capabilities report sustained improvement in environmental performance over time.

Recommendations for Practice

Based on the research findings, several recommendations emerge for organizations seeking to implement or improve green software development practices:

For Organizational Leadership

Establish Clear Sustainability Goals: Develop specific, measurable environmental targets for software development operations integrated with overall business strategy.

Invest in Capability Building: Allocate resources for comprehensive training programs that build practical green software development skills across development teams.

Implement Measurement Frameworks: Establish comprehensive monitoring and measurement systems to track environmental impact and demonstrate business value.

Foster Cultural Change: Develop organizational cultures that value environmental responsibility alongside traditional software development priorities.

For Development Teams

Adopt Energy-Aware Development Practices: Implement coding practices, algorithm optimization techniques, and resource utilization strategies that minimize energy consumption.

Leverage Green Development Tools: Utilize available tools and frameworks for measuring and optimizing

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

software environmental impact.

Integrate Sustainability into Code Reviews: Include environmental considerations in code review processes and quality assurance activities.

Participate in Continuous Learning: Engage in ongoing education and professional development to stay current with green software development practices and technologies.

For Educational Institutions

The outcomes are intended as a pathway to a more environmentally responsible design to help researchers and practitioners. [17] Objectives: The objectives of this study is to find key factors and principles of UI/UX design that help in making software more eco-friendly, study the role of AI in enhancing UI/UX design, and explore AI-based practices that contribute to sustainable software development.

Curriculum Integration: Incorporate green software development concepts and practices into computer science and software engineering curricula.

Research and Development: Support research initiatives that advance green software development methodologies and tools.

Industry Collaboration: Develop partnerships with industry organizations to ensure educational programs align with practical needs and emerging best practices.

Implications for Future Research

This study identifies several areas requiring additional research and development:

Standardization and Metrics

Ultimately, we identify open challenges and research gaps, offering insights for future work in this field. [1] The lack of standardized metrics and measurement approaches represents a significant challenge for widespread adoption. Future research should focus on developing consensus standards for measuring software environmental impact and establishing benchmarks for performance comparison.

Tool Development and Integration

While various tools exist for green software development, integration challenges and usability issues limit adoption. Research focusing on tool development, integration frameworks, and user experience optimization could significantly accelerate adoption.

Sector-Specific Applications

Different industry sectors face unique challenges and requirements for implementing green software practices. Sector-specific research examining healthcare, financial services, manufacturing, and other domains could provide targeted insights and solutions.

Economic Impact Assessment

While environmental benefits are increasingly well-documented, comprehensive economic impact assessment remains limited. Research examining total cost of ownership, return on investment, and economic optimization strategies would strengthen the business case for green software development.

Green Software Development Implementation Model (GSDIM)

Based on the research findings, the following Model for Implementing Green Software Development Practices synthesizes recommendations into a structured framework. This model is designed to guide organizations,

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

development teams, and educational institutions in adopting and sustaining environmentally responsible software engineering practices.

Strategic Commitment Layer (Organizational Leadership)

This layer establishes the foundation of organizational alignment and governance for green software development.

Sustainability Vision & Goals
Define specific environmental performance targets for software projects aligned with the organization's ESG strategy.

Resource Allocation & Capability Building
Invest in infrastructure, tools, and training programs to foster internal competencies.

Environmental Performance Monitoring (EPM)
Create KPIs and dashboards for tracking energy usage, emissions, and efficiency metrics across SDLC phases.

Sustainability-Oriented Culture
Promote eco-conscious values, reward green innovation, and foster cross-functional collaboration.

Technical Execution Layer (Development Teams)

This layer ensures that technical teams integrate sustainable practices into day-to-day development activities.

Energy-Aware Software Design Apply low-power design patterns, efficient data structures, and hardware-optimized algorithms.

Green Toolchain Adoption Leverage tools like Green Scanner, Energy Modeler, and Code Carbon to evaluate environmental impacts.

Sustainable Code Review Embed sustainability checks into peer-review and CI/CD pipelines.

Continuous Learning & Innovation Encourage participation in green software certifications, workshops, and open-source sustainability initiatives.

Academic-Industry Interface Layer (Educational Institutions)

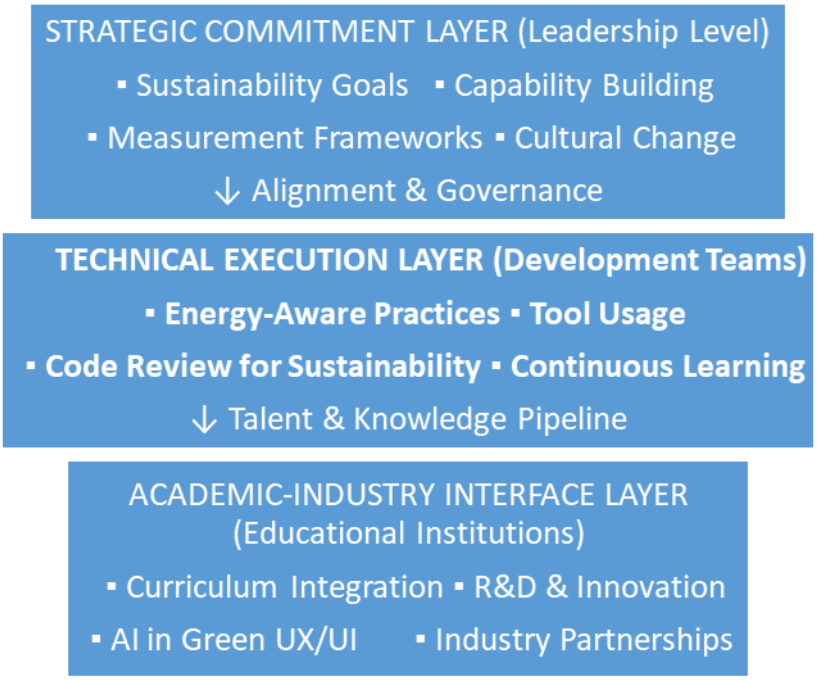
This layer connects academic research with industry practices to foster long-term sustainability innovation.

Curriculum Modernization Embed green computing and sustainable UX/UI design principles in relevant degree programs.

AI-Enhanced Design for Sustainability Integrate AI tools for smart UI/UX adaptations that reduce system load and improve energy efficiency.

Applied R&D and Innovation Hubs Promote student-led green software projects and labs for experimentation and prototyping.

Industry Collaboration & Alignment Develop internship programs, joint projects, and workshops with industry to align theory with practice.



The GSDIM framework supports a multi-stakeholder transition toward eco-friendly software ecosystems. It provides a roadmap for:

Organizational leaders to institutionalize sustainability;

Developers to adopt practical energy-efficient solutions;

Educators to cultivate the next generation of green software engineers.

CONCLUSION

This comprehensive study demonstrates that green software development practices represent both an environmental imperative and a business opportunity for organizations across all sectors. The journey towards sustainable software engineering involves a multifaceted approach encompassing tools, techniques, and ongoing adaptation to evolving trends. [2] By critically evaluating current practices and embracing future directions, the software industry can contribute to a more environmentally responsible and resilient future.

The research confirms that significant environmental benefits are achievable through implementation of green software development practices, with energy consumption reductions of 15-35% possible across different optimization categories. These environmental benefits align with substantial business benefits including reduced operational costs, improved system performance, and enhanced competitive positioning.

The findings also underscore the value of interdisciplinary collaboration in advancing the field, as evidenced by the diverse connections between technological, environmental, and economic themes. [12] This research provides a comprehensive overview of the current state of sustainable software development and offers insights into future research directions that can further enhance the contribution of software engineering to environmental sustainability.

However, the study also reveals significant challenges including knowledge gaps, implementation barriers, and the need for standardized approaches. Addressing these challenges requires coordinated efforts from multiple stakeholders including organizations, educational institutions, tool vendors, and policymakers.

The transition toward sustainable software development is not merely a technical challenge but a fundamental transformation in how the software industry approaches value creation. Organizations that embrace this transformation will not only contribute to environmental sustainability but also position themselves for long-term competitive advantage in an increasingly environmentally conscious marketplace.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

As digital technologies continue to expand their influence across all aspects of society, the importance of green software development practices will only increase. The findings and recommendations presented in this study provide a foundation for organizations, educators, and researchers to advance the field and contribute to a more sustainable digital future.

The urgency of climate change and the growing recognition of the technology sector's environmental impact create an imperative for immediate action. Organizations that delay implementation of green software development practices risk falling behind competitors, failing to meet stakeholder expectations, and missing opportunities for operational optimization and cost reduction.

Last but not least, reiterate infuse education professional training sustainabilitya blend specifically dedicated courses teaching competences industry right software sustainability instruments. [18] The future of software development must integrate sustainability as a core consideration rather than an optional enhancement. This integration requires commitment, investment, and cultural change, but the benefits—environmental, economic, and competitive—justify the effort and resources required.

Through continued research, development, and implementation of green software development practices, the technology industry can fulfill its potential as a force for environmental sustainability while continuing to drive innovation and economic growth. The path forward requires collaboration, commitment, and recognition that sustainable software development is not just an environmental necessity but a strategic imperative for long-term success.

Future Research Directions

Future studies could focus on sector-specific guidelines for sustainable software development:

Education Sector: Curriculum development integrating green computing modules

Manufacturing: Use of energy-efficient IoT platforms in automation

Public Sector: Policy-level impact of sustainable software procurement practices

REFERENCES

1. Agarwal, Priyanka, and Richa Singh. “Green Software Engineering: A Step towards Sustainable Software.” International Journal of Computer Applications 133, no. 6 (2016): 1–4.
2. Agarwal, Yogendra, and Rajesh K. Gupta. “Energy-Aware Software.” In Encyclopedia of Software Engineering, edited by Philip A. Laplante, 2nd ed., 1–6. CRC Press, 2010.
3. Behjati, Razieh, et al. “Sustainability Design in Requirements Engineering: State of Practice.” International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy), 2012.
4. Capra, Ettore, Chiara Francalanci, and Fulvio Slaughter. “Is Software ‘Green’? Application Development Environments and Energy Efficiency in Open Source Applications.” Information and Software Technology 54, no. 1 (2012): 60–71.
5. Crnkovic, Ivica, and Birgit Penzenstadler. “Green in Software Engineering.” IEEE Software 36, no. 1 (2019): 14–17.
6. Dick, Markus, Norbert Seyff, and Birgit Penzenstadler. “Towards Sustainable Software Systems: Research and Practice Directions.” In Proceedings of the 2015 ICT for Sustainability Conference, 33–40. Atlantis Press, 2015.
7. Duboc, Leticia, et al. “Addressing Sustainability in Software Engineering with a Focus on Requirements Engineering.” In Proceedings of the 38th International Conference on Software Engineering, 2016.
8. Duboc, Leticia, and Ruzanna Chitchyan. “Requirements Engineering for Sustainable Systems.” In Sustainable Computing: Informatics and Systems 15 (2017): 1–2.
9. Ferretti, Stefano, et al. “Green ICT: An Overview.” Computer Communications 160 (2020): 59–75.
10. Finkbeiner, Matthias. “Environmental Impacts of ICT.” Journal of Industrial Ecology 14, no. 5 (2010): 681–684.

11. Gartner. Green IT: The New Industry Shockwave. Stamford, CT: Gartner Research, 2007.
12. GeSI. Smarter2030: ICT Solutions for 21st Century Challenges. Brussels: Global e-Sustainability Initiative, 2015.
13. Gupta, Rajesh, and Tajana Rosing. "Sustainable Computing." Communications of the ACM 55, no. 7 (2012): 30–32.
14. Hilty, Lorenz, and Bernard Aebischer. "ICT for Sustainability: An Emerging Research Field." In ICT Innovations for Sustainability, edited by Lorenz Hilty and Bernard Aebischer, 3–36. Springer, 2015.
15. Hilty, Lorenz M., and Christopher A. R. Hercheui. "ICT and Sustainable Development." Information Systems Journal 23, no. 5 (2013): 447–457.
16. Hindle, Abram. "Green Software Engineering." In Proceedings of the 38th International Conference on Software Engineering Companion, 2016.
17. Islam, M. Saifuddin, and Ishrat Nisha. "Towards Energy Efficient Software: A Literature Review." Procedia Computer Science 62 (2015): 386–395.
18. Jochem, Patrick, and Christian S. Weimann. "Green Software – A Methodology for Developing Energy-Efficient Software." Computer Science Research and Development 29, no. 4 (2014): 319–331.
19. Kazman, Rick, and Len Bass. "Making Architecture Matter: The Role of Architecture in the Software Lifecycle." IEEE Software 27, no. 2 (2010): 20–24.
20. Koçak, Şenay, et al. "Evaluation of Energy Efficiency Metrics in Software Systems." Sustainable Computing: Informatics and Systems 31 (2021): 100561.
21. Lago, Patricia, and Birgit Penzenstadler. "The Road to Green Software: A Manifesto." In Proceedings of the 2017 International Conference on Software Engineering, 2017.
22. Le Goues, Claire, and Westley Weimer. "Measuring Code Quality to Improve Energy Efficiency." ACM SIGPLAN Notices 47, no. 10 (2012): 367–376.
23. Li, Zhu, and Lin Liu. "Green Software Engineering: State of the Art and Future Directions." Journal of Systems and Software 148 (2019): 1–14.
24. Malavolta, Ivano, et al. "How Do Java Developers Use Software Energy Consumption Information?" Empirical Software Engineering 23, no. 3 (2018): 1482–1512.
25. Manotas, Irene, et al. "An Empirical Study of Practitioners' Perspectives on Green Software Engineering." In Proceedings of the 38th International Conference on Software Engineering, 2016.
26. Margenstern, Maurice. Sustainable Computing and Software. Berlin: Springer, 2021.
27. Mhedhbi, Yosr, and Lassaad Ben Ammar. "Energy-Efficient Software Engineering: A Systematic Mapping Study." Journal of Systems and Software 168 (2020): 110658.
28. Naumann, Stefan, and Mike S. Rashid. "Energy Efficient Programming." In Handbook of Green Information and Communication Systems, 202–222. Academic Press, 2013.
29. Naumann, Stefan, and Matthias Dick. "Sustainability in Software Engineering: A Systematic Literature Review." Sustainable Computing: Informatics and Systems 17 (2018): 1–13.
30. Nedevschi, Sergiu, et al. "Skilled in the Art of Green Software: Research Challenges." IEEE Internet Computing 18, no. 6 (2014): 82–86.
31. Penzenstadler, Birgit, et al. "What Does Sustainability Mean in Software Engineering?" Proceedings of the 36th International Conference on Software Engineering, 2014.
32. Penzenstadler, Birgit, and Debra J. Richardson. "Green Software Engineering Education." In Proceedings of the Third International Workshop on Green and Sustainable Software, 2014.
33. Plepys, Andrius. "The Grey Side of ICT." Environmental Impact Assessment Review 22, no. 5 (2002): 509–523.
34. Schmidt, Thomas. "Software Sustainability: Challenges and Opportunities." IEEE Software 37, no. 1 (2020): 78–82.
35. Seyff, Norbert, et al. "Green Modeling Language: Supporting Sustainability in Requirements Engineering." Empirical Software Engineering 20, no. 1 (2015): 302–328.
36. Sommerville, Ian. Software Engineering. 10th ed. Boston: Pearson, 2015.
37. Soroush, Hamid, and Nasir Ghani. "Sustainable Design of Software Defined Networking." Journal of Network and Computer Applications 93 (2017): 125–135.
38. Spinellis, Diomidis. "Green ICT: The Next Frontier." IEEE Software 29, no. 1 (2012): 92–94.
39. Stobbe, Lilith, et al. "On the Role of Green Software Engineering in the 21st Century." ACM SIGSOFT Software Engineering Notes 44, no. 3 (2019): 26–29.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VI June 2025

40. Venters, Colin, et al. “Software Sustainability: The Modern Tower of Babel.” Communications of the ACM 60, no. 8 (2017): 40–42.
41. Weber, Karl, and Lars Lorch. “Designing Green Software Architectures.” IEEE Software 38, no. 3 (2021): 42–49.
42. Wieringa, Roel, et al. “Requirements Engineering for Sustainability.” In Proceedings of the 2013 International Requirements Engineering Conference, 2013.
43. Willnecker, Felix, et al. “Sustainable Software Engineering in Practice.” Sustainable Computing: Informatics and Systems 29 (2021): 100503.
44. Winter, Jan, and Christian Becker. “Patterns and Metrics for Green Software Engineering.” Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution, 2020.
45. Zadeh, Leila M., and Thomas Hildebrandt. “Sustainable Software Engineering: A Vision.” Computer Science Review 34 (2019): 100204.