applied sciences

MDPI

Article

Integrating Sentiment Analysis into Agile Feedback Loops for Continuous Improvement

Diogo Marçal [1], José Metrôlho [1,2] and Fernando Ribeiro [1,2,*]

[1] School of Technology, Polytechnic Institute of Castelo Branco, 6000-081 Castelo Branco, Portugal; d.marcal@ipcbcampus.pt (D.M.); metrolho@ipcb.pt (J.M.)
[2] CISeD—Research Center in Digital Services, Instituto Politécnico de Viseu, 3504-510 Viseu, Portugal
* Correspondence: fribeiro@ipcb.pt

Abstract

The pursuit of continuous improvement is a defining feature of agile software development, yet its success depends on the systematic collection and interpretation of team members' feedback. Conventional mechanisms, such as retrospectives and surveys, provide valuable insights but are often constrained by their episodic nature and susceptibility to subjective interpretation. This study examines the potential of Artificial Intelligence (AI), and in particular sentiment analysis, to complement feedback-driven practices and strengthen continuous improvement in agile contexts. Two literature reviews were conducted: one on applications of AI across software engineering domains and another focusing specifically on sentiment analysis in agile environments. Based on these insights, a prototype tool was developed to integrate sentiment analysis into task management workflows, enabling the structured collection and analysis of developers' perceptions of task descriptions. Semi-structured interviews with experienced project managers confirmed the relevance of this approach, highlighting its capacity to improve task clarity and foster more transparent and inclusive feedback processes. Participants emphasized the value of the proposed approach in generating rapid, automated insights, while also identifying potential limitations related to response fatigue and the reliability of AI-generated outcomes. The findings suggest that incorporating sentiment analysis into agile practices is both feasible and advantageous, providing a pathway to align technical objectives with developer experiences while enhancing motivation, collaboration, and operational efficiency.

Keywords: agile software development; continuous improvement; feedback analysis; sentiment analysis; task description quality

Academic Editors: Alessandro Gasparetto and Douglas O'Shaughnessy

Received: 29 September 2025
Revised: 27 October 2025
Accepted: 18 November 2025
Published: 20 November 2025

Citation: Marçal, D.; Metrôlho, J.; Ribeiro, F. Integrating Sentiment Analysis into Agile Feedback Loops for Continuous Improvement. *Appl. Sci.* **2025**, *15*, 12329. https://doi.org/10.3390/app152212329

1. Introduction

Continuous improvement is a systematic effort aimed at optimizing organizational processes, products, and services through planned and incremental actions [1,2]. Known as Kaizen [3], this philosophy originated in Japan and gained prominence in the automotive industry during the 1980s, particularly through the practices introduced by companies such as Toyota. Since then, it has become a cornerstone of operational excellence initiatives in a wide range of sectors. Its success relies heavily on collective commitment, especially from the management, and on the active involvement of employees, who are encouraged to critically reflect on their own work routines and propose concrete improvements on a regular basis [3,4]. In the context of software development, continuous improvement is strongly aligned with agile methodologies such as Scrum and Kanban. These approaches

promote adaptability and learning through short, iterative cycles, enabling teams to deliver value incrementally while responding to changing requirements and priorities [2,3]. Regular feedback loops, including sprint reviews and retrospective meetings, play a central role in these frameworks. They serve as structured opportunities for teams to assess progress, discuss challenges, and define process refinements that enhance both team performance and product quality [4]. Despite the operational frameworks in place, the effectiveness of continuous improvement in agile environments ultimately depends on how well the perceptions, experiences, and concerns of individual team members are understood and acted upon. Developers are directly involved in task execution and are thus well positioned to provide insights into issues such as unclear requirements, excessive workload, inadequate tooling, or communication breakdowns. Capturing and analyzing this feedback in a structured way is therefore essential to inform decision-making, guide task definition, and maintain a healthy team dynamic over time.

Traditionally, feedback collection has relied on explicit, manually gathered inputs, such as surveys, meetings or interviews, which, while useful, are often time-consuming and limited in scale. Moreover, much of the sentiment embedded in day-to-day project communications remains implicit and unstructured, scattered across artefacts such as commit messages, issue trackers, code reviews, and chat logs. This is where techniques such as sentiment analysis, also known as opinion mining, can play a transformative role. Sentiment analysis is a Natural Language Processing (NLP) technique that allows for the automatic classification of the emotional tone of a given text, distinguishing between positive, negative, and neutral expressions [5,6]. When applied to textual data generated in software development projects, it provides a scalable means to uncover patterns of emotion and attitude among team members. Considering that an estimated 80% of corporate data is unstructured [7], the ability to analyze such content systematically can yield valuable insights that would otherwise go unnoticed.

In the software development domain, several studies have explored how Artificial Intelligence (AI) and NLP can enhance key dimensions of software project management. Examples include automated requirements prioritization [8], task effort estimation in story points [9], use-case quality assessment [10], project success prediction [11], and data-driven retrospectives for workload forecasting [12]. Collectively, these studies demonstrate the potential of AI and NLP to increase efficiency, accuracy, and objectivity in project management activities that were previously dependent on subjective judgment. However, most of these approaches focus primarily on quantitative and operational aspects, such as effort estimation, task assignment, or risk prediction, while paying limited attention to the qualitative and affective dimensions of software development, including how developers perceive, interpret, and emotionally respond to their work environment. In fact, sentiment analysis has been increasingly explored as a tool to support team management, quality assurance, and process improvement. But it can also help detect signs of frustration or confusion related to specific tasks, identify moments of declining morale, or reveal recurring sources of inefficiency. When used continuously and in combination with other agile practices, this approach can contribute to more proactive and informed interventions by project managers, ensuring that small issues are addressed before they evolve into critical bottlenecks. Furthermore, it can support more accurate and empathetic task definition, aligning technical planning more closely with the actual experiences of those carrying out the work.

This study seeks to address that gap by investigating how AI-based sentiment analysis can be embedded within agile feedback loops to continuously capture and interpret developers' perceptions, and how it can be used to improve task clarity while fostering more transparent and inclusive feedback processes. In contrast to prior research that primar-

ily emphasizes automation and prediction, this approach focuses on integrating opinion mining directly into the iterative feedback mechanisms that drive continuous improvement.

To achieve this goal, this study combines two complementary research components: a dual literature review examining (i) how AI has been applied across various subfields of software engineering, including project management, testing, requirements engineering, and development, and (ii) how sentiment analysis has been used to extract value from developer feedback in agile contexts; and the design and evaluation of TaskIT Version 1.0 (hereafter, TaskIT) a prototype AI-driven sentiment-analysis tool that continuously captures developers' perceptions of the clarity and quality of the task descriptions they execute. This integrative approach bridges the gap between human-centred feedback practices and AI-enhanced management systems, fostering a more adaptive and empathy-aware model of continuous improvement in software engineering.

To assess the practical relevance of this approach, the study conducted semi-structured interviews with a cohort of experienced software-engineering project managers. Their insights confirm that surfacing sentiment trends in near-real time can expose latent bottlenecks, clarify ambiguities in task definition, and prompt earlier, more focused interventions. When the tool's analytics are embedded within established project-management platforms, interviewees observed a clear potential to streamline decision-making, reinforce developer well-being, and ultimately enhance both process efficiency and product quality. These findings motivate the empirical study presented in the remainder of this article and underscore the value of integrating AI-based sentiment analysis into the everyday workflow of agile software projects.

Thus, the main contributions of this article are as follows:

- Contribution to the Literature on AI Applications in Software Engineering Process Improvement: By combining a literature review with an applied case study, the work bridges gaps between theoretical AI capabilities and practical challenges faced in software project management.
- Demonstration of Sentiment Analysis as a Proactive Risk Detection Mechanism: The study highlights how sentiment trends can signal emerging issues, such as unclear tasks or team stress.
- Through semi-structured interviews, the study gathers expert feedback on the usability, relevance, and potential impact of AI-based sentiment analysis in managing agile software projects, providing valuable practical perspectives to complement the technical development.
- Integration of Sentiment Analysis with Agile Project Management Practices: A methodological approach provides a foundation for further exploration of AI-enhanced feedback analysis in software engineering, encouraging the development of more sophisticated, context-aware sentiment analytics tools.

2. Materials and Methods

Software engineering research increasingly emphasizes the integration of human-centered practices with technology-driven innovations to enhance development processes and project outcomes. Within this context, two interrelated paths have gained prominence: structured feedback mechanisms that promote continuous improvement in agile environments, and the application of AI in software project management. While these areas differ in focus and methodology, they are connected by the shared objective of enhancing productivity, decision-making, and process efficiency. Understanding how these complementary approaches operate, one focusing on human-driven improvement, the other on technology-enabled optimization, provides a foundation for examining their combined potential in advancing software engineering practices.

Section 2.1 examines studies on the systematic use of feedback as a driver of improvement at the individual, team, and process levels. Building on this understanding of human-centered improvement, Section 2.2 addresses the role of AI in software project management. By integrating these perspectives, this section highlights the mutually reinforcing nature of feedback and AI interventions.

2.1. Leveraging Feedback for Continuous Improvement

As software development becomes increasingly complex and fast-paced, particularly in agile environments, the ability to adapt quickly is essential for delivering high-quality products and sustaining team performance. Feedback plays a central role in this process. When actively sought and thoughtfully integrated into daily workflows, it not only helps identify bugs and inefficiencies but also reveals opportunities to improve practices, processes, and solutions. Beyond technical refinements, feedback promotes learning and shared reflection, supporting continuous team growth. In agile contexts, regular feedback loops, from users, team members, and stakeholders, are crucial to ensuring that products remain aligned with real-world needs.

The objective of this review is to examine how feedback mechanisms contribute to continuous improvement in agile software development, focusing on their sources, the levels at which they operate, the competencies they enhance, and the organizational benefits they provide. It reviews studies that investigate how feedback mechanisms can be leveraged in agile software development. The analysis emphasizes the sources of feedback, the levels at which it operates, individual, team, or process, the competencies it enhances, and the organizational benefits it produces. The analysis is grounded in a review of related work conducted according to the PRISMA methodology [13].

To objectively understand the contribution of feedback to continuous improvement in the context of software development, the review is guided by four research questions (RQ):

RQ1—What sources of feedback are used to promote continuous improvement?
RQ2—Does continuous improvement through feedback occur at the level of individuals, processes, teams, or all of these?
RQ3—What individual competencies of each development team member can benefit from the feedback process?
RQ4—In what ways do companies and software development projects using agile methodologies benefit from the feedback process?

2.1.1. Inclusion Criteria

In line with the research objectives, the following inclusion criteria were defined to determine which studies would be considered for analysis:

- Address software development or the management of software development projects.
- Refer to feedback processes of any type or origin, provided they occur within software development teams.
- Refer to software development based on agile methodologies.
- Address the topic of continuous improvement in the context of software development.
- Published between 2014 and 2024.

2.1.2. Exclusion Criteria

To ensure that only works directly related to the subject under investigation were analyzed, the following exclusion criteria were applied:

- Studies that did not address the topic of software development or its management.
- Studies that did not address feedback from members of development teams.
- Studies that did not address agile methodologies.

- Studies that did not refer to continuous improvement within the scope of software development.
- Studies that were retracted, unavailable, or duplicated.

2.1.3. Data Sources

The search for relevant works was carried out using the Scopus scientific database (www.scopus.com) and the IEEE Xplore digital library (https://ieeexplore.ieee.org/). These databases were selected because they offer extensive coverage of high-quality research in computer science and software engineering, ensuring the inclusion of the most relevant and reliable studies. In addition, due to the limited number of records retrieved from Scopus and IEEE Xplore, Google Scholar (https://scholar.google.com/) was also used to identify additional relevant works.

2.1.4. Selection Process

An advanced search was performed employing multiple query strings that combined relevant keywords to identify studies examining how feedback from team members, within the context of agile software development, can be harnessed to foster continuous improvement. The search was restricted to the title and abstract fields, applying the following query expression:

(software AND (develop* OR project OR manage* OR process OR method*))
AND (feedback OR comment* OR comunicat* OR collaborati*)
AND agile AND team* AND improvem*

The search, conducted between early November 2024 and late December of the same year, yielded a total of 89 studies. Of these, 86 studies were excluded for not aligning with the objectives of the intended research:

- 65 excluded after reading the title.
- 17 excluded after reading the abstract.
- 4 excluded after a full-text review.

This process resulted in three related studies, two of which were highly similar, authored by the same researcher, and referred to the same work. Consequently, only one of them was considered for analysis [14]. Since the initial search yielded only two studies [14,15] for analysis, an additional open search was conducted in other digital libraries, including IEEE Xplore and Google Scholar. In these subsequent searches, the same expressions and terms applied in Scopus were used. This process made it possible to identify three additional related articles [16–18], which were also included in the analysis.

2.1.5. Analysis of the Selected Studies

The analysis, carried out to address the RQ, is presented in Table 1.

The studies reveal a diverse range of feedback sources in agile software development (RQ1). Feedback may arise from user stories, tickets, commits, and retrospective meetings [14], as well as from formal managerial channels and informal peer interactions, involving both team leaders and team members [15–18]. Additionally, self-assessment practices serve as a complementary source of feedback [17]. Collectively, this indicates that feedback emerges from both technical artifacts and interpersonal exchanges, underscoring their multifaceted role in promoting continuous improvement.

Regarding RQ2, continuous improvement occurs across multiple organizational levels. Most studies focus primarily on the individual level, highlighting enhancements in personal skills and productivity [15–18], while others examine process-level improvements, emphasizing the refinement of development practices and workflows [14]. Feedback also influences team-level collaboration and guidance [18]. Consequently, feedback mechanisms

contribute to improvement across individuals, teams, and processes, depending on the source and structure of the feedback.

Table 1. Comparative summary of studies on leveraging feedback for continuous improvement.

Study	RQ1	RQ2	RQ3	RQ4
[14]	Derived from user stories, tickets, commits, and team reflections during retrospective meetings.	Process.	Does not mention individual skills benefited.	Use of data generated by activities to produce more accurate feedback that can improve both the software product and the way it is developed.
[15]	Formal feedback provided by management and informal feedback from both management and direct work colleagues.	Individuals and their skills.	Communication, technical skills, commitment, proactivity, cooperation, and individual productivity.	Increased productivity through enhanced employee motivation.
[16]	Provided by management and direct work colleagues.	Individuals and their skills.	Individual productivity, technical knowledge, punctuality, management skills, communication, initiative, and commitment.	Teams motivated by positive feedback and improved skills are more productive and achieve their goals more easily.
[17]	Self-assessment by each individual, complemented by feedback from direct colleagues, and overseen by management.	Individuals and their skills.	Communication, productivity, quality, technical knowledge, punctuality, work management, commitment, teamwork, willingness to improve, and initiative.	Human capital with enhanced skills through accurate and personalized development processes is among the most important factors in organizations. Transparency in feedback and alignment between self-feedback and peer feedback.
[18]	Performance indicators, management, and team colleagues.	Individuals and their skills, and processes.	Willingness to learn, work motivation and reduction of impostor syndrome.	Feedback fosters greater collaboration and trust within teams, increasing operational performance and reducing undesired staff turnover. It also enhances the motivation of each individual.

Feedback enhances a wide range of competencies (RQ3). Recurrent themes include communication, technical skills, commitment, initiative, productivity, punctuality, teamwork, proactivity, and willingness to learn [15–18]. Feedback also fosters management and work organization skills, along with psychosocial benefits such as increased motivation and reduced impostor syndrome [18]. These findings suggest that feedback mechanisms support both hard skills and soft skills, contributing to comprehensive professional development.

Organizations and agile projects benefit through multiple channels (RQ4). Improved product quality and development accuracy [14], enhanced individual and team productivity [15,16], increased motivation and engagement [16,18], and greater trust and collaboration within teams [18] are highlighted. Feedback also enables more transparent and

personalized development processes, strengthening human capital and aligning individual performance with organizational goals [17]. Collectively, these benefits highlight the pivotal role of feedback in promoting both operational efficiency and strategic alignment.

The responses to the RQ reveal that feedback in agile software development originates from a broad spectrum of sources, including technical artifacts such as user stories, tickets, commits, and retrospective meetings, as well as formal managerial channels, informal peer interactions, self-assessments, and performance indicators (RQ1). Continuous improvement is driven at multiple organizational levels, individual, team, and process, depending on the source and structure of feedback (RQ2). At the individual level, feedback fosters both technical and interpersonal competencies, such as communication, productivity, teamwork, commitment, initiative, and willingness to learn, while also enhancing motivation and reducing psychological barriers like impostor syndrome (RQ3). For organizations, the benefits extend to higher product quality, increased accuracy in development, improved collaboration and trust, stronger human capital, and closer alignment between individual performance and strategic goals (RQ4). Collectively, these findings underscore the role of feedback as a multidimensional driver of improvement, integrating human development, operational efficiency, and strategic alignment within agile methodologies.

Although the reviewed studies employ distinct methodological approaches, several converging conclusions can be identified. A recurrent aspect is the distinction between formal feedback, which is well-defined and structured, and informal feedback, which arises spontaneously during day-to-day team activities [15,16,18]. Informal feedback is perceived as particularly valuable when exchanged frequently among team members and from management, with professionals emphasizing the importance of having the autonomy to choose the receivers of their feedback. While most do not regard providing feedback to all colleagues as burdensome, some caution that excessive breadth may reduce its acceptance and clarity. Feedback should be carefully planned, delivered, and communicated [17,18]. It should be grounded in objective data on individual performance, complemented by the perspectives of peers and supervisors on the individual's conduct across various dimensions. Conversely, when poorly delivered, feedback may be rejected by the receiver, resulting in diminished motivation, productivity, and engagement in improvement efforts. Several studies [14–16] highlight the role of peer-to-peer feedback. Another noteworthy aspect, highlighted in study [18], indicates that the use of dedicated software, allowing feedback to be provided and monitored by various team members, particularly by the responsible management, may represent a way to enhance the feedback process in agile development environments. Ultimately, feedback serves as a catalyst for both motivation and productivity, as motivated professionals, aware of their strengths and weaknesses, are more receptive to improvement, thereby fostering technical growth, stronger collaboration, and strategic goal attainment in agile software development contexts.

2.2. Leveraging AI and NLP for Enhanced Software Project Management

This section presents an analysis of studies that propose solutions integrating AI or NLP mechanisms into the processes and activities of software project management. The goal of the review is to explore how AI and NLP are applied in software project management, focusing on the areas they aim to improve, the tools and data used, the results achieved, and the benefits reported compared to conventional practices. As in the research presented in Section 2.1, the PRISMA methodology [13] was also used to conduct the analysis of the related studies.

For this analysis, five research questions (RQs) were considered:

- RQ5—Which project management activities are addressed and intended to be improved using AI or NLP in each study?

- RQ6—What technologies are used in the development of the proposed solutions in each analyzed study?
- RQ7—What is the origin of the data, and what are its properties used in the development of proposed solutions in each study?
- RQ8—What is the final product or output of the solution developed in each analyzed study?
- RQ9—What advantages do the authors of each analyzed study identify in using their proposed solutions compared to conventional approaches in software project management?

2.2.1. Inclusion Criteria

Inclusion criteria were established to ensure that the most relevant related works were selected. Studies to be analyzed in this research phase were required to:

- Address at least one of the following topics: AI or NLP.
- Cover aspects and/or typical activities of software development.
- Present concrete solutions involving the use of the aforementioned technologies.
- Explore how these technologies contribute to the improvement of software project management.
- Be published after 2014.

2.2.2. Exclusion Criteria

The following exclusion criteria were applied:

- Studies related to areas other than software development.
- Studies published in languages other than Portuguese or English.
- Studies published more than ten years ago.
- Studies addressing a technological area in a generic manner for improving software project management activities.

2.2.3. Data Sources

For this study, only the Scopus database was used. It was chosen for its extensive coverage of peer-reviewed literature in computer science and software engineering, as well as its reliable indexing, comprehensive metadata, and advanced search functionalities, which facilitate systematic and reproducible reviews.

2.2.4. Selection Process

The method and procedure used to identify studies related to the topic under investigation are described below. A search was conducted with the objective of identifying articles that present practical solutions involving the integration of AI and NLP to improve software project management. A time limit of approximately 10 years was applied, starting from 2014, since this period marked a significant increase in the use of such technologies. The selected studies had to be written in either Portuguese or English, and the search terms had to appear in the abstract. The search query used was the following:

"project manag*"
AND "software"
AND ("artificial intelligence" OR "natural language")

The execution of this query resulted in a total of 109 studies, from which the following were excluded:

- 35 excluded after reading the title.
- 49 excluded after reading the abstract.
- 1 excluded due to unavailability.
- 14 excluded after full text review.

Thus, of the initial 109 studies, 10 were considered relevant to the topic under analysis. Among these, two were systematic reviews and were therefore not subjected to direct individual analysis. However, these two literature reviews were included in a comparative analysis conducted after the individual review of the remaining eight studies, to identify points of convergence with the analyzed works.

2.2.5. Analysis of the Selected Studies

This section presents the results of the analysis of the selected studies based on the research questions outlined earlier. It summarizes both the unique characteristics of each study and their commonalities. Table 2 presents the findings, organizing the studies according to their responses to the research questions.

The project management activities addressed and targeted for improvement using AI or NLP (RQ5) include requirements prioritization, task estimation, project success prediction, use case quality evaluation, and bug report management. Three studies [9,19,20] focus on task/issue effort estimation. These solutions predict the effort required for a task based on its textual description, with estimations expressed using different units or metrics, such as minutes, days, or story point ranges/classes. The remaining studies address a broader range of activities, including requirements prioritization [8], use case quality evaluation [10], and project success prediction [11].

Most studies report on employing ML and NLP algorithms, with Python as the primary programming language (RQ6).

The data used by these algorithms (RQ7) predominantly originates from real software development projects, although some studies rely on synthetic datasets or bug reports. Additionally, these studies commonly utilize task attributes from real projects, such as titles, descriptions, and actual durations.

The final deliverables of the proposed solutions (RQ8) vary across studies, with some examples including effort estimation for task completion, quality assessment of use case descriptions, and task prioritization.

Addressing RQ9, which concerns the advantages most frequently highlighted by the authors in using their proposed solutions compared to conventional approaches in software project management, the analysis shows that integrating AI and NLP technologies can streamline processes and activities that would otherwise require significant time and resources if performed manually. These activities become data-driven rather than reliant on the experience of individual professionals. By saving time, reducing resource consumption, and improving accuracy, organizations are better positioned to achieve their objectives efficiently.

Despite the diversity of methods, several commonalities were identified at various levels, particularly regarding the targeted activities that the studies and their proposed solutions aim to enhance, as well as the advantages claimed compared to traditional practices. A recurring theme across multiple studies is the benefit of integrating AI and NLP mechanisms into project management. Explicitly or implicitly, the proposed solutions aim to automate processes and activities that are typically performed manually and empirically by software project managers. Several studies [8,11,12,20] highlight that, through this automation, the targeted activities no longer rely solely on the experience of individual professionals, who are prone to errors. Instead, management activities become data-driven, leveraging historical data to improve the accuracy of outcomes, such as task effort estimation. This approach increases objectivity and contributes significantly to the success of software projects. Another advantage of these automated solutions is that project professionals are relieved from performing these activities manually or empirically, allowing them to focus on higher-value tasks, such as development activities, thereby

saving time and reducing costs. Study [8] also reports an increase in scalability concerning the targeted activities it addresses.

Table 2. Comparative summary of studies on leveraging AI and NLP for enhanced software project management.

Study	RQ5	RQ6	RQ7 (Data Source)	RQ7 (Data Fields)	RQ8	RQ9
[8]	Automated requirements prioritization.	TF-IDF and K-means.	120 requirements written by 16 software development students.	Textual description, category, and manually assigned priority.	Requirements are automatically classified using MoSCoW rules.	Objectivity (independent of team experience), timesaving, and scalability.
[9]	Task effort estimation in story points.	Graph Neural Network.	23,000 issues from 16 projects.	Title, textual description, and actual story points.	Task classification within story point ranges (small, medium, large, extra-large).	Advantages over other automated methods. Improved prediction accuracy is highlighted.
[10]	Evaluation of use case quality.	NLP, spaCy modules, WordNet, among other Python tools.	35 use cases created by professionals.	Textual description evaluated in terms of length, structure, etc.	Determines whether a use case is of good quality and identifies issues if not.	The solution improves the quality of use case content.
[11]	Project success prediction based on its characteristics.	Python, Deep Neural Networks, and forecasting models.	Public datasets from real and simulated projects.	Budget, team size, qualitative risk level, among others.	Indicates whether a software project will succeed (1) or not (0).	More accurate predictions compared to empirical forecasting.
[12]	Analysis of project data to identify improvement opportunities for Sprint Retrospectives.	ML models for prediction and classification (KNN, linear regression).	Jira data from software projects.	Issue type, issue priority, sprint info, sprint velocity, started and completed backlog items, etc.	Forecast workload capacity for upcoming sprints. Classification of issues (task, bug, etc.).	Improved forecasting based on historical data analysis. Conventional methods are empirical and resource-intensive.
[19]	Effort estimation for client/user-requested change tasks.	Classification and prediction algorithms (Random Forest, k-grams).	1648 change requests/tasks from 7 real clients. Real data from tools like Jira.	Software type, project year, task description, actual task duration.	Estimated time in minutes to complete the task.	Automated effort estimation. Saves time and resources. Reduces dependency on human estimations that may be error-prone in new projects.
[20]	Time estimation for development and testing tasks. Task assignment to the most qualified team member.	Chatbot, MongoDB, linear regression, XGBoost, LSTM, SVM, KNN.	Real Siemens projects tracked in Jira.	Issues: textual descriptions, associated project, priority. Professionals: name, project allocation, task history, role (dev or tester).	Time estimation (in days) to complete a task. Suggestion of first- and second-best candidates to perform the task.	Reduces errors in task estimation and assignment. These activities no longer rely solely on professional experience, but on automated data analysis.
[21]	Management of bug reports from mobile app users.	ML and NLP.	Bug reports submitted by users via a messaging app.	Audio, screenshots, and textual bug descriptions.	Transformation of bug reports into developer-assigned tasks, including detection of duplicate reports.	Easier for users to report bugs compared to conventional platforms.

The advantages and commonalities identified in this analysis align with the findings presented in [22]. According to the authors of that study, the incorporation of AI into software project management aims to replace traditional and empirical methods. They report that AI can make decision-making faster, more accurate, more objective, and less resource- and effort-intensive for both project management and development teams. For instance, NLP techniques are used to process textual elements generated by various project

activities, particularly to improve the specification and understanding of functionalities, an observation that is also supported by our analysis of the selected studies, especially those in [8,10].

The review presented in [23] focused on the integration of AI into software project management across three dimensions: Technology, Organization, and Environment. According to the authors of [23], AI integration enhances the objectivity and accuracy of predictions, such as the effort or time required to complete tasks. They identify the estimation of metrics, including effort, risk, and time, as one of the most significant areas based on the number of solutions analyzed. This conclusion is consistent with our findings, as three of the eight studies reviewed [9,19,20] specifically focus on these estimation activities. Another advantage highlighted in that study is the improved allocation of human resources, since management and planning activities that were previously performed manually can now be automated and executed more efficiently.

2.3. Complementary Roles of Feedback and AI in Agile Software Development

As evidenced by the analyses in Sections 2.1 and 2.2, both approaches may consistently contribute to improvement in agile software development: on one hand, structured and human-centered feedback practices; on the other, technology-driven optimization through AI and NLP. Each has proven effective in isolation, yet their combined use, particularly when mediated by dedicated platforms that allow feedback to be gathered, tracked, and interpreted by both team members and management, may offer considerable potential to strengthen feedback processes.

In agile settings, feedback drawn mainly from sources such as user stories, retrospectives, and peer reviews underpins the incremental refinement of both processes and deliverables. Relying exclusively on conventional channels imposes limitations arising from their infrequent occurrence, inconsistent engagement from participants, and the interpretative subjectivity inherent to qualitative feedback. These limitations highlight the opportunity to complement and extend established practices through purpose-built software incorporating AI capabilities, tools that enable continuous input collection from multiple communication streams, apply sentiment analysis to uncover underlying attitudes, and present evolving trends in accessible visual formats. For project leaders, such tools can support the continuous monitoring of the team's emotional and cognitive climate, rather than relying solely on periodic reports. When sentiment data is connected to operational indicators, such as task complexity, workload distribution, and delivery performance already captured by AI-enabled management systems, it becomes easier to identify early signs of declining morale, recurring sources of frustration, or breakdowns in communication, and to intervene before these factors compromise project outcomes. Integrating structured feedback with AI-driven analytics can help create a more transparent and inclusive work environment. Contributions from developers, testers, and other stakeholders can be systematically captured, categorized, and prioritized according to urgency and relevance, with actionable insights feeding directly into agile artefacts such as sprint backlogs or task boards. This approach strengthens, and makes more visible, the connection between the team's lived experience and the project's technical objectives.

It is important to consider that in agile methodologies, feedback analysis has traditionally been conducted by team leaders and project managers during retrospective meetings, offering a structured forum for collective reflection and open dialogue. This remains a core practice consistent with agile principles. A hybrid approach, where traditional retrospective-based discussions are complemented by continuous, AI-assisted monitoring, has the potential to deliver richer and more timely insights while preserving

the collaborative, human-centered ethos that underpins agile development, particularly if it is not overly intrusive and does not impose significant time demands on the team.

3. Team Leader Perspectives on Feedback in Software Project Management and the Role of AI in Its Analysis

To meet the study objectives, the research was structured into three interrelated stages.

In the first stage, two systematic literature reviews (Sections 2.1 and 2.2) were conducted to identify the main approaches to (i) leveraging feedback to foster continuous improvement in agile environments and (ii) applying AI and NLP techniques to software project management. This stage was essential to understand how feedback from development team members has been examined in the literature and how its integration with AI can be further enhanced to strengthen their complementary roles in agile software development.

In the second stage, a prototype of TaskIT was developed to operationalize these insights. TaskIT enables the structured collection and analysis of developers' perceptions of task descriptions, and its implementation combines conventional project management features with AI-based sentiment analysis embedded within agile feedback loops, allowing the continuous capture and interpretation of developers' feedback regarding task description quality.

In the third stage, semi-structured interviews with experienced project managers were conducted to empirically assess the proposed approach. This stage aimed to assess the practical relevance of the approach, particularly the potential contribution of AI-based sentiment analysis to managing agile software projects and providing valuable practical perspectives to complement the technical development. This stage was important to examine the extent to which software professionals considered the approach and the collected metrics relevant and useful for promoting the continuous improvement of task writing. These interviews also sought to assess the tool's potential for systematically evaluating developer perceptions of task descriptions and, ultimately, its contribution to continuous improvement.

3.1. A Tool for Assessing and Tracking the Improvement of Task Description Quality

TaskIT was developed to assess and monitor the quality of task descriptions, supporting continuous improvement in software development processes. The application integrates a Kanban board to track progress under agile methodologies such as Scrum and it enables professionals with different roles within a team to monitor the progress of a software development project. It was designed to enable the evaluation and monitoring of task description quality over time, thereby supporting and promoting continuous improvement.

TaskIT supports four user roles, which can be held simultaneously by a single user:

- Project Manager: Responsible for overseeing the software development project, managing user roles, configuring project settings, and tracking developer responses and the resulting analyses throughout the sprint.
- Unauthenticated User: Not logged into the system, with very limited functionality.
- Developer: Performs technical tasks such as programming. Developers can assign themselves to tasks, move them across the Kanban board to update their status, and, upon completing a task, provide feedback on the quality of its description.
- Task Creator: The only users who can create tasks, specifying descriptions, titles, and estimated story points.

TaskIT operates at the project level, with a Project Manager responsible for managing sprints and monitoring the evolution of task-writing performance over time. Each task records its author and executor, and incomplete tasks are automatically carried over to subsequent sprints.

When developers move a task to the Done column, they are prompted to answer questions on the clarity and quality of its description. These responses are analyzed using sentiment analysis and opinion mining, through Microsoft Azure services, to identify areas for improvement in future tasks. The system thus captures and processes developers' feedback, generating insights that can be discussed in retrospective meetings. By incorporating their perspectives, the platform not only enhances the clarity and conciseness of task descriptions, reducing blockers, delays, and unforeseen issues, but also fosters developer motivation by demonstrating that their input informs process improvement.

Task quality is assessed through seven guiding questions, each linked to a specific evaluation dimension. The first addresses the clarity of the description and whether it enabled task initiation without uncertainty. The second focuses on the presence of blocking elements that may have hindered execution. The third examines the adequacy of the estimated time allocated to the task, while the fourth considers the level of effort required and the complexity involved. The fifth relates to the objectivity of the description, particularly the absence of contradictions or ambiguities. The sixth invites general observations on the way the task was formulated, offering a broader assessment of writing quality. Finally, the seventh captures the overall experience of performing the task, with attention to the perceptions and feelings reported by the developer.

To provide a comprehensive view of feedback, the Project Manager has access to the Project Statistics page, which compiles several charts based on tasks marked as Done. These visualizations offer complementary perspectives on the quality of task descriptions and their evolution over time. One chart, presented in Figure 1, illustrates the distribution of tasks according to the predominant sentiment in each sprint, enabling comparison between positive and negative evaluations as well as the observation of changes across iterations.

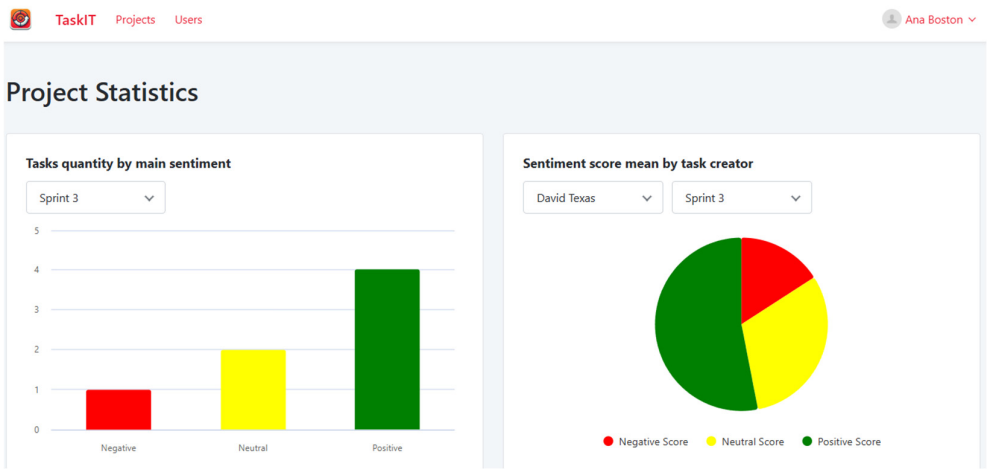

Figure 1. Distribution of tasks according to the predominant sentiment in each sprint.

A chart, organized by task creator and sprint, presents average sentiment scores for tasks authored by a specific contributor, making it possible to identify recurring tendencies in their writing (see Figure 2).

Another chart, presented in Figure 3, groups sentiment scores by task type, indicating which categories are perceived more positively in terms of clarity and quality. The longitudinal perspective is captured through a line chart, shown at the bottom of Figure 3, that tracks mean sentiment scores across sprints, thereby highlighting broader trends in task-writing practices. This feature also supports the identification of problematic patterns, which may lead the Project Manager to introduce corrective actions and subsequently evaluate their effectiveness. Finally, a chart that disaggregates sentiment scores by evaluation

topic and task creator provides a more detailed analysis of how developers assess different dimensions of task descriptions, each corresponding to predefined evaluation criteria.

Figure 2. Average sentiment scores for tasks authored by a specific contributor.

Figure 3. Sentiment scores by task type.

3.2. Semi-Structured Interviews with Team Leaders

Following the development of TaskIT and its application in several projects, interviews were conducted with a group of IT professionals to gather their perspectives. The primary objective of these interviews was to assess the extent to which the metrics generated by tools such as TaskIT are regarded as relevant and valuable for fostering the continuous improvement of task specification within project environments. Before the interview began, the purpose of the study and the procedures to be used during the interview were explained to the participant, including the type of information that would be requested and its intended use. The anonymization of the collected data was also clarified, ensuring that the participant fully understood all aspects of the project before agreeing to take part, including the associated risks, benefits, and procedures.

A total of four professionals participated in the study. Participant selection was based on expertise in software development and relevant project management experience. Two reported 14 years of experience, while the others had 10 and 16 years, respectively, resulting in an average of 13.5 years of professional experience. At the time of the interviews, all participants were employed in national or multinational IT companies and were actively engaged in software development. All had prior experience as project managers

or in leadership roles within development projects, and all reported responsibility for the specification and documentation of project tasks.

3.2.1. Interview Methodology

The research instrument consisted of a semi-structured interview protocol designed to capture project managers' perspectives on the relevance and usefulness of the TaskIT tool in supporting continuous improvement in task specification.

The protocol was organized into two sequential stages.

In the first stage, a brief presentation and demonstration introduced participants to the main functionalities of the TaskIT tool. This step was important to ensure that participants had a clear understanding of the tool's purpose and how it operated before sharing their opinions.

In the second stage, participants took part in individual semi-structured interviews comprising three open-ended questions: These questions aimed to explore: (1) potential improvements to TaskIT's feedback mechanisms; (2) additional indicators or metrics considered useful for enhancing task specification or related development activities; and (3) the current use of AI-based tools to support continuous improvement in software projects.

Following the interview, participants completed a brief questionnaire employing a five-point Likert scale to assess six aspects of the tool: (1) the adequacy of task performers as feedback providers; (2) the importance of continuous improvement in task specification; (3) the tool's capacity for rapid and automated feedback analysis; (4) the clarity of graphical outputs; (5) the usefulness of results for identifying improvements in task quality; and (6) the overall utility of the tool for supporting continuous improvement.

Each interview was conducted individually, allowing each participant to reflect freely on their professional experiences and on the role of AI-based feedback mechanisms in agile development settings.

3.2.2. Results

Regarding Question 1, participants recognized significant benefits in TaskIT, emphasizing its ability to make the feedback process trackable, agile, and automated. Suggested improvements included customizable questions, combining open and closed-ended items, adjusting feedback submission timing, and integrating TaskIT as a module within existing project management tools. One participant also proposed leveraging generative AI to evaluate task descriptions based on user feedback.

For Question 2, participants highlighted metrics such as estimated versus actual effort, task velocity, and burndown charts as the most relevant indicators for identifying areas for improvement in task execution.

Regarding Question 3, participants reported limited use of AI-based tools. One mentioned an AI system used by the human resources department, for employee feedback analysis, while two occasionally used AI to obtain secondary opinions on project-related matters. Overall, participants acknowledged the potential of AI but expressed a need to verify output due to concerns about reliability.

The results of the survey, consisting of six statements to which the participants responded, are presented in Table 3. Responses were consistently positive for all six statements.

Overall, both during the informal discussion and through survey responses, it can be stated that the project managers interviewed provided positive feedback regarding the usefulness of TaskIT.

During the informal discussion, several suggestions for additional functionalities were proposed, ranging from integration with widely used management systems to the customization of questions answered by developers. Despite these suggestions, all participants

noted that none undermined the relevance of the system, rather they would enhance its utility by expanding its functionalities.

Table 3. Survey results based on the six statements answered by participants.

	Strongly Disagree	Partially Disagree	Neither Agree nor Disagree	Partially Agree	Strongly Agree
Adequacy of task performers as feedback providers	0%	0%	0%	50%	50%
Importance of continuous improvement in task specification	0%	0%	0%	0%	100%
Tool capacity for rapid and automated feedback analysis	0%	0%	25%	25%	50%
Clarity of graphical outputs	0%	0%	25%	0%	75%
Usefulness of results for identifying improvements in task quality	0%	0%	0%	25%	75%
Overall utility of the tool for supporting continuous improvement	0%	0%	0%	25%	75%

One participant highlighted that TaskIT provides a fast and automated way to conduct the feedback process. This participant also noted that they had never seen a solution of this kind specifically aimed at promoting continuous improvement in software development activities, in this case, task specification. However, they mentioned that their organization uses dedicated software for employee feedback within the human resources department, which they consider highly useful. Two participants indicated that TaskIT's core concept would be particularly valuable in projects where most professionals have limited experience, such as in task specification. Another participant emphasized the tool's potential benefits for remote teams, where limited face-to-face interaction makes feedback harder to obtain and often less reliable. According to this participant, TaskIT could provide teams with a dedicated platform for feedback, encouraging more regular and precise communication. From the perspective of those analyzing feedback, the participant noted that TaskIT offers a central point of access, allowing the monitoring of feedback evolution over time.

In summary, the results from both the interviews and the survey indicate that the professionals consulted view TaskIT's contribution to promoting continuous improvement in task specification positively and as useful. Additionally, several suggestions for new functionalities or adjustments to existing features were identified, which, according to participants, do not undermine TaskIT's core concept.

4. Discussion and Final Remarks

The results of this study reinforce the importance of combining structured feedback practices with technological support to improve software development processes in agile contexts. Feedback has long been recognized as a central element in continuous improvement, yet conventional approaches, such as retrospective meetings or surveys, often suffer from limited frequency and subjectivity. The evaluation of TaskIT suggests that integrating sentiment analysis into daily workflows can mitigate these shortcomings, providing timely and systematic insights into the quality of task specifications.

From the interviews conducted with experienced team leaders, three aspects stand out. First, the consensus that developers are the most appropriate providers of feedback on task descriptions highlights the value of capturing perceptions directly from those responsible for execution. Second, participants valued the clarity of TaskIT visual outputs and the

automation of sentiment analysis, suggesting that such tools can complement retrospective discussions by making feedback more tangible and accessible. Third, the suggestions for integration with existing project management systems and for customization of evaluation questions underline the practical need for flexibility and adaptability if such tools are to be widely adopted in professional settings.

These findings converge with previous studies on the role of AI in project management, where automation has been shown to reduce dependency on subjective judgment and to improve the objectivity of decision-making. At the same time, the feedback from participants also reveals some reservations, particularly concerning the reliability of AI-generated results. This reflects a broader challenge identified in the literature: the need for transparency and explainability in AI-based tools to foster trust among professionals. Another limitation to consider is the potential for response fatigue when feedback is collected too frequently, which may affect both the quality of responses and the willingness of developers to engage.

To further contextualize these findings, Table 4 presents a comparison between our results and other studies discussed in the literature review. While prior research, such as [9,20], primarily focused on automating quantitative aspects of project management, such as effort estimation, task prioritization, or resource allocation, our study emphasizes the added value of addressing the human dimension behind these processes by incorporating qualitative and affective insights drawn directly from developers' perceptions. This broader analytical perspective enables project managers to complement traditional performance indicators with continuous sentiment insights, helping them understand not only what teams deliver but also how they feel about the work they do. Similarly, ref. [12] explored data-driven retrospectives to analyze project metrics, but their approach did not explicitly capture the emotional tone or perceived clarity of tasks. TaskIT extends these ideas by combining structured feedback with sentiment analysis, offering a balanced view that integrates both technical and human-centered aspects of continuous improvement.

Table 4. Comparative summary of related studies and current approach.

Study	Focus of the Research	Reported Advantages	Relation to the Present Work
[9]	Story-point classification using graph neural networks	Improved accuracy in effort estimation	TaskIT similarly automates interpretation but targets qualitative feedback rather than estimation metrics
[10]	NLP for user-story quality assessment	Improved quality of use case content	TaskIT extends this principle to operational task descriptions, validated by practitioner interviews
[12]	Forecast workload capacity	Improved forecasting	TaskIT complements these analytics by integrating sentiment data for richer team reflections
[20]	AI-based decision assistant for task assignment	Objective, data-driven decision-making	TaskIT adds an AI-based human-centered feedback layer, assessing how task definitions affect developer experience
Present Study (TaskIT)	Sentiment analysis of developer feedback	Real-time, qualitative insight; supports transparency and collaboration	TaskIT bridges human-centered feedback and AI-driven automation within agile feedback loops

These comparisons highlight that the contribution of this study lies not only in automating aspects of project management but also in augmenting the interpretative capacity of agile teams through continuous, emotion-aware feedback. Bringing together quantitative and qualitative insights offers a valuable complement to existing AI-based approaches, widening the scope of continuous improvement beyond operational efficiency to also include team well-being, engagement, and communication quality.

Taken together, the results suggest that the approach implemented in TaskIT can serve as a bridge between traditional, human-centered practices of agile development and the growing reliance on data-driven methods. By making developers' experiences more visible while providing managers with structured analyses, the tool contributes not only to greater task clarity but also to enhanced motivation and collaboration. In this sense, its value extends beyond operational efficiency, supporting a more inclusive and participatory model of continuous improvement.

Overall, the study shows that the integration of sentiment analysis into agile practices is feasible and potentially beneficial. TaskIT demonstrates that systematic collection and interpretation of feedback can make continuous improvement more concrete, offering a path for organizations to align technical objectives with the day-to-day experiences of their development teams.

4.1. Future Work

Future work could extend this research in several directions. A longitudinal evaluation of TaskIT across multiple organizations and team sizes would allow assessing its impact over time on metrics such as productivity, developer satisfaction, and task clarity. Exploring the use of the potential of generative AI in analyzing and suggesting improvements to task descriptions, while addressing issues of bias and oversight that such technologies may introduce. Another promising direction involves integrating TaskIT, or similar approaches, into popular project management tools to enable seamless adoption and continuous data collection.

In terms of use cases, sentiment-aware feedback systems could benefit distributed and remote teams, where communication barriers hinder transparency; educational contexts, helping software-engineering students reflect on their teamwork experiences; and organizations pursuing continuous delivery pipelines, where rapid feedback is essential for maintaining alignment between technical goals and team well-being.

4.2. Limitations and Threats to Validity

As with most exploratory investigations, this study presents certain limitations that warrant consideration. It constitutes an initial effort to demonstrate the feasibility and practical relevance of the proposed approach and, consequently, involved a relatively small group of project managers. This naturally constrains the diversity of perspectives represented, a typical characteristic of early-stage empirical research. Future work encompassing larger and more heterogeneous samples would enable a broader and more representative understanding of how tools such as TaskIT perform across distinct organizational and cultural contexts. A further limitation concerns the data collection process. Since the information was gathered through interviews and self-reported surveys, participants' responses may have been influenced by personal perceptions or varying degrees of familiarity with AI-based tools, rather than reflecting entirely objective assessments. In addition, the evaluation was carried out in a controlled setting and did not include longitudinal monitoring of its long-term effects on collaboration, productivity, or team dynamics. Finally, because sentiment analysis relies on pretrained language models, potential biases inherent in these algorithms may affect the interpretation of developers' feedback. Acknowledging these

factors is important for guiding future refinements and for ensuring that subsequent studies can validate and extend the present findings in real-world environments.

Author Contributions: Conceptualization, D.M., J.M. and F.R.; methodology, D.M., J.M. and F.R.; investigation, D.M., J.M. and F.R.; writing—original draft preparation, D.M., J.M. and F.R.; writing—review and editing, D.M., J.M. and F.R.; supervision, J.M. and F.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

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

References

1. Macedo, H.I.L. *Development of a Continuous Improvement Process for Agile Software Development Teams*; Universidade do Minho: Braga, Portugal, 2019.
2. Compreender a Melhoria Contínua. Available online: https://www.kaizen.com/pt/insights-pt/melhoria-continua-excelencia-operacional/ (accessed on 6 February 2025).
3. Understanding Continuous Improvement in Software Development. Available online: https://www.teamhub.com/blog/understanding-continuous-improvement-in-software-development/ (accessed on 6 February 2025).
4. What Is Continuous Improvement: Tools and Methodologies. Available online: https://www.atlassian.com/agile/project-management/continuous-improvement (accessed on 6 February 2025).
5. O Que é Análise de Sentimento? AWS. Available online: https://www.aws.amazon.com/what-is/sentiment-analysis/ (accessed on 19 January 2025).
6. What Is Sentiment Analysis? | IBM. Available online: https://www.ibm.com/think/topics/sentiment-analysis (accessed on 19 January 2025).
7. What Is Sentiment Analysis? GeeksforGeeks: Noida, India. 2025. Available online: https://www.geeksforgeeks.org/what-is-sentiment-analysis/ (accessed on 19 January 2025).
8. Izhar, R.; Cosh, K.; Bhatti, S.N. Enhancing Agile Software Development: A Novel Approach to Automated Requirements Prioritization. In Proceedings of the 21st International Joint Conference on Computer Science and Software Engineering (JCSSE), Phuket, Thailand, 19–22 June 2024; pp. 286–293.
9. Phan, H.; Jannesari, A. Story Point Level Classification by Text Level Graph Neural Network. In Proceedings of the 1st International Workshop on Natural Language-Based Software Engineering (NLBSE), Pittsburgh, PA, USA, 8 May 2022; pp. 75–78.
10. Jiménez, S.; Alanis, A.; Beltrán, C.; Juárez-Ramírez, R.; Ramírez-Noriega, A.; Tona, C. USQA: A User Story Quality Analyzer Prototype for Supporting Software Engineering Students. *Comput. Appl. Eng. Educ.* **2023**, *31*, 1014–1024. [CrossRef]
11. Tarawneh, M.; AbdAlwahed, H.; AlZyoud, F. Innovating Project Management: AI Applications for Success Prediction and Resource Optimization. In *Lecture Notes in Networks and Systems*; Springer: Cham, Switzerland, 2024; Volume 956 LNNS, pp. 382–391.
12. Sandoval-Alfaro, O.E.; Quintero-Meza, R.R. Application of Data Analytics Techniques for Decision Making in the Retrospective Stage of the Agile Scrum Methodology. In Proceedings of the 2021 Mexican International Conference on Computer Science (ENC), Morelia, Mexico, 9–11 August 2021.
13. Page, M.J.; McKenzie, J.E.; Bossuyt, P.M.; Boutron, I.; Hoffmann, T.C.; Mulrow, C.D.; Shamseer, L.; Tetzlaff, J.M.; Akl, E.A.; Brennan, S.E.; et al. The PRISMA 2020 Statement: An Updated Guideline for Reporting Systematic Reviews. *BMJ* **2021**, *372*, n71. [CrossRef] [PubMed]
14. Matthies, C. Feedback in Scrum: Data-Informed Retrospectives. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Montreal, QC, Canada, 25–31 May 2019.
15. Ribeiro, A.B.C.; Alves, C.F. A Survey Research on Feedback Practices in Agile Software Development Teams. In Proceedings of the ACM International Conference Proceeding Series, Virtual Event Brazil, 8–11 November 2021; Association for Computing Machinery: New York, NY, USA, 2022.

16. Morales-Trujillo, M.; Galster, M. On Evidence-Based Feedback Practices in Software Engineering for Continuous People Improvement. In Proceedings of the Software Engineering Education Conference, Tokyo, Japan, 7–9 August 2023; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2023; Volume 2023-August, pp. 158–162.
17. Siebra, C.; Sodre, L.; Quintino, J.; Da Silva, F.Q.B.; Santos, A.L.M. Collaborative Feedback and Its Effects on Software Teams. *IEEE Softw.* **2020**, *37*, 85–93. [CrossRef]
18. Ribeiro, A.B.C.; Alves, C.F. Feedback in Virtual Software Development Teams. In Proceedings of the Iberian Conference on Information Systems and Technologies (CISTI), Aveiro, Portugal, 20–23 June 2023; IEEE Computer Society: Washington, DC, USA, 2023; Volume 2023-June.
19. Eren, K.K.; Ozbey, C.; Eken, B.; Tosun, A. Customer Requests Matter: Early Stage Software Effort Estimation Using k-Grams. In Proceedings of the Proceedings of the ACM Symposium on Applied Computing, Brno, Czechia, 30 March–3 April 2020; pp. 1540–1547.
20. Ebrahim, E.; Sayed, M.; Youssef, M.; Essam, H.; El-Fattah, S.A.; Ashraf, D.; Magdy, O.; Eladawi, R. AI Decision Assistant ChatBot for Software Release Planning and Optimized Resource Allocation. In Proceedings of the Proceedings—5th IEEE International Conference on Artificial Intelligence Testing (AITest), Athens, Greece, 17–20 July 2023; pp. 55–60.
21. Feng, Y.; Liu, Q.; Dou, M.; Liu, J.; Chen, Z. Mubug: A Mobile Service for Rapid Bug Tracking. *Sci. China Inf. Sci.* **2016**, *59*, 1–5. [CrossRef]
22. Sarwar, H.; Rahman, M. A Systematic Short Review of Machine Learning and Artificial Intelligence Integration in Current Project Management Techniques. In Proceedings of the 2024 4th IEEE International Conference on Software Engineering and Artificial Intelligence (SEAI), Xiamen, China, 21–23 June 2024; pp. 262–270.
23. Mohammad, A.; Chirchir, B. Challenges of Integrating Artificial Intelligence in Software Project Planning: A Systematic Literature Review. *Digital* **2024**, *4*, 555–571. [CrossRef]