Integrating Sustainability Metrics into Project and Portfolio Performance Assessment in Agile Software Development: A Data-Driven Scoring Model

Cristian Fagarasan 1,*†, Ciprian Cristea 2, Maria Cristea 2, Ovidiu Popa 1 and Adrian Pisla 1,∗†

1 Design Engineering and Robotics Department, Machine Building Faculty, Technical University of Cluj-Napoca, 103-105 Muncii Avenue, 400641 Cluj-Napoca, Romania; popaovidiu92@yahoo.com
2 Electrical Machines and Drives Department, Faculty of Electrical Engineering, Technical University of Cluj-Napoca, 26-28 G. Baritiu Street, 400027 Cluj-Napoca, Romania; ciprian.cristea@emd.utcluj.ro (C.C.); maria.cristea@enm.utcluj.ro (M.C.)
* Correspondence: office.cristif@gmail.com (C.F.); adrian.pisla@muri.utcluj.ro (A.P.); Tel.: +40-745539634 (C.F.)

Abstract: In today’s rapidly evolving information technology sectors, agile methodologies have been employed by software development organizations to facilitate the large-scale, efficient, and swift development of digital products. Notably, it is a widely accepted principle that an increase in project delivery predictability results from more effective portfolio management. Despite the abundant resources within software engineering that address project management and agile development performance, the measurement of portfolio delivery performance integrating sustainability principles is under-researched. This paper aims to underline a solution to bridge this gap by proposing a data-driven scoring model explicitly designed for software firms to integrate sustainability metrics into their project and portfolio performance assessment. The model is primarily aimed at monitoring and enhancing delivery performance while also reinforcing the sustainability of the software development lifecycle. A thorough literature review was conducted to discern gaps in existing practices, followed by development of a scoring model melded with delivery and sustainability metrics seamlessly. Validated through a case study, the findings reveal that the model influences the performance and sustainability dynamics within software development entities. The insights gained from this study underscore the pivotal role of a harmonized delivery and sustainability metrics system in enhancing the sustainability and efficiency of software development undertakings.

Keywords: software development; project management; agile; lean; portfolio performance; data-driven decision making; sustainability metrics; software development lifecycle

1. Introduction

Project and portfolio management have long been acknowledged as effective strategies for driving strategic transformations within organizations [1]. While there was once confusion surrounding these structures, a clear distinction now exists between these two methods of organizing project-based work [2]. As organizations increasingly adopt these temporary structures to deliver value, the literature on project and portfolio management has expanded. It now delves into topics such as resource allocation, interdependencies among portfolio components, and the creation of governance and decision-making models [3]. Project-oriented organizations often manage multiple projects concurrently, which results in a complex portfolio that becomes increasingly difficult to manage as the number of projects increases. These organizations must navigate dynamic boundaries and contexts, maintain relationships with diverse social environments, and perform integrative functions such as project portfolio management, which encompasses all projects an organization holds at a given time and their interrelationships [4].
In an era where software is increasingly vital in driving business processes, software project management has emerged as a critical discipline for addressing the unique challenges associated with developing and implementing software solutions [5]. Managing project portfolios can differ significantly, as it is directly influenced by the unique environmental conditions in which a project is managed, necessitating tailored approaches to address specific organizational contexts [6]. Consequently, developing effective management processes and methods for project-oriented organizations is essential for achieving successful outcomes. Measuring software delivery performance is inherently challenging, partly due to the intangible nature of inventory and the simultaneous design and delivery activities, especially in agile software development methodologies [7]. As designs evolve based on implementation insights, establishing a valid, reliable metric for software delivery performance assessment becomes crucial. An effective software delivery performance metric should have two key characteristics. First, it must emphasize a global outcome to prevent competition between teams. For example, rewarding developers for throughput and operations for stability can lead to a “wall of confusion” where poor quality code is handed over to operations, resulting in cumbersome change management processes. Secondly, the metric should prioritize outcomes over output, ensuring organizational goals are met without promoting excessive busy work. In high-performing technology organizations, four key metrics have been identified for measuring software delivery performance: delivery lead and cycle time, deployment frequency, mean time to restore service, and change fail rate [8].

The notion of sustainability is quickly gaining traction in the business sector and predicted to be a leading trend in the upcoming years. This focus on sustainability is not limited to its social and political impacts. It also influences market operations and customer needs and, as a result, shapes entrepreneurial strategies [9]. As stated [10], sustainability emphasizes meeting current needs without compromising the ability of future generations to satisfy their requirements. It consists of three elements: social, economic, and environmental factors. Implementing these principles within the information technology sector has led to the emergence of a new discipline, green software engineering, which focuses on writing energy-efficient software. In practical terms, even though the hardware is ultimately responsible for energy consumption for any programmable device, the software determines how energy is used and managed [11]. While software may appear intangible and irrelevant to sustainability, it significantly impacts resource usage. Efficient software consumes less power, requires less hardware, and can intelligently control behavior to reduce energy consumption [12]. When exploring the concept of sustainability within the realm of software and software engineering, it is crucial to analyze the matter from two distinct perspectives: the sustainability of the software itself and the utilization of software to achieve sustainability [13]. The former, referred to as sustainable software, is characterized by its durability, maintainability, and cost-efficiency. In essence, sustainable software entails developing, deploying, and using software that minimizes negative impacts or even generates a positive influence, directly or indirectly, on the economy, society, human beings, and the environment [14].

Despite the growing recognition of sustainability’s importance within software engineering and its potential impact on society and the environment, there is still a gap in the literature concerning integrating sustainability metrics into project and portfolio performance assessment within agile software development contexts. Acknowledging this deficiency, the present study aims to bridge this gap by introducing a data-driven scoring model that effectively combines project management principles with sustainability considerations. The proposed model enables the measurement of the sustainability of code, thereby fostering a more comprehensive understanding of the relationship between project delivery performance and sustainable practices. This approach highlights the significance of promoting long-term sustainability throughout the software development process while simultaneously addressing potential challenges associated with project delivery.
The proposed key performance indicators (KPIs) not only facilitate a better understanding of the frequency and velocity of value delivery but also emphasize the importance of sustainable practices in software development.

2. Materials and Methods

The research questions that this paper seeks to address encompass several key areas of project and portfolio performance management. In agile software development, organizations grapple with the challenge of incorporating sustainability metrics into their delivery methods for a comprehensive assessment of the project portfolio’s health. Additionally, there is a quest to identify which indicators yield the most insightful and actionable understandings of project delivery performance, particularly for sustainability. This research also explores how the introduced data-driven scoring model streamlines the incorporation of sustainability metrics while simultaneously amplifying project and portfolio performance evaluation in software development practices.

Recognizing the multifaceted challenges of integrating sustainability into project portfolio performance management, an integrative methodology was employed, juxtaposing a review of the existing literature with the development and validation of a novel model. The main goal was to enhance the project and portfolio performance assessment by seamlessly embedding sustainability metrics in agile software environments.

The framework of the methodology adopted in this research for enhancing project and portfolio performance assessment by embedding sustainability metrics in agile software environments is showcased in Figure 1.

This exploration commences with a deep dive into the existing literature and an evaluation of practices in software development performance and sustainability. Subsequent steps entail understanding the specific context of the organizational environment, defining the lifecycle and corresponding performance metrics, and pinpointing code KPIs influencing sustainability. This foundation led to the formulation of an integrated model that intertwines delivery performance with sustainability metrics. The model’s robustness and applicability are then put to the test through real-world case studies, with results visualized for optimal interpretation.

The Project Management Institute’s Project Management Book of Knowledge outlines the five phases of project management: initiating, planning, executing, monitoring, and closing [15]. Ideas are transformed into capabilities and, later, outcomes through these stages. The software development industry presents specific variations in the application of this methodology, characterized by the organizational structure of projectized or product-oriented approaches [16]. The former involves organizing teams around individual projects, with the disbandment of teams once the project is completed. In contrast, the latter emphasizes the development of company products, with ongoing work by product development teams beyond the completion of each project. In software development companies, portfolios comprise all software projects and may be affected by factors such as the organizational
The understanding of effective project portfolio management (PPM) is still limited, but existing research methods reveal that PPM effectiveness can be measured by taking into consideration several strategic attributes, like strategic alignment, adaptability, and expected value, but also some operational attributes, which include, but are not limited to, visibility, transparency in decision making, and predictability in project delivery. It has been proven that the greater the level of predictability in project delivery is, the more effective the management of the project portfolio [18]. This paper focuses on the operational attributes of a portfolio, specifically on project and portfolio delivery, considering factors that would enhance the sustainability of agile software delivery.

In the software development industry context, it is common practice for companies to adopt frameworks and methodologies to articulate their objectives and outline the steps to achieve them. The organizational strategy formulation serves as the guiding principle for the composition of the portfolio elements. The Objectives and Key Results (OKRs) framework, a widely adopted leadership tool for goal setting and progress tracking, has proven effective among leading technology companies.

This concept was first developed at Intel and soon gained popularity among other Silicon Valley companies. In 1999, Google adopted this goal-setting system, which played a vital role in the company’s impressive growth [19]. According to a study run in October 2022, Google searches for the OKRs theme grew by approximately 500% in the last six years, and the software management solutions centered on OKRs currently stand at USD 1.5 billion, which shows a significant investment in platforms that assist organizations in implementing this framework [20]. The structure of a company’s portfolio is composed of multiple layers, with each layer corresponding to a specific OKR level, as seen in Figure 2.

![Figure 2. Connecting the portfolio and project levels with OKR levels.](image)

The company portfolio provides a holistic representation of all the projects undertaken by an organization. Within this portfolio, every project is seen as contributing to the realization of the organization’s vision and strategic goals. When the term “portfolio” is used, all projects initiated within the company and specific functions or sub-functions are encompassed. Whether long-term undertakings or short-term tasks, these projects align with the company’s objectives and are instrumental in their achievement.

The objectives represent the organization’s desired outcomes, while the key results define how they will be accomplished. Objectives should be inspiring, specific, and actionable by the organization’s team and can be set at annual or quarterly intervals. On the other hand, key results must be verifiable, measurable, and set quarterly, ensuring that each project contributes to achieving the company’s overarching goals and strategy.

This structure was the basis of the proposed data-driven model for evaluating the health of all portfolio levels, leveraging project management tools commonly used in the information technology sector for software development. The aim is to provide real-time,
accurate delivery performance data that cater to the needs of all stakeholders, as outlined in Figure 3, thereby satisfying the requirement for effective delivery performance tracking.

Figure 3. Connecting the portfolio and project levels with stakeholder levels.

As delivery excellence is crucial to attaining the set OKRs, organizations must emphasize the importance of implementing proper metrics and ensuring transparency in their portfolio governance. Systematic and comprehensive data collection and easily interpretable dashboards can provide valuable insights into project and portfolio delivery performance. The executing phase is the longest and toughest stage of the software development lifecycle. Even if projects are delivered using agile methods, proper definition of metrics is vital for success [21].

Defining and implementing delivery KPIs is crucial for reflecting the health of each temporary structure within the organization. Moreover, identifying deviations is crucial for effective portfolio governance, with transparency and formalization being key to achieving data quality. Data collection must be systematic and comprehensive, starting from the lowest level of project management and aggregated into easily interpretable upper-organizational-level dashboards to provide accurate and meaningful insights into portfolio delivery performance. While the reporting systems are transparent, interpreting, analyzing, and converting this information into actionable insights truly matter [22]. By effectively interpreting, analyzing, and converting this information into actionable insights, function leaders and high-ranking executives can make informed decisions, remove roadblocks, and ultimately contribute to successfully realizing their organization’s OKRs.

The attainment of company objectives hinges on the successful realization of specified key results, driving the initiation of projects. Ensuring projects are delivered promptly is pivotal for a company to meet its goals. Therefore, it is vital to identify and track metrics that provide a clear perspective on project and portfolio performance. A model is here introduced that evaluates project delivery and overall portfolio performance. Furthermore, by incorporating sustainability measures, a comprehensive score for the portfolio can be derived.

2.1. Project Delivery Performance

The project lifecycle, traditionally consisting of five phases, was tailored to the information technology industry, resulting in a seven-phase software development lifecycle that comprises: analysis, planning, design, implementation, testing, deployment, and maintenance. This paper tracks projects through five stages: discovery, planning, development, evaluation, and closing. Each stage involves specific inputs, tasks to be carried out, and outputs that pave the way for subsequent phases.

Organizations with mature software delivery processes are more likely to complete projects on schedule: such teams deliver 63% of their projects on time, while less mature teams deliver only 39% [23]. A project management information system (PMIS) is essential for monitoring all organizational projects, as it allows for data collection and analysis. Utilizing a PMIS is critical for cost-effective and successful project execution. An important feature of such software applications is the automated collection and reporting of key
performance indicators, which provide essential input for project monitoring. Market-available project management systems offer standard reports that project teams can use to track ongoing trends and make necessary adjustments.

A systematic approach to assessing the delivery health (DH) of an organization’s software portfolio is necessary, as isolated tracking of project delivery performance (PDP) is insufficient [24]. To extract meaningful insights from the PMIS, gathering and analyzing relevant metrics is crucial, which requires collecting input data directly from project metadata.

2.1.1. Delivery Health

The project health metric is a valuable tool for assessing a project’s progress and overall success. This metric is determined by comparing the initial deadline established with stakeholders during the planning phase to the projected delivery date the software development team agreed upon during the execution phase. Throughout the project initiation phase, stakeholders collaborate to identify realistic delivery dates for the proposed project. As the project transitions into the planning stage, software development teams gain access to a wealth of information that allows them to strategize more effectively. According to [25], there are four key stages in the agile planning lifecycle: preliminary planning, planning, release planning, and iteration planning. Software development teams can offer increasingly precise estimates as project deliverables move through these stages. This continuous reevaluation of the projected delivery date ensures stakeholders remain well-informed and expectations are managed appropriately. The agile planning approach emphasizes flexibility and adaptability, allowing for ongoing adjustments as needed, compensating for the uncertainty of the business environment [26]. The schedule variance, or the difference between the initially agreed-upon deadline and the updated projected delivery date, offers valuable insight into a project’s actual progress compared to its expected progress [27].

In the case of delivery health, the delivery date agreed upon with stakeholders precedes the projected delivery date agreed upon with the product development team, and the project is considered delayed until corrective measures are implemented. This information can be used to refine future project planning and guarantee successful project delivery by setting clear expectations and creating more accurate timelines. It can also facilitate collaboration and communication among team members and stakeholders by serving as a common language that can be used to discuss the project’s status and potential challenges, fostering an environment of transparency and mutual understanding. Furthermore, the insights gained from schedule variance analysis can assist project managers in effective risk management by implementing mitigation strategies to minimize project delays and ensure a successful outcome.

The utility of the project health metric can be enhanced by introducing a delivery score model that solely focuses on the days’ difference between the initial deadline and the projected delivery date. This model employs a sigmoid function to convert the days’ difference into a score that ranges from 0 to 1, with 0 signifying a considerable delay and 1 representing outstanding progress. The sigmoid function is shown in Equation (1).

\[
f(x) = \frac{1}{1 + e^x}
\]  

where \( f(x) \) is the sigmoid function; \( e \) is the base of the natural logarithm, approximately equal to 2.71828; and \( x \) is the input to the function, which can be any real-valued number. In the context of a sigmoid function, \( x \) is transformed such that the output of the function is bounded between 0 and 1, providing a smooth, s-shaped curve. To tailor this function to the current scoring model, it is essential to modify the input variable \( x \) in accordance with the current delay. The variable \( x \) will be defined as follows:

\[
x = k \cdot (CD - MAD)
\]
where CD is the current delay; MAD is the maximum accepted delay, which can be calculated as a percentage of the project lifecycle to date; k is the steepness parameter. Based on the formulas above, the delivery health score can then be calculated as follows:

$$DH = \frac{1}{(1 + e^{k(CD - MAD)})}$$ (3)

where DH is the delivery health score. If CD is 0, then DH will be considered equal to 1. The steepness parameter introduces more flexibility and control over calculating the DH score. It provides a more accurate representation of project health based on the nature of project deadlines, where small delays may not significantly impact the overall project health. In contrast, larger ones can have a more pronounced effect. The steepness parameter adjusts the score’s sensitivity to the days’ difference variations. In other words, the k parameter helps adjust the steepness of the sigmoid function [28]. The steepness factor can be divided into three main aspects: selection, impact, and practical implications:

1. Selection:
   Choosing an appropriate steepness factor depends on the specific project context and the desired level of sensitivity for the delivery health score. An optimal steepness factor can be determined by analyzing historical project data and expert opinions to ensure that the DH score accurately reflects project performance. To select a suitable parameter value, project managers should consider factors such as the project’s complexity, expected variability in delivery dates, and the desired responsiveness of the DH score to changes in the days’ difference;

2. Impact:
   The steepness parameter impacts the shape of the sigmoid curve, which in turn affects the DH score. The choice of steepness factor directly impacts how sensitive the DH score is to changes in project performance. A smaller steepness factor results in a steeper sigmoid curve, meaning that the DH score will change more dramatically in response to small changes in the days’ difference. Conversely, a larger steepness factor will produce a flatter sigmoid curve, leading to a more gradual change in the DH score as the days’ difference varies;

3. Practical implications:
   In practice, the steepness factor should be chosen to provide meaningful and actionable insights to project managers and stakeholders. If the steepness factor is too small, the DH score may be overly sensitive, causing it to fluctuate dramatically with minor changes in the days’ difference, making it difficult to interpret and potentially leading to hasty or unnecessary actions. On the other hand, if the steepness factor is too large, the DH score may not be responsive enough to indicate important shifts in project performance, potentially masking underlying issues that require attention.

2.1.2. Lifecycle Time

In today’s fast-paced technological landscape, measuring the software development lifecycle (SDLC) time for projects is crucial for optimizing efficiency, resource allocation, and overall project success. Numerous SDLC models have evolved alongside the methodologies used in software development [29,30]. As an essential metric for portfolio delivery, the assessment of the time required for a project to traverse its entire lifecycle, from inception to realization, offers vital insights into the performance of the software development process. The lifecycle time (LT) metric sheds light on the efficiency and rapidity of the employed model, allowing organizational leaders to identify bottlenecks, overhead costs, and areas for improvement. Streamlining project delivery becomes more achievable with this understanding. Furthermore, documenting the information gathered from each project creates a valuable addition to the organization’s lessons-learned repository for future reference.
The project lifecycle KPI reflects the number of calendar days between the initiation of
the discovery phase and the transition to the evaluation stage. To calculate this metric, two
critical time points must be captured: the start date of the discovery phase and the date
when the project was transitioned to the evaluation column. A sigmoid function can be
used to evaluate the efficiency of a project’s lifecycle time by comparing it with the average
lifecycle time of analogous projects. This approach allows us to derive a lifecycle time score.

\[
LT = \frac{1}{1 + e^{k \cdot (CLT - ALT)}}
\]  

(4)

where LT is the lifecycle time; CLT is the current lifecycle time; ALT is the average lifecycle
time. The k parameter adjusts the score’s sensitivity to variations in lifecycle time. CLT
refers to the lifecycle time of the project under consideration, while ALT represents the
average lifecycle time of similar projects within the organization. The LT score ranges from
0 to 1, with 0 signifying poor efficiency and 1 indicating outstanding efficiency.

2.1.3. Phase-Based Tracking

Drawing inspiration from lean manufacturing, the phase-based tracking (PT) metric
indicates the time elapsed from when the work commences to when it is accepted [31]. As
part of the model proposed in this paper, the PT metric focuses on the time spent in the
planning and development phases of the SDLC, deliberately excluding the time dedicated
to the discovery, evaluation, and closing phases. Comparing the results of this metric to the
overall duration of the project lifecycle enables conclusions regarding the proportion of time
allocated to the discovery, evaluation, and closing phases compared to the time invested in
planning and development. Two data points must be captured to measure this metric: the
date the project was transitioned into planning and the date it was moved into evaluation.
The difference in days between these dates represents the project’s phased-based time,
providing insight into the efficiency and effectiveness of project execution processes.

This analysis illuminates the relative duration of different segments within the soft-
ware development lifecycle, identifying areas where process improvements are warranted.
Measuring the time spent in the planning and development stages provides insights into
the team’s ability to achieve the desired outcomes. The PT score can be calculated simi-
larly to the LT score by utilizing a sigmoid function to assess the efficiency of a project’s
phase-based tracking compared to the average phase-based tracking of similar projects:

\[
PT = \frac{1}{1 + e^{k \cdot (CPT - APT)}}
\]

(5)

where CPT is the current phase-based tracking and APT is the average phase-based tracking.
CPT denotes the phase-based tracking of the project under consideration, while APT
signifies the average phase-based tracking of similar projects within the organization. The k parameter adjusts the score’s sensitivity to variations in phase-based tracking. Ranging
from 0 to 1, the PT score signifies poor efficiency with a score of 0 and outstanding efficiency
with a score of 1.

2.1.4. Project delivery Performance Score

The PDP score provides a holistic assessment of a project’s progress, efficiency, and
overall success. To compute the PDP score, a weighted average of the three individual
scores is used, with weights assigned based on their relative importance in the specific
context:

\[
PDP = (w1 \cdot DH) + (w2 \cdot LT) + (w3 \cdot PT)
\]

(6)

where w1, w2, and w3 are the weights assigned to the DH, LT, and PT scores, respectively.
The sum of the weights should equal 1. These weights can be determined based on the
organization’s priorities, industry standards, or expert judgment. The PDP score will also
range from 0 to 1, with 0 indicating poor project delivery performance and 1 signifying exceptional performance.

2.2. Sustainability Applied to SDLC

Energy consumption in software is a performance metric that is part of the non-functional requirements of a software application [32]. Ideally, all applications are optimized to consume minimal energy and maintain low power usage. Furthermore, as new product versions are developed, this KPI should be measured as part of the regression testing to ensure that energy consumption in newer versions is optimized and does not exceed that of previous iterations. Software development companies seeking to create more energy-efficient code often utilize market-available tools that assess code quality. These tools enhance maintainability and efficiency, ultimately contributing to improved software sustainability. One example of these tools is SonarQube, a code analysis tool that focuses on detecting code quality issues, such as bugs, vulnerabilities, and code smells, ensuring that written code is reliable, secure, readable, and modular [33]. This is one of the most adopted open-source static analysis tools: it is being used in more than 100,000 open-source projects and by more than 200,000 development teams [34]. The metrics that have the most impact on sustainability are presented in Table 1 [35].

Table 1. Code metrics that impact sustainability.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Unit of Measure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Smells</td>
<td>Number of instances</td>
<td>Instances of code not following best practices, making it harder to maintain or less efficient</td>
</tr>
<tr>
<td>Technical Debt</td>
<td>Minutes or hours of work required to fix</td>
<td>Work required to address maintainability, efficiency, and reliability issues in the codebase</td>
</tr>
<tr>
<td>Code Complexity</td>
<td>Dimensionless (cyclomatic complexity)</td>
<td>The number of linearly independent paths through a program’s source code, reflecting its complexity</td>
</tr>
<tr>
<td>Code Duplication</td>
<td>Percentage</td>
<td>The portion of duplicated lines in the codebase, with a lower percentage indicating less redundancy</td>
</tr>
<tr>
<td>Test Coverage</td>
<td>Percentage</td>
<td>The proportion of the codebase covered by automated tests, with a higher percentage indicating better testing</td>
</tr>
<tr>
<td>Security Vulnerabilities</td>
<td>Number of instances</td>
<td>The number of security-related issues detected in the codebase, potentially leading to exploits or instability</td>
</tr>
</tbody>
</table>

Technical debt, characterized by compromises made during software development, stands out as a metric with a high impact on sustainability due to the future challenges it presents in maintenance and scaling. Equally significant is code complexity, where an increase indicates multiple paths and conditions that can make the software hard to grasp, test, and sustain over time. Security vulnerabilities, critical in the digital age, possess an indirect high sustainability impact as they can lead to substantial setbacks, including data breaches and loss of user trust. Code smells and duplication, though not immediate threats, indicate potential sustainability issues, making the software challenging to update and maintain. Meanwhile, test coverage has a more indirect influence on sustainability. It assures that, as the software progresses, existing functions remain intact, promoting a more resilient development lifecycle. This scoring model is contingent upon allocating weights to each KPI, reflecting their respective significance in terms of sustainability, as seen in Table 2.
Table 2. Weights assigned to the KPIs as part of the sustainability score model.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Smells</td>
<td>0.15</td>
</tr>
<tr>
<td>Technical Debt</td>
<td>0.2</td>
</tr>
<tr>
<td>Code Complexity</td>
<td>0.2</td>
</tr>
<tr>
<td>Code Duplication</td>
<td>0.15</td>
</tr>
<tr>
<td>Test Coverage</td>
<td>0.2</td>
</tr>
<tr>
<td>Security Vulnerabilities</td>
<td>0.10</td>
</tr>
</tbody>
</table>

The weighting in this sustainability score model reflects the varying impact of KPIs on software project sustainability. Code smells, indicative of potential issues, are assigned a weight of 0.15, reflecting their moderate impact on maintainability. Technical debt and code complexity, both significantly affecting energy efficiency and maintainability, are given a higher weight of 0.2, highlighting their critical role in sustainability. Code duplication, which affects maintainability and indirectly impacts energy efficiency, has a weight of 0.15, suggesting its less severe but still relevant influence. Test coverage earns a weight of 0.2, underscoring its contribution to system stability and indirect support for maintainability. Finally, despite the serious potential consequences, security vulnerabilities are assigned a weight of 0.10, recognizing their indirect influence on sustainability via resource-intensive remediation efforts. Raw KPI values need to be normalized to a range from 0 to 1, where 0 represents the worst possible value and 1 represents the best possible value, to facilitate analysis and interpretation on a common scale.

This ensures that each KPI contributes fairly to the overall score, without any single KPI dominating the result due to differences in measurement units. The following formula will be applied to most KPIs because lower values indicate better sustainability:

$$NKPI = 1 - \frac{CKPI - \text{min}}{\text{max} - \text{min}}$$  \hspace{1cm} (7)

where NKPI is the normalized KPI value; CKPI is the current KPI value; min is the minimum possible value; max is the maximum possible value. For normalizing the test coverage value, the formula should be reversed, since higher test coverage values indicate better sustainability; hence, the following formula can be used:

$$NTC = \frac{CTC - \text{min}}{\text{max} - \text{min}}$$  \hspace{1cm} (8)

where NTC is the normalized test coverage value and CTC is the current test coverage value. The next step is to calculate the weighted score for each KPI by multiplying the normalized value by the weight assigned to that KPI:

$$WKPI = NKPI \times \text{KPIW}$$  \hspace{1cm} (9)

where WKPI is the weighted KPI score and KPIW is the KPI weight. Finally, the overall sustainability score is calculated by adding up the weighted scores of each KPI:

$$SS = \sum_{i=1}^{n} WKPI_i$$  \hspace{1cm} (10)

where SS is the overall sustainability score and n is the number of WKPIs. While SonarQube does not provide predefined minimum and maximum values for all its metrics, it is essential to understand their general behavior and characteristics when used in sustainability assessment.

The maximum values in Table 3 are selected based on general assumptions for medium-to-large projects and should be adjusted according to the specific context of the organizations and industry standards for a more accurate sustainability scoring model. For example,
the maximum value for code smells and security vulnerabilities could be set based on project size and historical data.

Table 3. Predefined minimum and maximum values for the SonarQube.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Smells</td>
<td>0</td>
<td>1000</td>
</tr>
<tr>
<td>Technical Debt</td>
<td>0</td>
<td>500 h</td>
</tr>
<tr>
<td>Code Complexity</td>
<td>0</td>
<td>10,000</td>
</tr>
<tr>
<td>Code Duplication</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Test Coverage</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Security Vulnerabilities</td>
<td>0</td>
<td>100</td>
</tr>
</tbody>
</table>

2.3. Integrating Sustainability into Project Performance Assessment

Integrating sustainability into project performance assessments has become crucial for organizations aiming to ensure long-term success and maintain a competitive edge. By incorporating an SS that evaluates KPIs relevant to software sustainability, organizations can understand their projects’ overall sustainability and performance.

To further enhance the project performance assessment, the SS can be integrated with the PDP score, which combines three essential project performance metrics: DH, LT, and PT. This comprehensive approach provides organizations valuable insights into their projects’ performance while ensuring that sustainability considerations are embedded throughout the project lifecycle. The combined score can be named the comprehensive performance score, and it is calculated using a weighted average of the PDP and SS scores:

$$CPS = w_p \cdot PDP + w_s \cdot SS$$ (11)

where CPS is the comprehensive performance score and $w_p$ and $w_s$ are the weights assigned to the PDP and SS, respectively. These weights can be determined based on the organization’s priorities, industry standards, or expert judgment. The sum of the weights should equal 1. The CPS score also ranges from 0 to 1, with 0 indicating poor overall performance and 1 signifying exceptional performance.

2.4. Project Portfolio Performance

Project portfolio management is recognized in the specialized literature as having a direct link to portfolio success [36]. This approach considers each project’s performance and relative importance within the portfolio, allowing for a more accurate and comprehensive evaluation of the overall portfolio performance. Data can be aggregated at the portfolio level to evaluate a portfolio score by calculating the weighted average of the individual project scores within the portfolio.

The following step-by-step process can be employed to calculate the portfolio score:

1. Calculation of individual project scores: For each project in the portfolio, the PDP score and SS can be calculated as described in previous sections;
2. Assignment of project weights: A weight can be assigned to each project based on its relative importance within the portfolio. The weights may be determined using factors such as project size, strategic alignment, or potential organizational impact. It should be ensured that the sum of all project weights equals 1;
3. Computation of weighted project scores: Each project’s PDP score and SS should be multiplied by their corresponding weights. This results in the weighted PDP score and SS for each project;
4. Aggregation of weighted project scores: The weighted PDP scores and SSs of all projects in the portfolio should be summed separately to obtain the aggregated weighted PDP score and aggregated weighted SS for the entire portfolio;
5. Determination of the overall portfolio score: The aggregated weighted PDP scores and SSs should be combined using a weighted average method. This provides the overall portfolio score, reflecting the entire portfolio’s performance and sustainability.

3. Case Study

This case study aims to showcase the practical application of the process model introduced as part of this research. By conducting a step-by-step analysis, the study highlights the different stages of the implementation process and assesses the resulting outcomes. The chosen portfolio, illustrated in Figure 4, serves as the foundation for demonstrating the proposed process model.

![Portfolio structure selected for the case study.](image)

For the scope of this case study, the application of the scoring model will be limited to the project and portfolio levels. The scoring model will be employed with a set of seven projects constituting a single portfolio. These projects are at various stages of the software development lifecycle, and each has a distinct project development team assigned to it. The case study will commence with the collection of relevant data from each project, including their respective KPIs and any other pertinent information. Next, the model’s parameters, such as scaling factors and weights, will be calibrated based on the collected data, domain knowledge, and other organizational factors. Following the calibration, the model will be applied to calculate each project’s performance and sustainability scores and, ultimately, the portfolio.

3.1. Software Lifecycle Workflow Definition

The methodology for monitoring the progression of projects throughout their lifecycle can be established through the utilization of a project management system employed by the organization. This study showcases the implementation of a Kanban board to manage the company’s portfolio, wherein each column represents a distinct stage in the software development lifecycle. The portfolio consists of various projects, each represented by a corresponding ticket on the board. It is mandatory for all projects to traverse through each phase in the sequence presented in Figure 5.

![SDLC workflow definition.](image)

The cancellation of a project may occur due to various reasons. In such cases, updating the corresponding project and moving it to the “Canceled” status is imperative. The existence of canceled projects is a normal phenomenon within portfolios. The purpose of the discovery phase is to perform a cost–benefit analysis by evaluating the potential monetary value that the project can bring to the organization in relation to the organizational resources required to achieve its objectives [37]. Any project transitioned from the discovery
phase to canceled will not be considered waste. Conversely, projects canceled during the planning, development, or evaluation phases will be recorded as sunk costs.

3.2. Extracting the Data Points Needed to Calculate the Projects’ Score

Table 4 summarizes the relevant data points that must be extracted to calculate the selected KPIs and, ultimately, the PDP score, SS, and CPS.

<table>
<thead>
<tr>
<th>Project</th>
<th>Delivery Health</th>
<th>Lifecycle Time</th>
<th>Phase-Based Time</th>
<th>Completed</th>
<th>Canceled</th>
</tr>
</thead>
<tbody>
<tr>
<td>PX</td>
<td>Current Delay</td>
<td>Current LT</td>
<td>Current PT</td>
<td>Yes/No</td>
<td>Yes/No</td>
</tr>
<tr>
<td></td>
<td>Initial Delivery Date</td>
<td>Start Date</td>
<td>Planning Start Date</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Projected</td>
<td>Evaluation</td>
<td>Evaluation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delivery Date</td>
<td>Start Date</td>
<td>Start Date</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In line with the case study under discussion, Table 5 exhibits a compilation of distinct data points from seven distinct projects currently incorporated in one of the organization’s portfolios.

Table 5. Company portfolio—project status.

<table>
<thead>
<tr>
<th>Project</th>
<th>Delivery Health</th>
<th>Lifecycle Time</th>
<th>Phase-Based Time</th>
<th>Completed</th>
<th>Canceled</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0 days</td>
<td>93 days</td>
<td>77 days</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>N/A</td>
<td>15 September 2021</td>
<td>1 October 2021</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>26 days</td>
<td>238 days</td>
<td>138 days</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>1 August 2022</td>
<td>1 January 2022</td>
<td>11 April 2022</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>27 August 2022</td>
<td>27 August 2022</td>
<td>27 August 2022</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>0 days</td>
<td>169 days</td>
<td>120 days</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>17 December 2022</td>
<td>1 July 2022</td>
<td>19 August 2022</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17 December 2022</td>
<td>17 December 2022</td>
<td>17 December 2022</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>13 days</td>
<td>147 days</td>
<td>106 days</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>22 April 2023</td>
<td>9 November 2022</td>
<td>19 January 2022</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 May 2023</td>
<td>5 May 2023</td>
<td>5 May 2023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>7 days</td>
<td>137 days</td>
<td>119 days</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>16 June 2023</td>
<td>19 December 2022</td>
<td>6 January 2023</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>31 July 2023</td>
<td>5 May 2023</td>
<td>5 May 2023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>17 days</td>
<td>128 days</td>
<td>97 days</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>28 February 2023</td>
<td>9 November 2022</td>
<td>10 December 2022</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17 March 2023</td>
<td>17 March 2023</td>
<td>17 March 2023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>0 days</td>
<td>149 days</td>
<td>N/A</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>31 March 2024</td>
<td>7 December 2022</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>31 March 2024</td>
<td>5 May 2023</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In essence, the data collected from the projects within the portfolio will not merely be a collection of numbers and figures. Instead, it will be transformed into actionable insights and strategic tools that enhance the organization’s ability to manage its projects and portfolios effectively.

3.3. Calculating the DH, LT, PT, and PDP Scores

Within the portfolio under review, seven projects can be identified, each at different stages: one has been discontinued, four have reached their conclusion, and two are currently in progress. The organization that serves as the foundation of this case study is tasked with
managing multiple active and completed portfolios, encompassing over 400 projects in total. This vast pool of historical data is critical in determining the numerical values needed for the variables of the proposed scoring model, as depicted in Table 6. The diversity of these projects provides a broad and varied dataset, ensuring the developed scoring model’s applicability across various situations.

### Table 6. Numeric variable values generated based on organizational historical data.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Accepted Delay</td>
<td>$0.10 \times LT$</td>
</tr>
<tr>
<td>Average LT</td>
<td>146</td>
</tr>
<tr>
<td>Average PT</td>
<td>111</td>
</tr>
<tr>
<td>$k$</td>
<td>0.1</td>
</tr>
</tbody>
</table>

At this point, all the requisite data for calculating the DH, LT, and PT scores for each project have been obtained. Utilizing the equations detailed earlier, the numerical values of the KPIs can be computed. These calculated values are consolidated and presented in Table 7.

### Table 7. DH, LT, PT, and PDP scores for the projects under review.

<table>
<thead>
<tr>
<th>Project</th>
<th>DH Score</th>
<th>LT Score</th>
<th>PT Score</th>
<th>PDP Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1</td>
<td>0.9950</td>
<td>0.9677</td>
<td>0.9907</td>
</tr>
<tr>
<td>P2</td>
<td>0.4452</td>
<td>0.0001</td>
<td>0.0630</td>
<td>0.2384</td>
</tr>
<tr>
<td>P3</td>
<td>1</td>
<td>0.0911</td>
<td>0.2891</td>
<td>0.5950</td>
</tr>
<tr>
<td>P4</td>
<td>0.5424</td>
<td>0.4750</td>
<td>0.6225</td>
<td>0.5456</td>
</tr>
<tr>
<td>P5</td>
<td>0.6615</td>
<td>0.7109</td>
<td>0.3100</td>
<td>0.5860</td>
</tr>
<tr>
<td>P6</td>
<td>0.3965</td>
<td>0.8581</td>
<td>0.8022</td>
<td>0.6133</td>
</tr>
<tr>
<td>P7</td>
<td>1</td>
<td>0.4256</td>
<td>0.0</td>
<td>0.6064</td>
</tr>
</tbody>
</table>

To compute the PDP score, assigning values to the weight parameters $w_1$, $w_2$, and $w_3$ is essential. According to the organization’s specific circumstances, which serve as the basis for this case study, the DH metric holds the highest significance. Therefore, the weight parameter $w_1$ is assigned a value of 0.5. The remaining two metrics constituting the PDP score are considered equally important in this organizational context. Consequently, both weights $w_2$ and $w_3$ are allocated a value of 0.25 each.

3.4. Calculating the SSs for the Projects under Review

Detailed reports of bugs, code smells, and security vulnerabilities are provided by SonarQube, along with the calculation of technical debt and a host of other important metrics. Before data extraction, it was ensured that the SonarQube instance was properly configured, and the projects were correctly set up within the platform. This involved the SonarQube scanner being set up on the machines where the code resided and scanner properties being configured to include the specific files and directories for analysis, excluding any irrelevant to the analysis. It was ensured that all code was compiled and all dependencies were resolved before running the scanner. Once the SonarQube scanner was correctly configured, the scanner was executed, pushing the results to the SonarQube server.

This process was repeated for each project, P1 through P7, ensuring consistency in approach and valid comparative data across all projects. The extracted data can be seen in Table 8. To maintain the validity of the data, it was ensured that the analysis was always performed on the most recent version of the codebase.
Table 8. SonarQube metrics extracted for the projects under review.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>331</td>
<td>385</td>
<td>10,463</td>
<td>25.2</td>
<td>25.2</td>
<td>120</td>
</tr>
<tr>
<td>P2</td>
<td>1080</td>
<td>475</td>
<td>11,000</td>
<td>75.1</td>
<td>95.1</td>
<td>150</td>
</tr>
<tr>
<td>P3</td>
<td>371</td>
<td>286</td>
<td>8937</td>
<td>80.8</td>
<td>60.5</td>
<td>60</td>
</tr>
<tr>
<td>P4</td>
<td>85</td>
<td>451</td>
<td>2672</td>
<td>37.6</td>
<td>54.7</td>
<td>20</td>
</tr>
<tr>
<td>P5</td>
<td>428</td>
<td>520</td>
<td>8429</td>
<td>68.7</td>
<td>80.8</td>
<td>89</td>
</tr>
<tr>
<td>P6</td>
<td>339</td>
<td>391</td>
<td>5927</td>
<td>15.3</td>
<td>37.2</td>
<td>140</td>
</tr>
<tr>
<td>P7</td>
<td>163</td>
<td>74</td>
<td>4916</td>
<td>54.7</td>
<td>51.2</td>
<td>63</td>
</tr>
</tbody>
</table>

This approach allows for a dynamic and up-to-date project assessment, making it possible to track improvements over time and the impact of changes on code quality to be assessed.

The extracted data from SonarQube underwent a normalization process, allowing for a fair and objective comparison of the projects under review. The process involved adjusting the values measured on different scales to a notionally common scale. The KPI values, which have been normalized, are presented in Table 9.

Table 9. Normalized KPI values.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0.669</td>
<td>0.23</td>
<td>−0.0463</td>
<td>0.748</td>
<td>0.252</td>
<td>−0.2</td>
</tr>
<tr>
<td>P2</td>
<td>−0.08</td>
<td>0.05</td>
<td>−0.1</td>
<td>0.249</td>
<td>0.951</td>
<td>−0.5</td>
</tr>
<tr>
<td>P3</td>
<td>0.629</td>
<td>0.428</td>
<td>0.1063</td>
<td>0.192</td>
<td>0.605</td>
<td>0.4</td>
</tr>
<tr>
<td>P4</td>
<td>0.915</td>
<td>0.098</td>
<td>0.7328</td>
<td>0.624</td>
<td>0.547</td>
<td>0.8</td>
</tr>
<tr>
<td>P5</td>
<td>0.572</td>
<td>−0.04</td>
<td>0.1571</td>
<td>0.313</td>
<td>0.808</td>
<td>0.11</td>
</tr>
<tr>
<td>P6</td>
<td>0.661</td>
<td>0.218</td>
<td>0.4073</td>
<td>0.847</td>
<td>0.372</td>
<td>−0.4</td>
</tr>
<tr>
<td>P7</td>
<td>0.837</td>
<td>0.852</td>
<td>0.5084</td>
<td>0.453</td>
<td>0.512</td>
<td>0.37</td>
</tr>
</tbody>
</table>

The occasional appearance of negative values in the normalized metrics must be acknowledged. These negative values are generated when the actual KPI values surpass the “worst-case” benchmarks initially established. In the normalization formula adopted, values that exceed these benchmarks result in outcomes lower than 0. Such outcomes indicate specific metrics and project performances falling below the anticipated worst-case scenarios. This scenario can be addressed by periodically revisiting and adjusting the benchmarks, especially in domains prone to significant metric fluctuations. Such a practice ensures the continued relevance and accuracy of the normalization process. Secondly, the introduction of a buffer range beyond the initially defined worst-case scenario can help. By doing so, unexpected variations can be accommodated without negatively skewing the results.

Following the normalization process, each KPI was assigned a specific weight based on its perceived importance in determining software sustainability, as shown in Table 2. These weights were then applied to the normalized values to derive a weighted score for each KPI. Aggregating these weighted scores yielded a unique sustainability score for each project, serving as a comprehensive indicator of the project’s software sustainability.

P4 and P7 emerge as the most sustainable projects within the portfolio according to the measured KPIs. In contrast, P1 and P2 demonstrate the least sustainability. The remaining projects, P3, P5, and P6, exhibit varying degrees of sustainability, falling between the two extremes. These results continue to underscore the varying degrees of sustainability across the portfolio, as illustrated in Table 10.
Table 10. SSs for the projects under review.

<table>
<thead>
<tr>
<th>Project</th>
<th>SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0.2797</td>
</tr>
<tr>
<td>P2</td>
<td>0.1556</td>
</tr>
<tr>
<td>P3</td>
<td>0.3910</td>
</tr>
<tr>
<td>P4</td>
<td>0.5864</td>
</tr>
<tr>
<td>P5</td>
<td>0.3288</td>
</tr>
<tr>
<td>P6</td>
<td>0.3857</td>
</tr>
<tr>
<td>P7</td>
<td>0.6050</td>
</tr>
</tbody>
</table>

Following the computation of each project’s individual project delivery performance and sustainability scores, an additional calculation was carried out to derive a holistic portfolio score. This was accomplished by applying a unique set of weights to each project, reflecting their strategic importance within the portfolio, as observed in Table 11.

Table 11. Project weights and weighted PDP scores and SSs for the projects under review.

<table>
<thead>
<tr>
<th>Project</th>
<th>Weight</th>
<th>Weighted PDP Score</th>
<th>Weighted SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0.05</td>
<td>0.0495</td>
<td>0.0140</td>
</tr>
<tr>
<td>P2</td>
<td>0.3</td>
<td>0.0715</td>
<td>0.0467</td>
</tr>
<tr>
<td>P3</td>
<td>0.1</td>
<td>0.0595</td>
<td>0.0391</td>
</tr>
<tr>
<td>P4</td>
<td>0.15</td>
<td>0.0818</td>
<td>0.0880</td>
</tr>
<tr>
<td>P5</td>
<td>0.2</td>
<td>0.1172</td>
<td>0.0658</td>
</tr>
<tr>
<td>P6</td>
<td>0.15</td>
<td>0.0920</td>
<td>0.0578</td>
</tr>
<tr>
<td>P7</td>
<td>0.05</td>
<td>0.0303</td>
<td>0.0302</td>
</tr>
<tr>
<td>Total</td>
<td>1</td>
<td>0.5019</td>
<td>0.3416</td>
</tr>
</tbody>
</table>

These weights considered factors such as each project’s alignment with the organization’s broader strategy, the potential impact on customers, future revenue generation, resource requirements, and associated risk levels.

By multiplying the individual project weights by their corresponding PDP scores and SSs, each project’s weighted sustainability and delivery performance scores were derived. The sum of these weighted scores yielded a total portfolio score. The weighted SSs and PDP scores were combined into one overall portfolio score to provide a single, comprehensive measure of the portfolio’s performance. This approach was adopted with the objective of creating an indicator that reflects both the sustainability and delivery performance of the projects within the portfolio. In this process, each of the weighted SSs and PDP scores was assigned a specific weight based on their perceived importance for the overall goals of the portfolio.

These weights would typically be determined based on factors such as the organization’s strategic priorities, industry standards, and specific portfolio objectives. For instance, if the organization’s strategy emphasizes software sustainability over delivery performance more, a greater weight could be assigned to the SS. For this case study, the assigned weight for the PDP score was 0.6 and that for the SS was 0.4.

Using the previously calculated portfolio PDP score (0.4769) and SS (0.4789), the overall comprehensive performance score (CPS) can now be calculated as follows:

\[
CPS = (0.6 \times 0.4769) + (0.4 \times 0.4789) = 0.4378
\]

Therefore, considering the given weights for delivery performance and sustainability, the overall portfolio score is approximately 0.4378. This score represents a weighted combination of the portfolio’s delivery and sustainability performance, providing a comprehensive measure of the portfolio’s overall performance.
3.5. Consolidating the Extracted Data into a Portfolio Delivery Dashboard

The subsequent step involves presenting the data in a visually appealing way for ease of interpretation. Functional leaders and senior company executives often require a concise overview of the delivery aspect of their portfolio, with a visual dashboard presenting aggregated data being the preferred format.

Figure 6 illustrates the proportion of projects categorized as “Timely”, “Delayed”, and “Off Track” within the company’s project portfolio. Data examination shows that two thirds (66.6%) of the projects are not meeting their designated schedules. Of these delayed projects, precisely one half (33.3%) fall within the MAD threshold, effectively handling their delay status. The remaining half (33.3%) have surpassed the MAD, causing a potential overall delay in the delivery of the project portfolio.

Figure 6. Portfolio delivery health.

The data depicted in Figure 6 highlight that, when viewed at the portfolio level, 310 days were dedicated to project discovery activities, and 657 days were allocated for planning and development (phase-based time). This information indicates that a substantial amount of time was devoted to the initial stages of project development, which can likely contribute to the overall success of the portfolio. The importance of thorough planning and development cannot be overstated, as it lays the foundation for a successful project outcome.

While project discovery is an essential stage, it is crucial to note that a substantial allocation of time to it does not necessarily guarantee the portfolio’s success. The data presented in Figure 7 can be transposed into a ratio to demonstrate that 32.1% of the portfolio’s total time was dedicated to project discovery activities. In comparison, 67.9% was allocated to planning and development, as seen in Figure 8.

Figure 7. Project discovery and phase-based time.
Implementing best practices regarding the effort invested in the discovery phase is deemed crucial for the success of projects within organizations. Generally, it should be ensured that the discovery phase is comprehensive enough for the project’s scope to be understood by the project team.

Upon thorough analysis of the portfolio data, a deeper delve into the performance of each project individually becomes crucial. This approach allows for a better understanding of the key performance indicators. When the data are broken down into smaller segments, patterns become easier to discern, potential areas of improvement can be identified, and informed decisions can be made. Therefore, a thorough evaluation of the individual projects within the portfolio is imperative for making well-informed decisions that lead to optimal results. This detailed information can be found in Figure 9.

This data are further scrutinized in Figure 10 and contextualized alongside other company projects and in alignment with the internal metrics employed by the software company to gauge project and portfolio delivery performance.

Based on the project status displayed in Figure 11, just over a quarter of the projects are currently in progress. More than half have reached completion, while a small fraction have been canceled.

The distribution of these statuses can fluctuate based on the specific context of the organization and its portfolio. Generally, a larger proportion of completed projects compared to those initiated and currently underway is often seen as a sign of solid organizational project management. The relatively low percentage of canceled projects—specifically, 14.3% in this case—can be viewed as a positive, as it suggests the organization has effectively avoided the unnecessary expenses related to project cancellations. A more detailed review of individual projects and their outcomes might be required to fully understand the factors
contributing to the lower completion rate and how the potential sunk costs might affect the portfolio.

Figure 10. DH, LT, and PT scores.

Figure 11. The ratio of in-progress, completed, and canceled projects.

The KPIs utilized to calculate the SS are visually represented in Figure 12, prior to applying project weights to determine the portfolio SS. Notably, these values exhibit variability and can even assume negative values, indicating different areas of strength and weakness in each project’s codebase. This perspective provides insight into the portfolio projects from the energy-efficiency standpoint in this case.

Figure 12. Sustainability KPIs and average SSs.

Finally, Figure 13 visually represents the scores used to monitor portfolio delivery performance, with sustainability factors integrated, as a line graph for each project.
The PDP score and SS are higher as they depict the initial scores of the projects before applying the weight factor. On the other hand, the weighted scores are lower as they account for each project’s significance within the portfolio, leading to greater variability than the unweighted scores. Intriguingly, the weighted PDP score and SS are nearly identical to the final CPS, demonstrating that the CPS is closely aligned with these weighted project scores. This close alignment suggests that the CPS effectively encapsulates the project delivery performance and sustainability scores with due consideration of their relative importance within the portfolio.

4. Discussion

Project and portfolio performance measurement plays a crucial role in evaluating the organizational landscape and facilitating the identification of strengths and weaknesses. It enables organizations to monitor their advancement towards established goals and objectives and to implement necessary adjustments as circumstances dictate.

Furthermore, performance measurement assesses the efficiency and effectiveness of the systems utilized to deliver value, thus enabling organizations to make modifications to enhance productivity and realize superior outcomes. The performance measurement principles are also relevant when applied to the software development domain. Effectively addressing stakeholder requirements necessitates the evaluation of delivery performance across various levels and presenting consolidated data transparently and understandably, thereby supporting well-informed decision making throughout the organization. By continuously implementing incremental improvements, software development organizations can enhance project execution, increase success rates, and promote the sustainability of their software development lifecycle.

The detailed metrics are designed to evaluate a software portfolio's delivery and sustainability aspects. Each measurement is given a custom-weighted score, determined by the project’s relevance within the portfolio and the organization’s priorities. Any software company and software project development team wishing to monitor their delivery performance at both project and portfolio levels can utilize this innovative scoring model.

Data from the case study were used to present pertinent performance information for the components of the software portfolio. Each organization should establish thresholds for the acceptable number of projects labeled as delayed or off-course and what is deemed a healthy ratio at the portfolio level.
The high occurrence of delayed and off-course projects within the portfolio underscores the necessity for a thorough review of current project management strategies and implementation of required enhancements.

As recommended in [38], project managers can use risk management techniques to prevent delays or disruptions and to gather data that could indicate the cause of delay, ensuring traceability of cause and effect. As indicated in [39], there is a 60% likelihood for significant IT projects to fail. Therefore, it is crucial to allocate appropriate time for discovery to prevent frequent alterations in objectives and scope and to use scheduling models that optimize the timing of software projects. If the portfolio seldom delivers value, the investment in discovery may become a sunk cost, negatively impacting the portfolio’s financial performance. Agile principles must be embraced to counter this risk, prioritizing regular value delivery [40]. This strategy ensures that resources committed to the discovery and planning stages are effectively used and yield a positive return on investment.

The effort needed for a discovery phase is not fixed or a median, as it can fluctuate based on factors like project complexity, team size, and project goals and objectives. The effort invested in this phase should be tailored to the project’s unique requirements. To facilitate this, organizations must establish their best practices and integrate them into company policies. Once these best practices are in place, specific benchmarks can be set to ensure that the discovery phase’s efforts align with their goals and objectives.

Once guidelines and thresholds have been set, the data extracted can be utilized in the scoring model outlined in this paper to compute the delivery health (DH), lifecycle time (LT), and phase-based time (PT) scores. This process provides a perspective on the project delivery performance within the portfolio, and by calculating the average, a portfolio-level score can be generated. Applying this weighted method ensures relevance and considers each project’s significance within the portfolio. Given that some projects have more strategic importance than others, they may have a greater influence on the overall PDP portfolio score.

As the weight of criteria can considerably affect the decision-making process outcome, it is crucial to give special attention to the objectivity factors of criteria weights, as suggested in [41]. Similar to the “Created vs. Resolved” gadget in Jira, which displays the count of issues created versus those resolved in a project, as mentioned in [42], a comparable method can be implemented at the portfolio level. Examining the correlation between ongoing, completed, and canceled projects within a portfolio provides a comprehensive insight into the portfolio’s performance and health. Incorporating canceled projects into this ratio ensures visibility and control over the associated costs, as no value is retrieved from this category of projects. As stated in [43], project cancellations in the USA account for an annual cost of USD 75 billion, signifying their substantial economic impact. The overall ratio indicates the efficiency of project initiation, selection, and prioritization processes, as well as the ability to deliver projects successfully. It functions as a crucial metric of portfolio progress, thereby aiding in maintaining timeline stability.

The fact that sustainability KPIs are monitored throughout the software development lifecycle allows them to serve as markers of energy-efficient coding. Furthermore, techniques such as code refactoring and process reengineering are utilized to minimize the energy usage of existing software solutions [44]. The scores provided by the scoring model are relative to the maximum acceptable values established within the organization. In the case study presented in this paper, there is potential for enhancement based on the numerical values of the KPIs. Their average is marginally below 0.5, indicating that the code could be optimized for greater efficiency. Integrating software sustainability KPIs with project and portfolio delivery metrics showcases a commitment to green software development. This approach is not only environmentally conscious but also economically strategic, as it aims to prolong the lifespan of hardware [45]. Doing so helps avoid the unnecessary expense and resource consumption associated with replacing hardware systems before the end of their expected lifetime. In accordance with the weighting system presented in the scoring model, the CPS is computed.
This score signifies a numerical value representing the final aggregated score at the portfolio level. This score serves as a benchmark for assessing the current state of the portfolio and identifying areas that may require attention or improvement. The calculated CPS score for the case study is 0.4777, slightly below 0.5, indicating room for improvement in the portfolio’s performance. It is important to note that the goal should be to increase this score over time, reflecting sustainability and delivery performance improvements.

Designed with scalability in mind, the model is capable of expansion. In its initial iteration, the process model employs simple and easily quantifiable metrics, providing a foundation for portfolio performance evaluation. The model can be further enhanced by incorporating additional metrics to broaden the scope of portfolio performance measurement. These could encompass various aspects, such as resource utilization, adherence to project schedules, customer satisfaction, and risk management, at both project and portfolio levels. For future iterations, it would be beneficial to enrich the process model by integrating metrics that specifically target the operational performance of the portfolio, with a particular focus on portfolio agility.

Portfolio agility is defined as the portfolio’s ability to adapt its operations responsively, enabling dynamic adjustments to shift strategic trajectories [46]. Changes in customer or stakeholder demands, competitor actions, technological advancements, or regulatory or legal changes could prompt these shifts. Including such metrics would facilitate a more holistic assessment of the portfolio’s overall performance.

5. Conclusions

The field of information technology has undergone transformative changes in recent times, primarily driven by advancements in project management practices. Notably, the widespread adoption of agile methodologies has significantly enhanced software development efficiency. This trend’s importance is underscored by the PMI report, which highlights that 90% of global senior executives believe agility to be crucial for business success [47].

Emerging from this evolutionary shift towards agile methodologies, there is a newfound opportunity to integrate sustainability metrics into project and portfolio performance assessments. As portfolios continue to grow, the practicality of manual tracking methods remains the same, stressing the need for automated systems that can efficiently handle larger datasets.

In response to this need, this paper introduces a novel process model meticulously designed for software organizations, enabling them to measure portfolio performance without incurring hefty operational costs. The model’s foundation rests on existing project management systems, effectively organizing projects into a cohesive, easily measurable portfolio.

At the heart of the proposed model lies a set of performance metrics deeply rooted in the software development workflow. These metrics, sourced from established project management systems, empower organizations with a wealth of data. Organizations can glean invaluable insights and discern emerging trends by processing this data, facilitating real-time portfolio evaluation and informed decision making.

The research not only presents this model but also has practical implications. The findings underscore the model’s unparalleled efficacy, providing a pragmatic solution for the simultaneous assessment of project and portfolio performance with a dual emphasis on delivery and sustainability. This dual focus reiterates the role of measuring portfolio performance in championing the causes of sustainability and efficiency within software development.

There are limitations associated with this study, particularly the need for additional validation of the selected metrics and KPIs, initially based on the prior literature and expert consultations. Despite this foundation, a comprehensive evaluation against multiple completed projects and established sustainability indicators—such as project longevity, user base stability, and maintenance efforts—is still pending. To address these shortcomings, future research will focus on correlating sustainability scores with practical indicators for ground truthing, soliciting external evaluations of the KPIs by panels of industry experts.
and academic experts, and refining the metric system through sensitivity analyses and stakeholder feedback.

In conclusion, this paper adds a spotlight on the potential of embedding sustainability metrics within the realms of agile software development performance assessments. The proposed model serves as a beacon for software organizations worldwide, promoting enhanced delivery efficiency while bolstering the sustainability ethos of the software development lifecycle. With the software development landscape in a state of perpetual evolution, the emphasis on sustainability principles is set to amplify. This research endeavors to lay the groundwork in this direction, advocating the principles of continuous innovation in project and portfolio management. This indicates a future characterized by environmentally conscious and efficient software processes.

Author Contributions: Conceptualization, C.F., C.C. and A.P.; methodology, C.F., M.C. and O.P.; validation, C.F and C.C.; data curation, C.C. and M.C.; writing—original draft preparation, C.F. and O.P.; writing—review and editing, C.C., M.C. and A.P.; visualization, C.F. and O.P.; supervision, C.C. and A.P.; project administration, C.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Technical University of Cluj-Napoca through the Romanian Ministry of Research and Innovation, CCCDI—UEFISCDI, project number PN-III-P2-2.1-PED-2021-3430/608PED/2022 (Hope2Walk), within PNCDI III. Additional support was provided by the project New Frontiers in Adaptive Modular Robotics for Patient-centered Medical Rehabilitation—ASKLEPIOS funded by the European Union—NextGenerationEU and the Romanian Government under the National Recovery and Resilience Plan for Romania, contract no. 760071/23 May 2023, code CF 121/15 November 2022, with the Romanian Ministry of Research, Innovation and Digitalization within Component 9, investment I8.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Acknowledgments: This work was partially supported by a grant from the Romanian Ministry of Research and Innovation, CCCDI—UEFISCDI, project number PN-III-P2-2.1-PED-2021-3430/608PED/2022 (Hope2Walk), within PNCDI III and partially supported by the project New Frontiers in Adaptive Modular Robotics for Patient-centered Medical Rehabilitation—ASKLEPIOS funded by the European Union—NextGenerationEU and the Romanian Government under the National Recovery and Resilience Plan for Romania, contract no. 760071/23 May 2023, code CF 121/15 November 2022, with the Romanian Ministry of Research, Innovation and Digitalization within Component 9, investment I8.

Conflicts of Interest: The authors declare no conflict of interest.

References


42. Atlassian. Available online: https://confluence.atlassian.com/display/JIRA051/Adding+the+Created+vs+Resolved+Gadget (accessed on 20 May 2023).
43. Ahonen, J.J.; Savolainen, P. Software engineering projects may fail before they are started: Post-mortem analysis of five cancelled projects. J. Syst. Softw. 2010, 83, 2175–2187. [CrossRef]
44. Şanlıalp, İ.; Öztürk, M.M.; Yiğit, T. Energy Efficiency Analysis of Code Refactoring Techniques for Green and Sustainable Software in Portable Devices. Electronics 2022, 11, 442. [CrossRef]
47. Fulea, M.; Mocan, B.; Dragomir, M.; Murar, M. On Increasing Service Organizations’ Agility: An Artifact-Based Framework to Elicit Improvement Initiatives. Sustainability 2023, 15, 10189. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.