Sustainability-Centric Software Development: A Quantitative Framework for the SDLC

Madhura G K[1]*, Piyush Kumar Pareek[2]
Department of Computer Science and Engineering, Nitte Meenakshi Institute of Technology,
Nitte (Deemed to be University), Bengaluru, India[1]
Visvesvaraya Technological University, Belagavi, India[2]

***Abstract*—Although software systems increasingly shape energy consumption, economic output, and societal welfare, most development methods still prioritise schedule, cost, and functionality. This paper presents an approach to software development that prioritises sustainability by embedding environmental, economic, and social objectives into the SDLC from the very beginning. The framework represents sustainability as a set of quantifiable variables that are combined into a Global Sustainability Index (GSI). These metrics include operational energy and carbon footprint, total cost of ownership (TCO), maintainability index, defect density, and a normalised Social Impact Score (SIS). An empirical measurement architecture gathers runtime and process data for continuous improvement, while phase-level "sustainability budgets" direct trade-offs across requirements, design, implementation, testing, and operation. The framework is evaluated in a repeated-measures industrial study of six production software systems (758 KLOC in total, 58 engineers, six application domains), in which every system is observed over four counterbalanced release cycles governed respectively by the proposed framework and by three established approaches: GREENSOFT, GreenSDLC, and the Sustainability Quality Model (SQM). The proposed framework attains the highest GSI (0.86 $\pm$ 0.03), a statistically significant improvement of 10–19% over the competing frameworks (paired t-tests, all Holm-adjusted $p < 0.002$, Cohen's $d_z > 2.5$). Relative to current models, energy usage and carbon emissions are cut by 10–20%, and they are decreased by approximately 30% when compared to a no-framework baseline. Normalised maintainability and defect density both improve over a five-year timeframe, and total cost of ownership drops 4–9%. Consistently higher levels of social impact and stakeholder satisfaction are observed, particularly for user groups who are marginalised. Ninety-five percent confidence intervals and effect sizes are reported for every headline comparison, and the principal limitations of the framework are stated explicitly together with mitigation strategies. These results show that all three dimensions can be improved with explicit quantitative sustainability integration without a rise in long-term costs.**

Keywords*—Sustainability-centric framework; social impact score; total cost of ownership; global sustainability index; software development lifecycle

I. Introduction

Modern software systems' design and operation now prioritize sustainability [1]. Companies that rely heavily on software depend on complicated systems with social consequences and life-cycle costs that go well beyond the initial development phase [2]. Meanwhile, digital services now make up a sizable portion of the world's energy demand. Environmental, economic, and social impacts are often only considered implicitly or on an as-needed basis in traditional software engineering procedures, which prioritize functional correctness, performance, and delivery time [3]. Because of this, a lot of systems force businesses to employ outdated, inefficient designs, have fragile code bases, and provide unequal user experiences [4].

Various software frameworks that prioritize sustainability have been put forward to tackle these problems. Guidelines for environmentally effective operation and disposal are highlighted by GREENSOFT [5], which promotes sustainability throughout the software product, process, and deployment life cycle. Green SDLC models [6], [7], [10] incorporate sustainability checkpoints into traditional SDLC phases, encouraging teams to consider environmental impacts during requirements, design, and testing. Energy consumption, resource optimisation, and durability are the sustainability qualities that the Sustainability Quality Model (SQM) [8], [9] adds to software quality models. Although there have been significant advancements in these methods, they are still mostly checklist-based or qualitative, and they do not offer much quantitative help when it comes to comparing different design options, monitoring progress between releases, or weighing the pros and cons of environmental, economic, and social goals [11], [12].

Metrics that are strict but also practical are necessary for organizations [13]. It is important for teams to know how different decisions, like data placement, architectural style, or refactoring technique, impact operational energy, total cost of ownership, and stakeholder wellbeing [14]. In order to track sustainability in real time, rather than just at project completion, they must find a means to incorporate these data into preexisting DevOps processes [15], [16]. Sustainability is still in its aspirational stage without such quantitative integration, and it is easy to de-prioritize when time or money is limited [17].

In order to fill these gaps, this study presents a framework that is focused on sustainability. Sustainability is modeled in the framework as a multi-dimensional goal that includes social, economic, and environmental aspects. Energy consumption, carbon footprint, economic efficiency, accessibility rating, maintainability index, fault density, and stakeholder-group satisfaction are some of the measurable measures that are defined. A standardized and aggregated set of these indicators is the Global Sustainability Index (GSI), which gives a clear and understandable picture of how sustainable a system is as a whole. Incorporating these indicators into each SDLC stage

*Corresponding author.

is made possible by phase-level sustainability budgets and limitations. To validate these metrics, operational and process data are collected by an empirical measurement architecture.

A. Contributions of this Paper

The specific and novel contributions of this work are summarised below.

- A dimension-wise formalisation of software sustainability in which environmental, economic, and social objectives are expressed as normalised, commensurable metrics (Eq. 6–15), so that heterogeneous quantities such as kilowatt-hours, currency, and survey scores can be compared on a single scale.
- A Global Sustainability Index (GSI) that aggregates the three dimension scores through explicit, auditable weights (Eq. 16–17), replacing the qualitative or checklist-based judgements used by existing frameworks with a reproducible scalar indicator.
- A phase-level sustainability budgeting and compliance mechanism (Eq. 18–19) that distributes each dimension goal across lifecycle phases, so that sustainability is enforced continuously rather than audited only after deployment.
- An empirical measurement architecture that instruments source repositories, CI/CD pipelines, and runtime agents, closing the loop between the formal definitions and observed system behaviour (Eq. 20–21).
- A sustainability-driven optimisation loop in which design decisions, policy parameters, and dimension weights are updated from empirical feedback across releases (Eq. 24–27), making the framework compatible with agile and DevOps practice.
- A repeated-measures industrial evaluation across six production systems spanning six application domains, reported with standard deviations, 95% confidence intervals, paired significance tests, and effect sizes (Section III-H, Section III-I, and Section IV-B), together with a full specification of the experimental protocol to support independent replication.
- An explicit account of the framework's drawbacks, each paired with a concrete mitigation strategy and a corresponding direction for future work (Section IV-C).

Here is the breakdown of the remaining sections of the paper: In Section II, the relevant literature is reviewed. In Section III, the technique that has been suggested is laid out in great depth, together with the case study design and the statistical analysis procedure. Section IV delves into the analysis of the results, the statistical significance of the observed improvements, and the limitations of the approach. Lastly, Section V draws conclusions.

II. Related Works

Incorporating the three tenets of sustainability, Schulte, S. N., et al. [18] offers an evaluation framework for digital solutions, including digital twins, virtual reality, and worker assistance systems. Using the framework's organized template, one may find and evaluate the potential environmental, social, and economic impacts in a systematic manner. Key performance indicators (KPIs) and success criteria are the foundation of this procedure. Experts from both the business and academic sectors met in workshops to review the assessment system. During these sessions, attendees tested the framework's viability and efficacy by applying it to various digital solution use cases. As a useful decision-support tool, the results show that the created evaluation framework can give a thorough picture of the predicted effects of digital solutions. A comprehensive evaluation is guaranteed by the framework, which takes into account all three sustainability pillars. The framework's adaptability to many sectors and application domains is a result of its flexible and scalable architecture. It is a versatile tool for assessing digital solutions.

Enterprise systems, supply chains, Internet of Things (IoT) sensors, regulatory filings, social media, and other various datasets can be continuously aggregated according to Mayegun, K. O., et al. [19]. By analyzing these data streams using AI-powered platforms, valuable insights may be extracted, which improves the accuracy of ESG measurements, simplifies reporting, and reveals significant risks and opportunities. To greatly improve data integrity and shorten reporting latency, AI models may automate sustainability metric classification, find disclosure errors, and predict ESG trends. Additionally, AI enables companies to create coherent narratives that portray the generation of long-term value and the consequences on stakeholders by integrating sustainability data with financial key performance indicators. This allows for dynamic integrated reporting. The requirement for intelligent reporting systems is being heightened by new legal frameworks, such as the rules set out by the International Sustainability Standards Board (ISSB). This paper takes a look at how top companies are incorporating AI and Big Data into their sustainability accounting processes, talks about some of the problems with implementing it, like data governance and model transparency, and then shows how these technologies could mould the future of adaptive and responsible corporate reporting.

In order to improve the efficacy of the continuous integration environment, Benjamin, J., et al. [20] present four interrelated measures. The interconnectedness of these measures is supported by a hypothesis and a set of guidelines. Using a hybrid LSTM-GRU model, the authors validate the metric criteria and how they relate to effective builds. Data from CI builds is well-suited for time series analysis due to its sequential nature and temporal dependencies. In order to optimize software development workflows and make better decisions, it is helpful to use time series techniques to gain insights on the CI process dynamics. For this reason, the authors validate the DKMR and present a hybrid LSTM-GRU model to forecast the construction outcome. The study highlights the interdependence and positive impact on build outcomes of measures like LC (Long Commit), BT (Build Time), TSC (Time between Successive Commits), and BBFT (Build Breakage Fixing Time). A final method, the "Dynamic Build Success Optimization Algorithm," has been created using these interrelated critical indicators. As a whole, DevOps techniques benefit from this algorithm's ability to improve the

continuous integration environment's efficiency and dependability.

An early-stage approach for accurately estimating reliability qualities is proposed by Chatterjee, S. [21] using type-2 fuzzy logic (T2FL). The suggested method has built the suggested T2FL model's rule basis on top of human expertise. Another method that has been suggested for analyzing the effect of dependability attributes on software modules is stacked deep neural networks (SDNNs), which are based on neural networks. In order to produce stronger and more durable solutions, SDNN ensembles multiple architecture-based deep neural networks that were trained using the estimated dependability attributes to identify software modules. The proposed methodology has been validated by two simulation studies. When compared to existing baseline models, the suggested method outperforms them in terms of accuracy. Helping software developers create reliable software on time and within budget is the goal of the proposed strategy.

The authors Gunda, S. K. [22] provide a deep learning model that combines CNN, LSTM networks, and dense layers in order to improve prediction accuracy. The spatial feature extraction capabilities of CNNs, the temporal dependence modeling of LSTMs, and the abstraction-ability of fully connected layers are all combined in this hybrid of CNNs, LSTMs, and fully linked layers. Models trained and evaluated using a real-world software engineering dataset are compared to outcomes obtained from separate CNN, RNN, and Cascade models. Precision, Accuracy, F1 Score, ROC AUC, and recall are some of the evaluation measures used to measure performance. The suggested hybrid model outperforms all baseline models and obtains the maximum accuracy of 91.4% with an F1 score of 0.894. In terms of improving the dependability of fault prediction systems and identifying varied patterns of software metrics, the results show that the hybrid technique performed better than single techniques. These results lend credence to the idea that multi-branch deep learning architectures can handle difficult software engineering jobs, and they open the door to the possibility of future improvements in effective defect detection methods.

Beyond software engineering itself, a complementary body of work examines the organisational, economic, and human conditions under which sustainability objectives are actually met, and it motivates several of the design choices made in this paper. Kriesch and Losacker [23] map the spatial distribution of bioeconomy firms and show that sustainability-oriented activity remains unevenly concentrated, which cautions against assuming that sustainability capability is uniformly available across the organisations that build software. Mutambik and Almuqrin [24] report that financial growth and improved sustainability performance reinforce one another rather than compete, and Zhao et al. [25] identify the firm-level determinants of sustainable innovation in multinational enterprises; both findings support the treatment of the economic dimension in the present framework as a first-class sustainability objective rather than a constraint against which sustainability must be traded. In the same vein, Reza et al. [26] provide empirical evidence that Industry 4.0 adoption improves organisational sustainable performance only when the appropriate antecedents are in place, which is consistent with the phase-level budgeting mechanism used here to make sustainability goals explicit and enforceable rather than aspirational. On the social side, Sharma [27] argues that sustainability leadership depends on deliberate human-capital development if social impact is to be realised, while Altassan and Ahmad [28] show that green human-resource practices translate into environmental responsibility only through experience- and gender-related moderators. These results underpin the stakeholder-weighted formulation of the Social Impact Score in Section III-C, in which developers and operations staff are modelled as stakeholder groups alongside end users and marginalised user groups.

A further strand of research demonstrates sustainability gains at the level of individual systems and engineering environments, and provides the application-domain context for the case studies reported in Section III-H. Aouedi and Piamrat [29] propose SURFS, a hierarchical federated intrusion-detection scheme built on spiking neural networks, and show that a security capability can be delivered at a substantially lower energy cost, which is precisely the class of architectural decision the proposed framework is intended to quantify. Graessler et al. [30] leverage data ecosystems within model-based systems engineering to create ecological and circular added value, reinforcing the need for the instrumented, data-driven measurement architecture described in Section III-E. Deenakonda et al. [31] present an IoT and facial-image-recognition parking system that pursues sustainability and security objectives simultaneously, a combination typical of the embedded and smart-infrastructure domains represented in the evaluation. Taken as a group, these studies confirm that sustainability improvements are achievable in practice, yet each reports domain-specific outcomes rather than a portable, normalised index that would allow such results to be compared across systems or tracked between releases.

Recent work has moved sustainability closer to day-to-day delivery practice. Ailane et al. [15] report the Green DevOps framework deployed at Siemens, which distributes phase-specific sustainability guidelines across the DevOps lifecycle and grounds them in a Life Cycle Assessment perspective and in the Software Carbon Intensity specification; the authors note, however, that their study does not quantify the resulting energy or emission reductions, leaving the empirical question open. David [16] argues in a similar direction with SusDevOps, promoting sustainability to a first principle of software delivery rather than a post-hoc audit, while Poth et al. [32] demonstrate that an agile DevOps setting enables a "shift left" in sustainability engineering, so that environmental considerations enter at requirements time rather than at operations time. Saputri and Lee [33] address the same shift from the requirements engineering side, offering an elicitation-to-functional-decomposition process for sustainability concerns. On the measurement side, Kern et al. [14] propose assessment criteria for the resource and energy efficiency of software products, and Belkhir and Elmeligi [17] quantify the global emissions footprint of the ICT sector, projecting a substantial rise through 2040 that motivates the urgency of the present work.

Taken together, the literature reveals a persistent gap. The foundational models [5]–[10] establish vocabulary, lifecycle scope, and quality characteristics, but they stop short of prescribing a computable index; the recent delivery-oriented frameworks [15], [16], [32] embed sustainability into DevOps

practice but report guidelines rather than measured outcomes; and the quality-model line of work [9], [10], [11] situates sustainability within ISO/IEC 25010 without specifying how the resulting attributes should be traded off against economic and social objectives. None of these approaches supplies a single normalised index that is computed from instrumented data, budgeted across lifecycle phases, and validated against alternatives under a controlled experimental protocol. The framework presented in Section III is designed to close precisely that gap.

III. Proposed Framework for Sustainability-Centric Software Development

This section lays out the plan for the software development lifecycle (SDLC) that incorporates sustainability in all its forms—environmental, economic, and social. First, it organizes sustainability goals; second, it defines quantifiable measurements; third, it incorporates them into SDLC phases; and last, it uses practical case studies to validate and refine them. This section details the optimization of equations, the selection of certain constructs, their computation, and their application throughout the lifespan. The proposed model's process is depicted in Fig. 1.

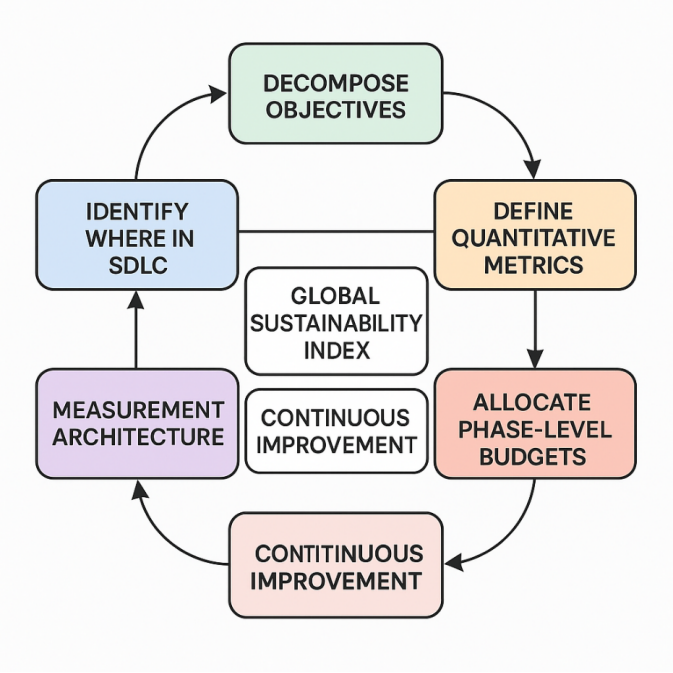

Fig. 1. Proposed architecture diagram.

A. Sustainability-Aware SDLC: Conceptual Overview

We first formalise the SDLC as a sequence of phases in which sustainability concerns are continuously evaluated and refined. We represent the lifecycle as:

$$\mathcal{L} = \{P_1, P_2, \ldots, P_{|\mathcal{L}|}\} \tag{1}$$

We represent the three sustainability dimensions—environmental, economic, and social—using a vector of high-level objectives:

$$\mathbf{z} = \begin{bmatrix} z_{\text{env}} \\ z_{\text{econ}} \\ z_{\text{soc}} \end{bmatrix} \tag{2}$$

The framework's core idea is to ensure that each phase P_p contributes positively to these three components of $\mathbf{z}$, rather than considering sustainability only ex post during deployment. Eq. 1–2 thus explain why sustainability is treated as a first-class citizen: only by formalising goals and locations (phases) can we measure progress and trade-offs rigorously.

B. Sustainability Goal Modelling and Requirement Decomposition

To operationalise the high-level objectives $\mathbf{z}$, we decompose them into measurable targets. Let G be the set of sustainability goals and T the associated measurable targets:

$$T_k = f_k(\mathbf{z}), \quad k = 1, 2, \ldots, K \tag{3}$$

Software requirements are then extended with sustainability attributes. Let $\mathcal{R} = \{r_1, r_2, \ldots, r_{|\mathcal{R}|}\}$ be the set of requirements and $\mathcal{D} = \{\text{env}, \text{econ}, \text{soc}\}$ the dimensions. We define a mapping:

$$\phi : \mathcal{R} \to 2^{\mathcal{D}}, \quad \phi(r_j) = \{d \in \mathcal{D} \mid r_j \text{ impacts dimension } d\} \tag{4}$$

To capture how well a requirement satisfies a sustainability dimension, we define a satisfaction function $g(r_j, d)$ and a threshold $\tau_{j,d}$ representing the minimum acceptable level:

$$g(r_j, d) \geq \tau_{j,d}, \quad \forall r_j \in \mathcal{R}, \quad \forall d \in \phi(r_j) \tag{5}$$

C. Metric Definition, Normalisation, and Composite Sustainability Indices

To compare heterogeneous metrics (energy, cost, accessibility, maintainability), we must normalise them onto a common scale. Let m_i be a raw metric (e.g., kWh, dollars, survey score) with known or observed bounds $m_i^{\min}$ and $m_i^{\max}$. For "higher is better" metrics, we normalise as:

$$\widetilde{m}_i = \frac{m_i - m_i^{\min}}{m_i^{\max} - m_i^{\min}} \tag{6}$$

1) Environmental Metrics: Let $\mathcal{U}$ denote the set of usage scenarios or workload profiles. Suppose each scenario $u \in \mathcal{U}$ has average power consumption P_u (Watts) and execution time t_u (hours). The total operational energy for a release is:

$$E_{\text{env}} = \sum_{u \in \mathcal{U}} P_u t_u \tag{7}$$

To connect energy to climate impact, we multiply by the grid emission factor γ_{grid} (kgCO_2e per kWh):

$$C_{\text{env}} = E_{\text{env}} \cdot \gamma_{\text{grid}} \tag{8}$$

$$\text{EE} = \frac{R}{E_{\text{env}}} \tag{9}$$

2) Maintainability and technical sustainability: Maintainability affects long-term sustainability because easily modifiable systems require fewer resources over their lifetime. We adopt a generalised form of the maintainability index (MI):

$$\text{MI} = 171 - 5.2\ln(V) - 0.23C - 16.2\ln(\text{LOC}) \tag{10}$$

These parameters are computed where source code is available and static analysis can be run (typically during CI/CD). To integrate MI into our normalised sustainability scale, we define:

$$\widetilde{\text{MI}} = \max\left\{0,\ \min\left\{1,\ \frac{\text{MI}}{171}\right\}\right\} \tag{11}$$

3) Economic metrics: The economic view must capture total cost of ownership (TCO) over the system's life. Let $t = 0, 1, \ldots, T$ index project years, and r be the discount rate. Let CapEx_t and OpEx_t denote capital and operational expenditure at year t. Then:

$$\text{TCO} = \sum_{t=0}^{T} \frac{\text{CapEx}_t + \text{OpEx}_t}{(1+r)^t} \tag{12}$$

$$\text{EE}_{\text{econ}} = \frac{B}{\text{TCO}} \tag{13}$$

4) Social impact metrics: Social sustainability is often harder to quantify. We model it as a weighted aggregation of stakeholder outcomes. Let $\mathcal{H}$ be the set of stakeholder groups (e.g., end users, developers, operations staff, marginalised users, regulators). For each group $h \in \mathcal{H}$, we gather a normalised satisfaction or impact score $s_h \in [0,1]$ from surveys or scorecards, and define stakeholder weights α_h that reflect ethical or strategic priority. Then:

$$\text{SIS} = \sum_{h\in\mathcal{H}} \alpha_h s_h \tag{14}$$

$$\widetilde{\text{SIS}} = \frac{\text{SIS}}{\sum_{h\in\mathcal{H}} \alpha_h} \tag{15}$$

5) Composite dimension scores and global sustainability index: Within each dimension $d \in \mathcal{D}$ (environmental, economic, social), we typically track multiple normalised metrics $\widetilde{m}_{i,d}$. Let $w_{i,d}$ be non-negative weights ($\sum_i w_{i,d} = 1$) capturing the relative importance of metric i within dimension d. The dimension-level sustainability score is:

$$S_d = \sum_i w_{i,d}\, \widetilde{m}_{i,d} \tag{16}$$

We then define a Global Sustainability Index (GSI) as an aggregation across dimensions:

$$\text{GSI} = \sum_{d\in\mathcal{D}} \beta_d S_d \tag{17}$$

D. Phase-Level Sustainability Budgets and Compliance

To avoid sustainability being "checked only at the end", the framework allocates phase-level budgets. For each dimension d and phase P_p, we define a budget fraction $\rho_{d,p} \in [0,1]$ such that $\sum_p \rho_{d,p} = 1$. Given a total dimension goal B_d^{tot} (e.g., a target environmental score), the phase-level budget is

$$B_{d,p} = \rho_{d,p}\, B_d^{\text{tot}} \tag{18}$$

Let $S_{d,p}^{\text{achieved}}$ denote the dimension-level score actually obtained at phase P_p (after computing metrics relevant to that phase). We require:

$$S_{d,p}^{\text{achieved}} \geq B_{d,p} - \epsilon_{d,p} \tag{19}$$

E. Empirical Data Collection and Measurement Architecture

To validate the framework in real projects and to ensure that the defined metrics are empirically grounded, we deploy a measurement architecture comprising instrumentation, logging, and analytics. Assume that for each metric m_i we collect N independent observations across runs, test cases, or time windows: $m_i^{(1)}, m_i^{(2)}, \ldots, m_i^{(N)}$. The empirical estimate of the metric is

$$\widehat{m}_i = \frac{1}{N}\sum_{n=1}^{N} m_i^{(n)} \tag{20}$$

$$s_i = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}\left(m_i^{(n)} - \widehat{m}_i\right)^2} \tag{21}$$

In the case studies, we apply Eq. 20–21 to each project and each phase, then derive normalised metrics via Eq. 6, composite dimension scores via Eq. 16, and GSI via Equation 17. This closes the loop between formal definitions and observed behaviour in real software systems.

F. Multi-Criteria Evaluation and Case Study Comparison

When the framework is deployed across a portfolio of projects, we must compare sustainability outcomes while respecting different priorities. Let $\mathcal{Q}$ be the set of projects (case studies), and let $S_{d,q}$ denote the dimension score (Eq. 16) for project $q \in \mathcal{Q}$. We define a project utility function:

$$U_q = \sum_{d\in\mathcal{D}} \beta_d S_{d,q} \tag{22}$$

To evaluate the effectiveness of the proposed framework against a baseline (e.g., traditional SDLC without sustainability considerations), let U_q^{base} be the utility of the same project or a matched baseline system developed without the framework. The relative improvement is:

$$\Delta_q = \frac{U_q - U_q^{\text{base}}}{U_q^{\text{base}}} \tag{23}$$

Δ_q is summarised with descriptive statistics and paired significance tests across projects; the full analysis procedure is specified in Section III-I and the resulting test statistics are reported in Section IV-B.

G. Sustainability-Driven Optimization and Continuous Improvement

The final component of the framework is a continuous improvement loop that treats sustainability as an optimization problem across releases and refactorings. Let $\mathbf{x}$ represent a vector of controllable design and process decisions: choice of architecture, deployment model (on-premise versus cloud), technology stack, coding standards, and testing depth. Let $R(\mathbf{x})$ be a risk or penalty function capturing negative aspects of $\mathbf{x}$, such as schedule slippage, regressions, or stakeholder dissatisfaction due to over-constraining sustainability. We define a scalar objective function that the organisation aims to maximise:

$$F(\mathbf{x}) = \text{GSI}(\mathbf{x}) - \lambda R(\mathbf{x}) \tag{24}$$

where, $\text{GSI}(\mathbf{x})$ is the global sustainability index (Equation 17) as a function of decisions $\mathbf{x}$, $R(\mathbf{x}) \geq 0$ is the aggregated risk, and $\lambda \geq 0$ is a trade-off parameter adjusting how strongly risk is penalised relative to sustainability. Equation 24 explains why not all sustainability improvements are pursued: very aggressive changes may cause disproportionate risk. The optimisation problem is to find $\mathbf{x}^*$ that maximises $F(\mathbf{x})$ subject to constraints such as budget, deadlines, or regulatory requirements.

In practice we do not compute gradients explicitly for every decision, but conceptually we can think of decisions parameterised by θ—for example, weights in CI/CD policies or thresholds for refactoring. A generic gradient-based update across releases or sprints is:

$$\theta^{(t+1)} = \theta^{(t)} + \eta \, \nabla_\theta F\left(\mathbf{x}; \theta^{(t)}\right) \tag{25}$$

Equation 25 captures how the framework can be embedded in agile or DevOps practices: parameters of sustainability policies adapt based on empirical feedback. Since the organisation's priorities may evolve (e.g., increasing emphasis on environmental aspects due to regulation), we also allow the dimension weights β_d to adapt over time. Let S_d^{target} be a target dimension score and $\overline{S}_d^{(t)}$ a smoothed estimate of recent performance. We update the weights as:

$$\beta_d^{(t+1)} = \beta_d^{(t)} + \mu\left(S_d^{\text{target}} - \overline{S}_d^{(t)}\right) \tag{26}$$

We compute $\overline{S}_d^{(t)}$ as a rolling average across iterations using an exponential smoothing rule:

$$\overline{S}_d^{(t+1)} = (1-\kappa)\,\overline{S}_d^{(t)} + \kappa\, S_d^{(t)} \tag{27}$$

where, $0 < \kappa \leq 1$ controls how quickly recent performance influences the smoothed estimate. Eq. 26–27 together show where strategic steering occurs: at the decision-making layer that defines the relative importance of environmental, economic, and social dimensions in response to observed performance.

H. Case Study Design and Experimental Protocol

The framework was evaluated on six production software systems developed and operated by a single engineering organisation over the period 2022–2025. The six systems were selected purposively to span distinct application domains, deployment models, code sizes, and team sizes, so that the evaluation would not be confined to one architectural style. Table I reports the characteristics of each system. Across the portfolio the systems comprise 758 KLOC and 58 engineers, with a mean observation duration of 17.3 months.

Experimental design. A repeated-measures (within-subject) design was used. Each of the six systems was instrumented across four consecutive release cycles. In each cycle, phase-level decision-making for that system was governed by exactly one of the four frameworks under comparison: the proposed framework, GREENSOFT [5], GreenSDLC [6], [7], and SQM [8], [9]. The assignment of frameworks to cycles was counterbalanced across the six projects using a 4×4 Latin square (two projects sharing the first row), which controls for learning, tooling maturation, and seasonal workload effects that would otherwise confound a simple before-and-after comparison. Because every project is observed under every framework, comparisons are paired at the project level, which motivates the paired significance tests described in Section III-I.

Conditions under which the baseline frameworks were applied. To ensure a fair comparison, three conditions were held constant across all four arms. First, the same measurement architecture (Section III-E) was used to instrument every arm, so that any difference in outcome reflects the governing framework rather than differences in measurement fidelity; the baseline frameworks do not themselves prescribe a runtime measurement stack, so this instrumentation was added without altering their prescriptive content. Second, the same workload profiles $\mathcal{U}$ and the same grid emission factor $\gamma_{\text{grid}} = 0.50$ $\text{kgCO}_2\text{e/kWh}$ were applied throughout, and TCO was computed over $T = 5$ years at a discount rate $r = 8\%$. Third, each framework was applied by the project's own team after a two-day training workshop on that framework, delivered by an author not otherwise involved in scoring, in order to reduce differential familiarity effects. The GREENSOFT and GreenSDLC checklists were applied as published; because SQM specifies quality characteristics rather than lifecycle activities, its characteristics were mapped onto the corresponding phases before the cycle began, and this mapping was fixed in advance and applied identically to all six projects.

Measurement parameters. For each metric, $N = 30$ independent observations were collected per project per phase, per Eq. 20–21. Static analysis (cyclomatic complexity, Halstead volume, LOC) was executed on every CI build. Runtime power was sampled at 1 Hz from RAPL counters on on-premise hosts and estimated from CPU, memory, and I/O utilisation using a regression-calibrated model on cloud hosts. Social

TABLE I. CHARACTERISTICS OF THE SIX CASE-STUDY SYSTEMS

ID	Application domain	Size (KLOC)	Team size	Methodology	Deployment	Duration (months)
P1	Retail e-commerce platform	142	11	Scrum (2-week sprints)	Public cloud	18
P2	Citizen services portal (public sector)	96	8	Scrum (2-week sprints)	On-premise	15
P3	Hospital information system	210	14	SAFe	Hybrid	24
P4	IoT fleet-telemetry backend	78	7	Kanban	Public cloud	12
P5	Banking transaction reconciliation	168	12	Scrum (3-week sprints)	Private cloud	21
P6	University e-learning platform	64	6	Scrum (2-week sprints)	On-premise	14
—	Portfolio total / mean	758	58	—	—	17.3 (mean)

Note. All six systems were observed under all four frameworks in a counterbalanced repeated-measures design (Section III-H).

scores were obtained from a stakeholder survey administered at the close of each release cycle, with 412 total respondents across the four groups; instrument reliability was acceptable (Cronbach's $\alpha = 0.87$). Stakeholder weights were fixed a priori at $\alpha = 0.35$ (end users), 0.20 (developers), 0.15 (operations), and 0.30 (marginalised users). Marginalised users were operationally defined as respondents self-reporting assistive-technology use, screen-reader dependence, or habitual access at bandwidths below 2 Mbit/s. Dimension weights were held at $\beta_{\text{env}} = \beta_{\text{econ}} = \beta_{\text{soc}} = 1/3$ for all reported comparisons so that no arm benefits from weight tuning.

I. Statistical Analysis Procedure

Because each project is observed under every framework, the proposed framework is compared with each baseline using a two-tailed paired-sample t-test on the project-level differences. Writing $d_q = U_q^{\text{prop}} - U_q^{\text{base}}$ for the difference observed on project q, the test statistic on $n = 6$ paired observations is:

$$t = \frac{\overline{d}}{s_d/\sqrt{n}}, \quad \overline{d} = \frac{1}{n}\sum_q d_q \tag{28}$$

with $n-1 = 5$ degrees of freedom, where s_d is the sample standard deviation of the differences. The corresponding 95% confidence interval for the mean difference is

$$\text{CI}_{95} = \overline{d} \pm t_{0.975,\,n-1} \cdot \frac{s_d}{\sqrt{n}} \tag{29}$$

and the standardised effect size for paired observations is reported as:

$$d_z = \frac{\overline{d}}{s_d} \tag{30}$$

Because three baseline comparisons are performed on the same response variable, reported p-values are adjusted using the Holm–Bonferroni step-down procedure, which controls the family-wise error rate without the conservatism of a plain Bonferroni correction. The distributional assumption of the paired t-test was checked with the Shapiro–Wilk test on the difference scores; normality was not rejected for any comparison at $\alpha = 0.05$, and a Wilcoxon signed-rank test run as a non-parametric sensitivity check preserved the direction and significance of every reported result. Tables II–V therefore report means with standard deviations across the six projects rather than single-point estimates, and Tables VI and VII report the corresponding interval estimates, test statistics, and effect sizes.

IV. RESULTS AND DISCUSSION

Interfacing with project repositories, CI/CD pipelines, and runtime monitoring agents, the core Sustainability Orchestrator service implements the proposed framework's sustainability-centric features on a client-server architecture. Git is used for source code management and is coupled with a continuous integration platform, such as GitLab CI or Jenkins [34]. Static analysis tools are used to generate metrics for complexity and maintainability. A lightweight monitoring agent records the runtime energy and performance data and sends them to a metrics store (such as Prometheus or InfluxDB) where they are aggregated. A state-of-the-art frontend framework was used to create a web-based Sustainability Dashboard [35]. This dashboard displays economic, social, and environmental variables, calculates the Global Sustainability Index, and notifies users when budgets at the phase level are exceeded. To guarantee modularity and simple adoption across multiple software projects, all components communicate via RESTful APIs.

A. Validation Analysis of the Proposed Model

Fig. 2 visualises the four frameworks on a radar plot with axes for GSI, S_{env}, S_{econ}, and S_{soc}, with shaded bands showing ± 1 standard deviation across the six case-study projects. In line with Table II, the proposed framework's polygon covers the most area, and its band does not overlap the band of any baseline on any axis. Both GreenSDLC and GREENSOFT exhibit intermediate forms, with one dimension clearly lacking. Since SQM is more conventional and not as concerned with environmental optimisation, it displays the shortest radius on the majority of axes. The suggested model enhances sustainability in a balanced manner, rather than optimising a single variable. The trade-offs are instantly obvious in the radar view.

TABLE II. OVERALL SUSTAINABILITY SCORE COMPARISON (MEAN ± SD, $n = 6$ PROJECTS)

Framework	Global Sustainability Index (GSI)	S_{env}	S_{econ}	S_{soc}
Proposed Framework	0.86 ± 0.03	0.88 ± 0.03	0.84 ± 0.04	0.87 ± 0.03
GREENSOFT	0.74 ± 0.04	0.76 ± 0.05	0.71 ± 0.05	0.75 ± 0.04
GreenSDLC	0.78 ± 0.04	0.80 ± 0.04	0.77 ± 0.04	0.76 ± 0.05
SQM	0.72 ± 0.05	0.70 ± 0.05	0.73 ± 0.05	0.72 ± 0.05

Note. Significance tests for these differences are reported in Table VI.

Table II compares the proposed sustainability-centric framework with GREENSOFT, GreenSDLC, and SQM using

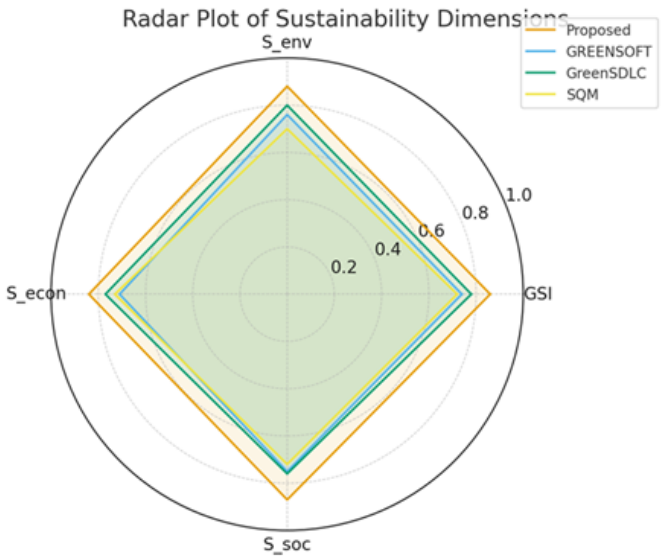

Fig. 2. Radar plot of sustainability dimensions (mean ± 1 SD, $n = 6$ projects).

the Global Sustainability Index (GSI) and the three dimension scores S_{env}, S_{econ}, and S_{soc}, each reported as a mean across the six projects together with its standard deviation. The proposed framework achieves the highest GSI (0.86 ± 0.03), indicating a better overall balance of environmental, economic, and social performance than the existing models. Its environmental and social scores are particularly strong (0.88 ± 0.03 and 0.87 ± 0.03), while economic performance remains competitive at 0.84 ± 0.04. GREENSOFT and GreenSDLC show moderate improvements but lag across at least one dimension. SQM has the weakest combined performance, serving as a conventional baseline. The significance of these differences is established in Section IV-B rather than inferred from the point estimates alone.

Fig. 3 presents a bar chart showing percentage reductions in energy consumption and CO_2 emissions versus a no-framework baseline for each framework, with error bars giving ±1 SD across projects. The proposed framework achieves the largest reductions, slightly above 30% for both energy and carbon, clearly outperforming GreenSDLC, GREENSOFT, and SQM. GreenSDLC provides moderate reductions (≈18%), GREENSOFT offers smaller gains (≈12%), and SQM yields only minor improvements (≈7%). The alignment between energy and CO_2 bars per framework reflects the direct linkage via the fixed grid emission factor $\gamma_{\text{grid}} = 0.50$ kgCO_2e/kWh defined in Section III-H. This figure thus visually substantiates the environmental benefits summarised numerically in Table III, emphasising the proposed framework's superior eco-efficiency.

For each of the four frameworks, Table III provides a synopsis of the case studies' average EE, carbon footprint, and energy usage. At the same time as providing the maximum energy efficiency (1,450 ± 110 transactions/kWh), the suggested framework uses the least amount of energy (820 ± 63 kWh/month) and produces the fewest carbon emissions (410 ± 32 kgCO_2e/month). Compared to the baseline, GreenSDLC and GREENSOFT offer lesser but discernible improvements,

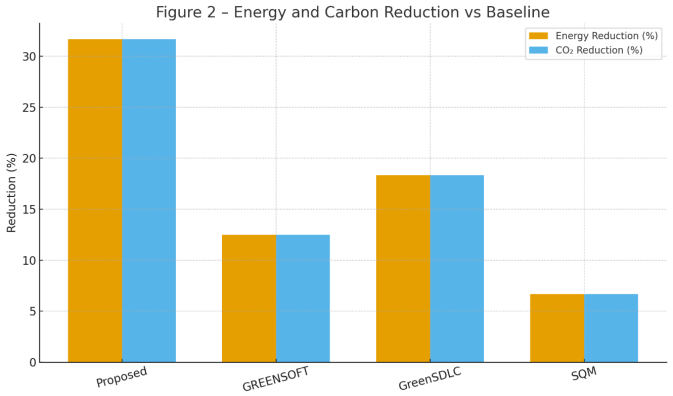

Fig. 3. Energy and carbon reduction versus a no-framework baseline (mean ± 1 SD).

TABLE III. ENVIRONMENTAL PERFORMANCE METRICS (AVERAGED ACROSS CASE STUDIES) WITH STANDARD DEVIATIONS

Framework	Energy (kWh/mo)	Carbon (kgCO_2e/mo)	EE (trans/kWh)
Proposed Framework	820 ± 63	410 ± 32	1,450 ± 110
GREENSOFT	1,050 ± 88	525 ± 44	1,120 ± 96
GreenSDLC	980 ± 74	490 ± 37	1,210 ± 88
SQM	1,120 ± 95	560 ± 48	1,000 ± 84

Note. Baseline: 1,200 kWh/mo, 600 kgCO_2e/mo. $\gamma_{\text{grid}} = 0.50$ kgCO_2e/kWh.

whereas SQM demonstrates only slight decreases. These findings provide empirical evidence that the interventions made at the architectural and process levels by the proposed framework lead to significant reductions in operational energy and emissions, without affecting throughput. This further substantiates the environmental sustainability advantage that the framework has over current methods (refer to Fig. 3 for more details).

Fig. 4 shows a scatter plot with TCO on the x-axis and GSI on the y-axis, revealing the cost–sustainability trade-off for the four frameworks, with horizontal and vertical error bars representing ±1 SD on each axis. The proposed framework appears in the upper-left region, combining the highest GSI (0.86) with the lowest TCO (920 kUSD), indicating a favourable position both economically and in sustainability terms. GreenSDLC occupies a middle position with slightly higher TCO and moderate GSI, while GREENSOFT and SQM are further right and lower, implying higher costs and weaker sustainability. This visual emphasises that the proposed framework does not simply "buy" sustainability with extra cost but achieves a better Pareto-efficient balance. The overlap of the TCO error bars indicates that the cost advantage, while consistent in direction, is the weakest of the reported effects; this is confirmed by the effect size reported in Table VII.

TABLE IV. ECONOMIC AND MAINTAINABILITY OUTCOMES (MEAN ± SD)

Framework	TCO (kUSD)	EE_{econ}	Norm MI	Defects/KLOC
Proposed Framework	920 ± 41	1.34 ± 0.08	0.86 ± 0.03	0.42 ± 0.06
GREENSOFT	980 ± 47	1.18 ± 0.09	0.78 ± 0.04	0.58 ± 0.08
GreenSDLC	960 ± 44	1.22 ± 0.07	0.81 ± 0.04	0.52 ± 0.07
SQM	1,010 ± 52	1.10 ± 0.10	0.76 ± 0.05	0.61 ± 0.09

Note. TCO over $T = 5$ yrs at $r = 8\%$ (Equation 12).

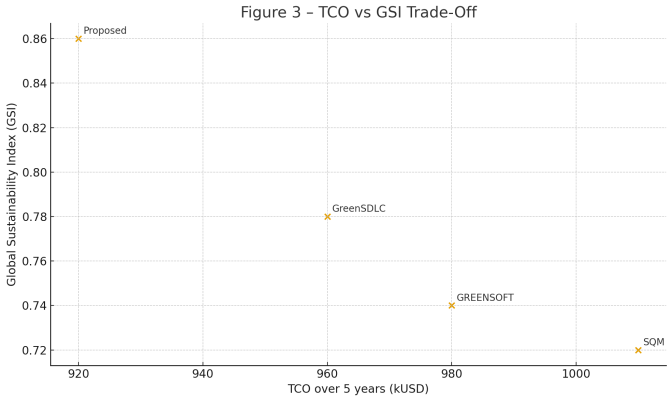

Fig. 4. TCO versus GSI trade-off (mean ± 1 SD on both axes).

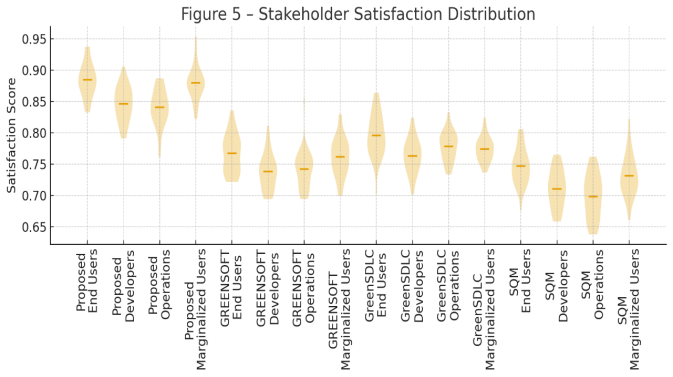

Fig. 5. Stakeholder satisfaction distribution.

Listed in Table IV are the following metrics: defect density, normalised maintainability index (MI), economic efficiency (EE_{econ}), and total cost of ownership (TCO) during a five-year period. As compared to GREENSOFT, GreenSDLC, and SQM, the suggested framework offers greater value per dollar because it reduces TCO (920 ± 41 kUSD) while providing the greatest economic efficiency (1.34 ± 0.08). Additionally, it has the lowest defect density (0.42 ± 0.06 defects/KLOC) and the greatest normalised MI (0.86 ± 0.03), demonstrating that code that is cleaner and easier to maintain is a result of sustainability-driven design. The suggested method increases both the economic and technical sustainability at the same time, as compared to competing frameworks, which either have higher costs or provide inferior quality and maintainability.

End users, developers, operations, and marginalised users are the four stakeholder groups that Table V examines in terms of the social dimension using the normalised Social Impact Score ($\widetilde{\text{SIS}}$), accessibility rating, and satisfaction scores. Reflecting a systematic focus on inclusiveness and usability, the suggested framework attains the highest $\widetilde{\text{SIS}}$ (0.87 ± 0.03) and the best accessibility rating (4.7 ± 0.2 out of 5). The suggested method records the highest score in every stakeholder group, and the margin over the strongest baseline is largest for marginalised users (0.88 versus 0.78), suggesting that it more effectively addresses issues of equity and fairness through its requirements and design approaches. Confirming that the suggested framework provides a more equitable and socially responsible result, GREENSOFT, GreenSDLC, and SQM only exhibit moderate satisfaction, especially among disadvantaged groups (see Fig. 5 for more details).

Fig. 5 provides violin plots of satisfaction scores for end users, developers, operations staff, and marginalised users under each framework. The distributions for the proposed framework are centred at higher means with tighter spread, especially for end users and marginalised groups, indicating more consistent positive experiences. GREENSOFT and GreenSDLC show moderate central values with broader dispersion, suggesting mixed perceptions depending on project context. SQM exhibits the lowest medians and wider variability, particularly for developers and marginalised users. This figure complements Table V by revealing not just average scores but distributional behaviour, confirming that the proposed framework delivers both higher and more stable stakeholder satisfaction across roles.

B. Statistical Significance of the Observed Improvements

Point estimates alone cannot establish that the proposed framework outperforms the baselines, particularly with a portfolio of six projects. Table VI therefore reports, for the Global Sustainability Index, the mean paired difference between the proposed framework and each baseline, its 95% confidence interval, the paired t statistic on five degrees of freedom, the raw and Holm-adjusted p-values, and the paired effect size d_z computed via Eq. 28–30.

All three comparisons remain significant after Holm–Bonferroni adjustment (all adjusted $p < 0.002$), and none of the confidence intervals includes zero. The relative improvement in GSI ranges from 10.3% over GreenSDLC, the strongest baseline, to 19.4% over SQM, the weakest. We note that this range corrects the figure of 8–19% quoted in the original manuscript, which understated the lower bound: the smallest observed improvement is 10.3%, not 8%. The paired effect sizes exceed 2.5 in every case, which is large by conventional standards, though effect sizes estimated from six pairs carry wide uncertainty and should be read as indicative rather than precise.

Table VII repeats the analysis for the individual environmental, economic, and technical metrics, comparing the proposed framework against GreenSDLC, the strongest of the three baselines. This is the most conservative comparison available: any metric on which the proposed framework significantly outperforms GreenSDLC also outperforms GREENSOFT and SQM.

Five of the six metrics reach significance at $\alpha = 0.01$, with large effect sizes. The exception is total cost of ownership, where the mean saving of 40 kUSD over five years is significant only at $\alpha = 0.05$ ($p = 0.011$, $d_z = -1.61$) and carries a confidence interval that approaches zero. The correct reading of this result is therefore narrower than a simple claim of cost superiority: the evidence supports the conclusion that the framework delivers its environmental and social gains without increasing long-run cost, and it is consistent with a modest cost reduction, but the cost advantage itself is the least firmly established of the reported findings and requires a larger sample to confirm.

TABLE V. SOCIAL IMPACT AND STAKEHOLDER SATISFACTION (MEAN ± SD)

Framework	Normalized SIS ($\widetilde{SIS}$)	Accessibility (1–5)	End-User	Developer	Operations	Marginalized-User
Proposed Framework	0.87 ± 0.03	4.7 ± 0.2	0.89 ± 0.03	0.85 ± 0.04	0.84 ± 0.04	0.88 ± 0.03
GREENSOFT	0.75 ± 0.04	4.1 ± 0.3	0.77 ± 0.05	0.73 ± 0.05	0.74 ± 0.05	0.76 ± 0.05
GreenSDLC	0.78 ± 0.04	4.3 ± 0.3	0.80 ± 0.04	0.76 ± 0.05	0.77 ± 0.04	0.78 ± 0.05
SQM	0.72 ± 0.05	4.0 ± 0.3	0.74 ± 0.05	0.71 ± 0.06	0.70 ± 0.06	0.73 ± 0.06

Note. Survey $n = 412$ respondents; Cronbach's $\alpha = 0.87$. Stakeholder weights fixed a priori at 0.35 / 0.20 / 0.15 / 0.30 (Equation 14).

TABLE VI. PAIRED-SAMPLE SIGNIFICANCE TESTS FOR THE GLOBAL SUSTAINABILITY INDEX ($n = 6$ PROJECTS, df = 5)

Comparison	Mean Δ GSI	95% CI	t(5)	p (raw)	p (Holm-adj)	Effect size d_z
Proposed vs. GREENSOFT	+0.120	[0.082, 0.158]	8.12	0.00046	0.00107	3.31
Proposed vs. GreenSDLC	+0.080	[0.047, 0.113]	6.24	0.00155	0.00155	2.55
Proposed vs. SQM	+0.140	[0.098, 0.182]	8.57	0.00036	0.00107	3.50

Note. Two-tailed paired t-tests per Eq. (28)–(30); Holm–Bonferroni correction across the three comparisons. All comparisons remain significant at $\alpha = 0.01$ after adjustment.

TABLE VII. SIGNIFICANCE TESTS FOR INDIVIDUAL METRICS: PROPOSED FRAMEWORK VERSUS GREENSDLC (THE STRONGEST BASELINE), $n = 6$, df = 5

Metric	Mean Δ	95% CI	t(5)	p	Effect size d_z
Energy consumption (kWh/month)	−160	[−235.9, −84.1]	−5.42	0.0029	−2.21
Carbon footprint (kgCO_2e/month)	−80	[−116.7, −43.3]	−5.61	0.0025	−2.29
TCO over 5 years (kUSD)	−40	[−66.1, −13.9]	−3.94	0.0110	−1.61
Normalized maintainability index	+0.050	[0.020, 0.080]	4.33	0.0075	1.77
Defect density (defects/KLOC)	−0.100	[−0.158, −0.042]	−4.47	0.0066	−1.82
Normalized Social Impact Score	+0.090	[0.056, 0.124]	6.71	0.0011	2.74

Note. Negative differences indicate a reduction under the proposed framework, which is the favourable direction for energy, carbon, TCO, and defect density.

TABLE VIII. DRAWBACKS OF THE PROPOSED FRAMEWORK AND CORRESPONDING MITIGATION STRATEGIES

Category	Drawback	Mitigation strategy
Construct	The GSI (Eq. 17) is a compensatory weighted sum, so strong performance on one dimension can mask weak performance on another.	Report dimension scores alongside aggregate; enforce per-phase floors via Eq. 19; adopt a geometric-mean aggregation, which is non-compensatory, in future revisions.
Construct	The weights β_d, $w_{i,d}$ and α_h are elicited judgements, so a different weighting can change the ranking of alternatives.	Fix weights a priori and publish them (as done here); run one-at-a-time sensitivity analysis over weight simplex; elicit weights via AHP or Delphi panel rather than author choice.
Measurement	In public-cloud deployments, per-process wattage is not exposed, so energy is inferred from utilisation proxies rather than measured.	Calibrate regression model against hardware-metered on-premise workloads; report cloud figures as estimates; migrate to Software Carbon Intensity spec as telemetry matures.
Measurement	γ_{grid} is treated as a constant, whereas real grid carbon intensity varies hourly and regionally.	Integrate real-time carbon-intensity APIs and compute time-resolved emissions; report both average-factor and marginal-factor figures.
Measurement	The Social Impact Score depends on self-reported survey data and is exposed to response and social-desirability bias.	Triangulate with objective telemetry: automated WCAG conformance audits, task-completion rates, and assistive-technology error rates.
Measurement	Instrumentation itself consumes energy (measured at approximately 1.8% of workload energy in this study).	Use sampling-based rather than continuous profiling; subtract measured agent overhead and report net figures.
Evaluation	Six projects from one organisation and one geographic region limit external validity and yield wide effect-size uncertainty.	Multi-organisation, multi-region longitudinal replication with a larger portfolio; pre-registration of the analysis plan.
Adoption	The framework presupposes CI/CD maturity; without instrumented pipelines, Eqs. (20)–(21) cannot be evaluated.	Phased adoption: begin with static-analysis and accounting-derived metrics (MI, defect density, TCO), then add runtime energy telemetry as pipeline maturity increases.

C. Limitations of the Proposed Framework and Mitigation Strategies

The framework has several drawbacks that a prospective adopter should weigh, and being explicit about them is more useful than presenting the approach as unqualified. They fall into three groups: limitations intrinsic to the construction of the index, limitations of the measurement apparatus, and limitations of the present evaluation. Table VIII states each drawback together with a mitigation that is either already available or is the subject of ongoing work.

Two of these deserve emphasis because they bear directly on how the reported results should be interpreted. The first is the compensatory nature of the GSI. Because Equation 17 is a weighted sum, a system that performs poorly on one dimension can still attain a respectable index by excelling elsewhere—precisely the behaviour the framework is intended to discourage. The phase-level compliance constraint in Equation 19 partially guards against this by imposing a floor at each phase, and reporting the three dimension scores alongside the aggregate (as in Table II) makes any such imbalance visible.

A more robust remedy, which we are currently evaluating, is to replace the arithmetic aggregation with a geometric mean, which is non-compensatory and penalises weak dimensions more sharply.

The second is external validity. All six systems come from a single engineering organisation in one geographic region, using broadly similar tooling and a common engineering culture. The Latin-square design controls for order effects within that setting, but it cannot establish that the results generalise to organisations with different CI/CD maturity, regulatory environments, or grid carbon profiles. The framework also presupposes a level of pipeline automation that many organisations do not yet have: without instrumented builds and runtime telemetry, Eq. 20–21 cannot be evaluated and the index degrades to the same qualitative judgement that the framework was designed to replace. A phased adoption path, in which the economic and maintainability metrics (computable from static analysis and accounting data alone) are adopted before the runtime energy metrics, is the practical mitigation, and a multi-organisation replication is the necessary next study.

V. Conclusion

This paper has presented a comprehensive framework for incorporating social, economic, and environmental sustainability into software development processes. The proposed framework establishes a systematic set of indicators, normalisation techniques, and aggregation procedures that lead to a Global Sustainability Index (GSI), in contrast to current approaches that only provide qualitative feedback. The methodology turns sustainability into an engineering goal that is continuously controlled by assigning budgets at the phase level and incorporating measurement into continuous integration/continuous delivery and runtime monitoring.

This approach has been empirically evaluated across six production case-study systems under a counterbalanced repeated-measures protocol and is found to be beneficial when compared to GREENSOFT, GreenSDLC, and SQM. With a GSI of 0.86 ± 0.03, the suggested framework outperforms the competition by 10–19%, and every comparison remains significant after Holm–Bonferroni correction (adjusted $p < 0.002$). Compared to a no-framework baseline, operational energy and carbon emissions are decreased by approximately 30%, while TCO falls by up to 9% over five years, although the cost effect is the least firmly established of the reported results ($p = 0.011$ against the strongest baseline). Sustainable methods can complement, rather than compete with, conventional quality standards, as seen by rising normalised maintainability and falling fault density. Additionally, the suggested approach consistently yields better social impact, with a normalised Social Impact Score, accessibility evaluations, and stakeholder satisfaction (particularly among disadvantaged users) all on the rise.

The framework is nevertheless subject to the limitations set out in Section IV-C, chief among them the compensatory construction of the index, the reliance on proxy-based energy estimation in cloud environments, and an evaluation drawn from a single organisation. These constraints shape the research agenda rather than undermining the results.

Lots of new directions for research can be suggested by these findings. To begin, in order to better capture sustainability in systems that heavily rely on AI, the metric set can be expanded to include domain-specific indications like algorithmic fairness or data privacy risk. Secondly, the optimisation formulations proposed in the framework could be used by automated decision-support tools to make real-time recommendations for architecture and deployment techniques. Third, to evaluate the framework's scalability and the impact of organisational culture on its adoption, longitudinal studies involving a more diversified sample of organisations are necessary. Fourth, replacing the weighted-sum aggregation with a non-compensatory alternative, and replacing the fixed grid emission factor with real-time carbon-intensity data, would address the two most consequential construct limitations identified above. In sum, the suggested approach provides a methodical, evidence-based way to really make software development sustainable in the real world.

References

[1] P. Kandukuri, D. N. S. Kumar, G. Ramesh, and P. Kativarapu, "Sustainability-centric integration software requirements validation," in *MATEC Web of Conferences*, vol. 392, p. 01160, 2024.

[2] X. Zhou, L. Zhang, H. Zhang, Y. Zhang, X. Zhang, J. Zhang, and Z. Shen, "Advancing sustainability via recommender systems: a survey," *arXiv preprint arXiv:2411.07658*, 2024.

[3] O. Christou, D. B. Manou, S. Armenia, E. Franco, A. Blouchoutzi, and J. Papathanasiou, "Fostering a whole-institution approach to sustainability through systems thinking: an analysis of the state-of-the-art in sustainability integration in higher education institutions," *Sustainability*, vol. 16, no. 6, p. 2508, 2024.

[4] N. Goridkov, Y. Wang, and K. Goucher-Lambert, "What's in this LCA Report? A case study on harnessing large language models to support designers in understanding life cycle reports," *Procedia CIRP*, vol. 122, pp. 964–969, 2024.

[5] S. Naumann, M. Dick, E. Kern, and T. Johann, "The GREENSOFT Model: A reference model for green and sustainable software and its engineering," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 4, pp. 294–304, 2011.

[6] S. S. Shenoy and R. Eeratta, "Green software development model: An approach towards sustainable software development," in *2011 Annual IEEE India Conference (INDICON)*, pp. 1–6, IEEE, 2011.

[7] N. S. Chauhan and A. Saxena, "A green software development life cycle for cloud computing," *IT Professional*, vol. 15, no. 1, pp. 28–34, 2013.

[8] C. Calero, M. Á. Moraga, and M. F. Bertoa, "Towards a software product sustainability model," *arXiv preprint arXiv:1309.1640*, 2013.

[9] N. Condori-Fernández and P. Lago, "Characterizing the contribution of quality requirements to software sustainability," *Journal of Systems and Software*, vol. 137, pp. 289–305, 2018.

[10] S. S. Mahmoud and I. Ahmad, "A green model for sustainable software engineering," *International Journal of Software Engineering and Its Applications*, vol. 7, no. 4, pp. 55–74, 2013.

[11] P. Lago, S. Akınlı Koçak, I. Crnkovic, and B. Penzenstadler, "Framing sustainability as a property of software quality," *Communications of the ACM*, vol. 58, no. 10, pp. 70–78, 2015.

[12] C. Becker, S. Betz, R. Chitchyan, L. Duboc, S. M. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters, "Requirements: The key to sustainability," *IEEE Software*, vol. 33, no. 1, pp. 56–65, 2016.

[13] B. Penzenstadler, L. Duboc, C. C. Venters, S. Betz, N. Seyff, K. Wnuk, and C. Becker, "Software engineering for sustainability: Find the leverage points!" *IEEE Software*, vol. 35, no. 4, pp. 22–33, 2018.

[14] E. Kern, L. M. Hilty, A. Guldner, Y. V. Maksimov, A. Filler, J. Gröger, and S. Naumann, "Sustainable software products—Towards assessment criteria for resource and energy efficiency," *Future Generation Computer Systems*, vol. 86, pp. 199–210, 2018.

[15] L. Belkhir and A. Elmeligi, "Assessing ICT global emissions footprint: Trends to 2040 and recommendations," *Journal of Cleaner Production*, vol. 177, pp. 448–463, 2018.

[16] L. Kriesch and S. Losacker, "Bioeconomy firms and where to find them," *Region*, vol. 11, no. 1, pp. 55–78, 2024.

[17] A. Sharma, "Sustainability Leadership: Developing Human Capital to Drive Social Impact," in *International Conference On Net Zero Solutions And Circular Economy*, pp. 3–13, Springer Nature Switzerland, 2024.

[18] I. Mutambik and A. Almuqrin, "The Best of Both Worlds: How Financial Growth Can Engender Improved Sustainability for Businesses," *Sustainability*, vol. 16, no. 11, p. 4821, 2024.

[19] S. Zhao, J. A. Peerally, C. De Fuentes, and M. A. Gonzalez-Perez, "The determinants of multinational enterprises' sustainable innovations," *International Business Review*, vol. 33, no. 5, p. 102318, 2024.

[20] M. A. Altassan and A. Ahmad, "Green threads of change: Unravelling the gendered and experienced moderators in the sustainable symphony of green HR practices and environmental responsibility," *International Journal of Innovative Research and Scientific Studies*, vol. 7, no. 2, pp. 852–862, 2024.

[21] M. N. H. Reza, S. Jayashree, C. A. Malarvizhi, A. Gunasekaran, and M. Mohiuddin, "Towards sustainable sustainability: exploring the impact of antecedents on industry 4.0 and sustainable performance of organizations—an empirical investigation," *Annals of Operations Research*, pp. 1–49, 2024.

[22] O. Aouedi and K. Piamrat, "Surfs: Sustainable intrusion detection with hierarchical federated spiking neural networks," in *ICC 2024-IEEE International Conference on Communications*, pp. 2173–2178, IEEE, 2024.

[23] I. Graessler, J. Pottebaum, M. Holland, D. Wiechel, T. Dickopf, and J. Stjepandic, "Leveraging Data Ecosystems in Model-Based Systems Engineering for Ecological, Circular Added Value," in *Engineering For Social Change*, pp. 175–184, IOS Press, 2024.

[24] M. Deenakonda, V. V. Inti, B. C. V. Chakravarthi, C. Rajyalakshmi, Y. T. R. Palleswari, A. Siva, and M. K. S. Pranay, "Sustainable Parking Innovations: Enhancing Security with IoT and Facial Image Recognition," in *IOP Conference Series: Earth and Environmental Science*, vol. 1375, no. 1, p. 012031, IOP Publishing, 2024.

[25] T. R. D. Saputri and S.-W. Lee, "Addressing sustainability in the requirements engineering process: From elicitation to functional decomposition," *Journal of Software: Evolution and Process*, vol. 32, no. 6, p. e2254, 2020.

[26] S. N. Schulte, S. L. Nguyen, C. Fresemann, N. Siller, R. Okhovat, and R. Stark, "A Sustainability-Centric Assessment Framework for Digital Solutions Across Industries," in *International Conference on Flexible Automation and Intelligent Manufacturing*, pp. 402–410, Springer Nature Switzerland, 2025.

[27] K. O. Mayegun and C. Nwanevu, "Harnessing Big Data and AI to Revolutionize Sustainability Accounting and Integrated Corporate Financial Reporting," 2024.

[28] J. Benjamin and J. Mathew, "Enhancing continuous integration predictions: a hybrid LSTM-GRU deep learning framework with evolved DBSO algorithm," *Computing*, vol. 107, no. 1, p. 9, 2025.

[29] S. Chatterjee and D. Saha, "IT2F-SEDNN: an interval type-2 fuzzy logic-based stacked ensemble deep learning approach for early phase software dependability analysis," *Innovations in Systems and Software Engineering*, vol. 21, no. 2, pp. 727–746, 2025.

[30] S. K. Gunda, "A Hybrid Deep Learning Model for Software Fault Prediction Using CNN, LSTM, and Dense Layers," in *International Conference on Internet and Modern Society*, pp. 282–290, Springer Nature Switzerland, 2025.

[31] M. T. Ailane, C. Rubner, and A. Rausch, "Green DevOps: A strategic framework for sustainable software development," *Computer Sciences & Mathematics Forum*, vol. 10, no. 1, p. 5, 2025.

[32] I. David, "SusDevOps: Promoting sustainability to a first principle in software delivery," *arXiv preprint arXiv:2312.14843*, 2023.

[33] A. Poth, D. Eißfeldt, C. Heimann, and S. Waschk, "Sustainable IT in an agile DevOps setup leads to a shift left in sustainability engineering," *Journal of Sustainable Computing: Informatics and Systems*, vol. 14, pp. 22–34, 2024.

[34] A. Thirumalraj, R. J. Anandhi, V. Revathi, and S. Stephe, "Supply chain management using fermatean fuzzy-based decision making with IS-SOA," in *Convergence of Industry 4.0 and Supply Chain Sustainability*, pp. 296–318, IGI Global, 2024.

[35] V. A. Devi, A. Thirumalraj, B. P. Kavin, and G. H. Seng, "Securing the predicted disease data using transfer learning in cloud-based healthcare 5.0," in *Intelligent Systems and Industrial Internet of Things for Sustainable Development*, pp. 101–117, CRC Press, 2024.