Adoption of the GITHUB platform as a data source for creating a decision support system in an agile development context

AUTHORS:

Adriana Lópes Fernándes
afernandes@autonoma.pt
Universidade Autónoma de Lisboa

António Cabeças
acabecas@autonoma.pt
Universidade Autónoma de Lisboa

Laercio Cruvinel Júnior
lcruvinel@autonoma.pt
Universidade Autónoma de Lisboa

Bruna Amorim dos Santos
bassm1@iscte-iul.pt
ISCTE- University Institute of Lisbon

ABSTRACT:

The increasing adoption of agile methodologies in software development by companies aims to enhance productivity, quality, and responsiveness to market changes. This scenario demands tools capable of reliably monitoring the development process through easily visualisable metrics, facilitating analysis and decision-making. The present study aimed to develop a decision support system that provides essential metrics for managing agile teams, utilising data from GitHub repositories. The methodology comprised three main phases: (1) initial requirements gathering through interviews with agile software development experts; (2) analysis of responses to define a preliminary list of requirements; and (3) validation and finalisation of the tool's functional requirements through a brainstorming session. The study identified key metrics used by managers in conducting agile projects. The resulting tool was developed and tested, proving to be intuitive and easily analysable for end users. It provides fundamental metrics and assists teams in optimising their products and work processes, contributing significantly to the effectiveness of agile software project management.

KEY WORDS: Agile Methodology, Agile Metrics, Agile Tools, GitHub.

INTRODUCTION

Systems of information (SI) play a crucial role in contemporary society, permeating virtually all daily activities, from residences to institutions such as schools and hospitals. To keep pace with the growing complexity and demand for SI, as well as rapid market changes, the field of software engineering has evolved significantly, incorporating new theories, approaches and methods for their development (Fernandes & Machado, 2017). Agile development approaches have been adopted by companies aiming to accelerate the launch of new products, improve quality and productivity, respond more quickly to changes and unexpected events, as well as enhance integration between people, project management and task monitoring (Venkatesh et al., 2020; Fernández-Diego et al., 2020, Beh et al., 2022).

To achieve the objectives of this approach, the software development process has been analysed and monitored through metrics considered agile. These provide concrete data on the status of task development, product quality and team productivity, offering visibility on the health of the company's development process (Budacu & Pocatili, 2018). Agile metrics allow for projecting delivery deadlines and making more assertive decisions, as well as promoting continuous improvement actions for agile software development teams (Albino, 2017; Choraś et al., 2020; Beh et al., 2022).

A variety of agile metrics are available. The appropriate choice requires a clear understanding of the intended objectives, so as to generate the expected benefits without compromising the performance of the process or product under development (Hazzan & Dubinsky, 2007; Budacu & Pocatili, 2018). Currently, various tools assist managers and project teams in monitoring work through agile metrics (Budacu & Pocatilu, 2018). However, many still present deficiencies in monitoring the situation and progress, making estimates, and tracking effort, quality and delivery time. These difficulties stem from the dispersion of information across different sources, resulting in complexity in data consolidation and, primarily, in decision-making (Budacu & Pocatilu, 2018; López et al., 2022).

GitHub has stood out as a widely used tool, assisting managers and development teams in efficiently managing agile projects (Ortu et al., 2020). As a repository hosting free versions of project code, GitHub allows collaboration in independent flows, review of work progress in development, visualisation of team advancement and documentation of errors (Beer, 2018). However, as it does not provide all the necessary information for project and team management, GitHub is often used in conjunction with other tools such as Jira Software, Microsoft Sharepoint, Broadcom Software, Kanbanize and GitLab.

This study proposes the development of a Decision Support System, designed to extract and analyse metrics essential for the effective management of agile

teams, using data from GitHub. The main objective is to optimise the decision-making process, providing valuable insights and automating the consolidation of critical information for monitoring agile projects. This tool aims to empower managers and development teams, allowing them to focus more on strategies and less on manual data collection, resulting in more efficient management and a significant increase in project productivity.

This research aims to deepen the understanding of crucial metrics employed by managers and teams in agile software development projects. The study proposes an innovative solution that optimises the collection and organisation of project data, centralising this information on a unified platform. This approach significantly facilitates the analysis and visualisation of results, allowing for more precise and well-founded decision-making. As a direct benefit to organisations, the proposed system promotes operational efficiency through the automation of data collection processes, greater precision in evaluating project and team performance, and proactive identification of bottlenecks and areas for improvement. Additionally, the solution offers support for continuous improvement of software development processes and promotes strategic alignment between project objectives and organisational goals. By providing actionable and real-time insights, this solution empowers organisations to quickly adapt their strategies, optimise resources and drive innovation in software development, resulting in a significant positive impact on the effectiveness and competitiveness of companies in the technology market.

CONCEPTUAL FRAMEWORK

The agile approach encompasses a set of practices and techniques developed to address common project management challenges and streamline processes. Rather than being a mere set of tools or a simple methodology, it represents a philosophy centred on effectiveness and flexibility (Islam, 2013; El Sheikh & Alnoukari, 2012).

The demand for agile methodologies surged between 1998 and 2002, marking the most productive period for their development and implementation (Abrahamsson et al., 2008). The Agile Manifesto for software development, penned in 2001, catalysed the emergence of numerous agile methods (El Sheikh & Alnoukari, 2012; Beh et al., 2022). This manifesto advocated principles such as rapid production of working code, frequent incremental changes, continuous customer collaboration, use of quality-improving metrics, test-driven development, pair programming, and code refactoring (Vijayasarathy & Turk, 2012; Hazzan & Dubinsky, 2007).

Of particular relevance to this study is the principle of software development metrics. These metrics emerged to provide tangible data for analysing and evaluating the construction process (Fenton & Bieman, 2014; Eisty et al., 2018,

Choraś et al., 2020), facilitating continuous improvement (Ram et al., 2019). They enable project progress tracking, estimation and planning, algorithm complexity evaluation, and assessment of software effectiveness and safety. Furthermore, they aid in understanding business goals, enhancing communication, and boosting team motivation (Suresh et al., 2012; Kupiainen et al., 2015; Ram et al., 2019; Fernández-Diego et al., 2020).

The selection of appropriate metrics is crucial after defining the project's objectives and understanding the problems to be solved by the information system. Eisty et al. (2018) categorise software development metrics into: 1. Code metrics: measuring complexity (e.g., McCabe, LCOM4) and other code characteristics (e.g., Defect Density). 2. Process metrics: collected over extended periods to provide insights into the development process (e.g., Productivity, Cycle Time, Number of Commits). 3. Test metrics: monitoring testing activities and providing insights into progress, productivity, and quality (e.g., Code Coverage, Test Coverage). 4. General quality metrics: related to desirable software properties (e.g., Interoperability, Portability, Sustainability). 5. Execution performance metrics: addressing runtime, storage, or scalability on high-performance computing platforms. 6. Recognition metrics: quantifying external interest in a project (e.g., Citations, Downloads).

Kupiainen et al. (2015) offer an alternative classification: 1. Sprint and project planning metrics: Including activity prioritisation, effort estimates (e.g., Use Case Points), project scope definition (e.g., Velocity), and development resources and flexibility (e.g., Team Effectiveness). 2. Process problem identification metrics: Helping understand and rectify issues in software engineering processes (e.g., Value Stream Maps). 3. Employee motivation metrics: Encouraging swift problem response (e.g., Build Status). 4. Project progress monitoring metrics: Tracking project evolution (e.g., Burndown), simplifying development aspects, increasing stakeholder visibility (e.g., Cost Type, Cycle Time), assessing project objective achievability (e.g., Lead Time), and adapting workflow to prevent overload (e.g., Work in Progress).

In conclusion, the judicious selection and application of these metrics are integral to the successful implementation of agile methodologies in software development projects. They provide valuable insights that enable teams to optimise their processes, enhance product quality, and respond effectively to the dynamic demands of the software development landscape.

Agile Metrics Applied in this Study

To monitor software development efficiency, enhance task delivery predictability, assess team productivity, and ultimately improve product quality (Albino, 2017), this study examined five key metrics.

Lead Time (LT) measures the duration from task initiation to result delivery. It encompasses the entire process, from concept to launch or order to delivery, providing a comprehensive view of project timelines (Niemi et al., 2021; Kupiainen et al., 2015). Another metric applied was Cycle Time focuses on the period between actual work commencement and task completion. As a critical indicator of process efficiency, it directly impacts production rates (Budacu & Pocatilu, 2018).

The Cumulative Flow Diagram (CFD) visually represents work evolution, highlighting obstacles and potential instabilities. It serves as an essential tool for tracking progress, forecasting task completion, and identifying workflow intervention needs (Caroli, 2020).

Work in Progress (WIP) quantifies the number of ongoing work items. Limiting WIP is crucial for enhancing work efficiency and capacity management. Reduced WIP leads to improved liquidity, cash flow, customer service, and decreased business risks (Hemalatha et al., 2021). Finally, the Throughput measures the number of items delivered within a specified timeframe, enabling teams to gauge their delivery capacity (Caroli, 2020).

These metrics, when utilised in conjunction, provide a comprehensive overview of the software development process. They offer valuable insights into team performance, workflow efficiency, and potential areas for improvement, thereby facilitating data-driven decision-making and continuous process optimisation in agile software development environments.

GitHub Platform

Version control systems are integral to numerous software engineering methodologies, enabling source code storage and change management. Git, predominantly utilised by open-source software development companies, is one such system. Platforms like GitHub offer free hosting for Git repositories (Ortu et al., 2020). GitHub, recognising Git's potential, built a web service layer atop its functionalities (Dawson & Straub, 2016). Beyond repository storage, GitHub facilitates collaboration on independent workflows, work-in-progress reviews, team progress visualisation, and documentation of errors and new features (Beer, 2018).

The proliferation of open-source software development (OSS) tools has led to yearly growth in users and projects. GitHub, the largest collaboration platform, hosts over a million OSS project repositories (Subramanian et al., 2022). It manages projects, issues, and changes, providing valuable metrics for addressing various software engineering challenges (Şeker et al., 2021).

Developer social platforms like GitHub are crucial for collaborative software

development. They offer spaces for project showcasing, source code sharing, collaboration seeking, and issue resolution. The ethos of sharing source code online aligns with the principle of making information freely available for community use. The content generated on these platforms is heavily reliant on community participation (Bhasin et al., 2021).

These platforms have revolutionised software development, fostering a global community of developers and promoting open collaboration, knowledge sharing, and continuous improvement in software engineering practices.

GitHub Adoption Benefits

It is possible to identify significant benefits on adopting GitHub as a data source for agile development, and we may refer the following advantages, namely, (i) GitHub integrates with other tools used by agile teams, such as ZenHub for managing agile epics and boards within GitHub itself. This integration reduces the need for switching between different platforms, improving productivity, and keeping project data centralized; (ii) GitHub usage allows to strength collaboration through its own features, that are aligned with agile practices. Teams can manage their workflows, from task assignment to continuous integration and deployment, making easier to track progress and to solve issues; (iii) GitHub supports automation tools and continuous delivery pipelines, essential for agile development. Teams can automate repetitive tasks, such as testing and deployments, allowing for more frequent and reliable releases; (iv) The platform allows a version control and history features ensure that all changes are well-documented relevant for continuous improvement and helps to maintain a single source of information for all team members. This is truly important for large projects with multiple contributors, as it helps to maintain a clear history of changes and avoids troubleshooting by identifying when and where issues were introduced (GitHup, 2024; GitHub Resources, 2024; InfoQ, 2024).

METHODOLOGY

Sample and data collection

This study was conducted in three phases to achieve its objectives. Initially, online interviews were carried out with various professionals involved in agile software development projects. Their responses were analysed to identify an initial list of requirements. Secondly, a brainstorming session was held with five selected respondents from the original thirty. This session aimed to refine ideas, validate requirements, and model the tool collaboratively. Finally, the developed tool underwent testing and evaluation by the same five participants who contributed to the brainstorming session. This methodical approach ensured a comprehensive development process, aligning the tool with user needs and

industry best practices in agile software development.

Online interviews

The initial data collection was carried out through an interview consisting of 15 semi-structured questions, developed from the literature review, about the tools and metrics used to manage projects. This type of data collection was considered the most appropriate when the goal is requirement gathering, as it is an easy way to reach a large number of people and obtain their views on various aspects of a system (Mishra & Mohanty, 2011).

Data collection began on 01/02/2022. The interview was answered by 30 employees from private software development companies working on agile projects. They had different roles and responsibilities in the projects and different levels of education (Table 1).

TABLE 1. Respondents profile

#	Position	Functions performed in agile projects
4	Business Analyst	Business analysis and documentation of business needs and requirements / Management of the product backlog, supporting the Product Owner and development team / Process modelling.
2	Application Support Engineer	Investigate and solve customer problems in production environments / manage and administer multiple applications.
3	Tester Analyst	Design test scenarios / perform manual and automated testing / report defects.
3	Product Manager	Release management, collaborating with stakeholders and the team to define and plan releases, maintaining the product roadmap, monitoring the product's progress, and defining and tracking product metrics.
6	Agile Master	Train and guide development teams / facilitate agile meetings: backlog refinement, planning, review, and retrospective / assist teams in problem-solving / monitor product OKRs.
7	Developer	Develop software solutions.
2	Functional Analyst	Support the Product Owner in managing and refining the Backlog / Managing the flow of deliveries / Specification of requirements with business areas, elaboration of User Stories and validation of them with Customers / Process modelling.
1	Infrastructure Consultant	Infrastructure Audit Monitoring / Information Technology Risk Analysis and Monitoring in Infrastructure / Creating and Adapting Infrastructure Solutions / Integration and User Testing.
2	Project Manager	Tasks, issue and problem management / Monitoring team progress and performance and contributing to improvement / Project process and result tracking / Tracking team metrics / Organizing and facilitating Scrum ceremonies.

The companies where the respondents worked come from a variety of areas, such as public management, healthcare sector, road transport management, vehicle industry, banking sector, airline company, and energy distribution sector.

Brainstorming Session

Following the analysis of interview data, which provided insights into user needs and problems, a two-stage brainstorming session was conducted. The first stage fostered group creativity, encouraging spontaneous idea-sharing among participants to develop practical solutions (Al-Samarraie & Hurmuzan, 2018). The second stage, adhering to Chemuturi's (2012) methodology, involved classifying and grouping ideas into system functionalities and software quality attributes (functional and non-functional). Participants then prioritised the most crucial ideas, culminating in a final list of tool requirements.

From the 30 interview respondents, five were selected for this session: a developer, an agile master, a project leader, a project manager, and a functional analyst. This selection was based on their expertise in agile approaches and the understanding that smaller groups tend to be more productive in software sessions (Laplante & Kassab, 2022). The session concluded with validated functional and non-functional requirements, detailing the tool's desired characteristics and functionalities. Subsequently, the tool was modelled using the Unified Modelling Language (UML), a visual language model for system requirements. UML is versatile, capable of describing designs, representing implementation details, and applicable across requirements analysis, systemic analysis, implementation, and testing phases (Keng & Terry, 2000).

This comprehensive approach ensured that the tool's development was firmly rooted in user needs and industry best practices, while maintaining a clear focus on both functional and non-functional aspects of the software.

Testing and evaluation of the tool

The tool evaluation was conducted by the same users who participated in the brainstorming session. The process was divided into two parts: firstly, individual 30-minute meetings were held with each selected user to present the tool, review requirements, and explain the evaluation objectives. Subsequently, the five users were given free rein to test the application.

In the second stage, these users performed tests and evaluation. The evaluation employed a questionnaire based on the product quality model defined in ISO/IEC 25010, which establishes eight quality characteristics. For this tool, five criteria were evaluated: 1. Usability: the degree to which users can achieve specific goals effectively, efficiently, and satisfactorily (Sarrab & Rehman, 2014). 2. Performance efficiency: performance relative to resource usage under established conditions. 3. Reliability: the system's ability to perform functions under specified conditions for a determined period. 4. Security: the degree of protection for information and data, ensuring appropriate access based on

authorisation levels. 5. Functional suitability: the degree to which the product provides functions meeting declared and implicit needs under specified conditions (ISO/IEC 25010, 2022).

The questionnaire was developed drawing from existing models such as the Ergolist (LABIUTIL, 2018), Web Usability Checklist (Patkai, 2022), and 25-point Website Usability Checklist (User Effect, 2009). Of the 21 questions, five evaluated usability, three assessed performance efficiency, two each for reliability and security, eight for functional suitability, and one open-ended question for comments or suggestions. All quality characteristic questions were structured using a five-point Likert scale (1 - Strongly Disagree, 5 - Strongly Agree).

RESULTS AND DISCUSSION

Agile metrics identification

Initially, in the interview, respondents were asked about the tools they use. Four respondents use only Jira Software for project and task management. The other respondents use various tools, including Jira Software along with Microsoft Sharepoint, an internally developed tool called AgileWork ; Broadcom Software; Microsoft Planner ; GitLab; Microsoft Excel with Power BI ; Kanbanize; and Microsoft Sharepoint. The majority of respondents (n=18) found the tools easy to use, however, 8 respondents stated that the tool they use is not useful for their team on a daily basis.

To understand the scenario of agile metrics adopted in companies, respondents were asked about which metrics they use to track project and software development team performance. The results showed that 15 respondents reported using Lead Time and Cycle Time, 9 reported WIP (Work in Progress), BurnDown, and BurnUp, and 6 reported Throughput and CFD (Cumulative Flow Diagram).

Most respondents (n=22) stated that the metrics they use are visualized in the tool itself. The remaining respondents reported visualizing them through Microsoft Excel, Power BI, or even not using a tool for visualization. However, 17 respondents consider the available metrics in the chosen tools to be insufficient for planning, estimating, and tracking projects and team tasks.

Since most respondents do not find the available metrics in their tools sufficient to assist in planning, estimating, and tracking projects and tasks, we sought to know which agile metrics would be important among those used. Respondents could choose more than one option, and the results showed that WIP was

considered the most important metric (n=25), followed by Lead Time and Throughput (n=22), Cycle Time (n=18), and finally, Cumulative Flow Diagram (CFD, n=14). To obtain information for the defined agile metrics, the GitHub Rest API was used through the GET request method.

To identify the functionalities of the agile metrics visualization tool, respondents were asked about the positive and negative points of the tools currently used. Regarding the most valued positive points, easy visualization, easy tracking of tasks/projects, data export, generation of graphs, automation, field customization, easy use (friendly), search functionality, presentation of various metrics, and intuitiveness were mentioned. Regarding the negative points, the most mentioned were difficulty in programming and customization, very steep learning curve, few graphs, and difficulty in understanding the current state of the project.

Subsequently, respondents were asked to identify the most important characteristics for a project and task management tool. Among the answers, the tool should allow real-time tracking of task progress, be adaptable to any work process, and allow clear information extraction. It should also generate reports with the most used metrics (mentioned earlier), be able to provide reports and graphs that facilitate analysis and decision-making and be an easy-to-use tool with the possibility of introducing filters and searches.

After learning about the tools used by each respondent, it was found that the main problem is the lack of information to assist in planning, estimating, and tracking projects and tasks, as more than half do not have sufficient data, do not display clear estimates, project graphs, and some necessary metrics. For these reasons, almost half of the respondents (n=14) opt for other means and tools to meet their needs, putting aside the official tool.

Brainstorming session result

Given the results and understanding of the needs of the respondents, a brainstorming session was scheduled to foster the collaboration of ideas and validate and define the final requirements of the tool. The session lasted approximately two hours and was divided into two parts. In the first part, which lasted an hour and twenty minutes, the respondents had the freedom to present their ideas and suggestions and discuss other contributions to the agile metrics visualization tool. In the second part, each participant selected three ideas they deemed important. The best ideas, considered by all present, were prioritized to define the final requirements. To assist the session, the free tool Miro was used, which made collaboration of ideas faster, easier, and more transparent. The most important features for a project and task tracking tool in agile software development teams are presented in Table 2.

TABLE 2. Main features for the tool

Characteristic	Description
Intuitive	Ease of use for users to navigate the system and make the most of its functionalities.
Flexible	Allow viewing different projects, filtering by different periods, and working with different data.
Transparent	The system should prioritize transparency so that everyone can consult the results.
Relevant metrics	Display only relevant metrics to assist in monitoring tasks, projects, and decision-making.
Up-to-date information	Access data in real-time.
Reliable	Information and metric calculations must be correct.
Secure	The system should allow authenticated users to view only the information they have access to.

Based on the definition of the presented characteristics, the specification of requirements was initiated, which is considered a critical step for the success of a system, since it defines the objectives and functions that need to be performed, what needs to be built, and its constraints (Sage & Rouse, 2014). Tables 3 and 4 present, respectively, the determined functional and non-functional requirements that should exist in the tool to meet the users' needs.

TABLE 3. Functional Requirements

FR#	Features
FR01	The system must be configured by the user.
FR02	The system must validate the user by access token.
FR03	The system must display the projects linked to the user and allow selecting the project for metrics viewing.
FR04	The system must allow viewing graphs by day, week, or month and filter by period.
FR05	The system must display the WIP (Work In Progress) metric, which consists of the number of tasks in progress within the specified period.
FR06	The system must display the number of tasks for each project phase within the specified period.
FR07	The system must display the Throughput metric, that is, the number of tasks completed within the specified period.
FR08	The system must display the time it took for a task to be developed, that is, the time it stayed "in progress" until its completion (Cycle Time metric), according to the specified period.
FR09	The system must present a Cumulative Flow Diagram (CFD) graph of tasks that go through the workflow and distinguish the stages.
FR10	The average Lead Time of tasks must be displayed within the selected period.
FR11	The system must present a Cumulative Flow Diagram (CFD) graph of tasks that go through the workflow and distinguish the stages.

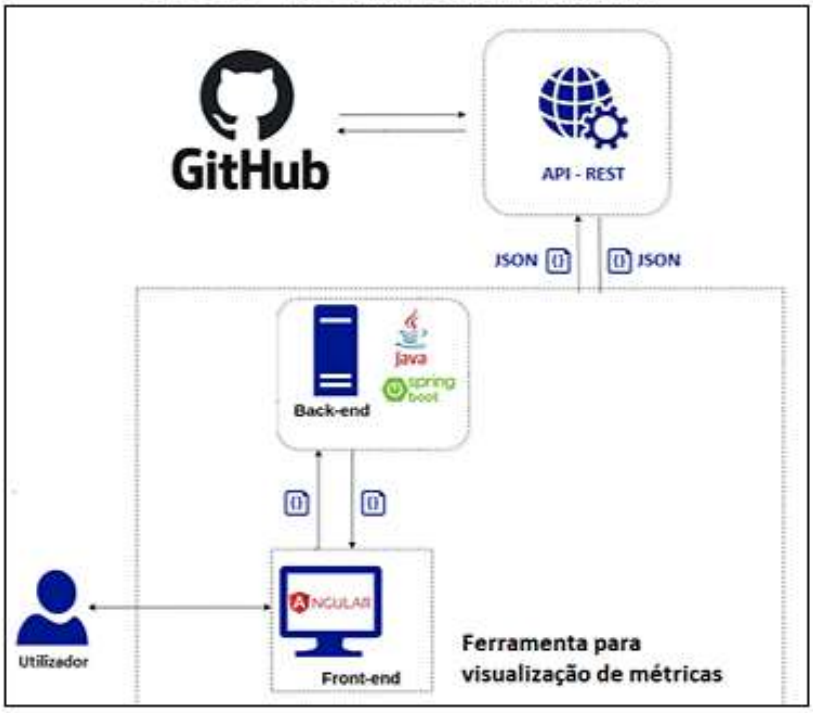
TABLE 4. Non-Functional Requirements

NFR#	Features
NFR01	The system must be web-based.
NFR02	The system must be compatible with the Google Chrome browser.
NFR03	The system must be integrated with the GitHub platform, using the GitHub Rest API v3.
NFR04	The must be developed in Java and use the Angular framework.
NFR05	The system must collect GitHub information in real-time.

Tool general architecture

First of all, Figure 1 illustrates the general architecture of the tool. It is a web application built entirely with open-source technologies. For the front-end, the Angular framework based on JavaScript was used to construct the application interface with the user. For the back end, the Java programming language was used along with Spring Boot (Java framework) which facilitates and optimizes the development time of Java applications.

FIGURE 1. General tool architecture



Communication with the GitHub platform for data collection was done through the GitHub Rest API v3. When the user requests the creation of metrics, a request is made to the API which returns the data to a JSON structure. Subsequently, the tool interprets and processes this data for the calculation and display of real-time information.

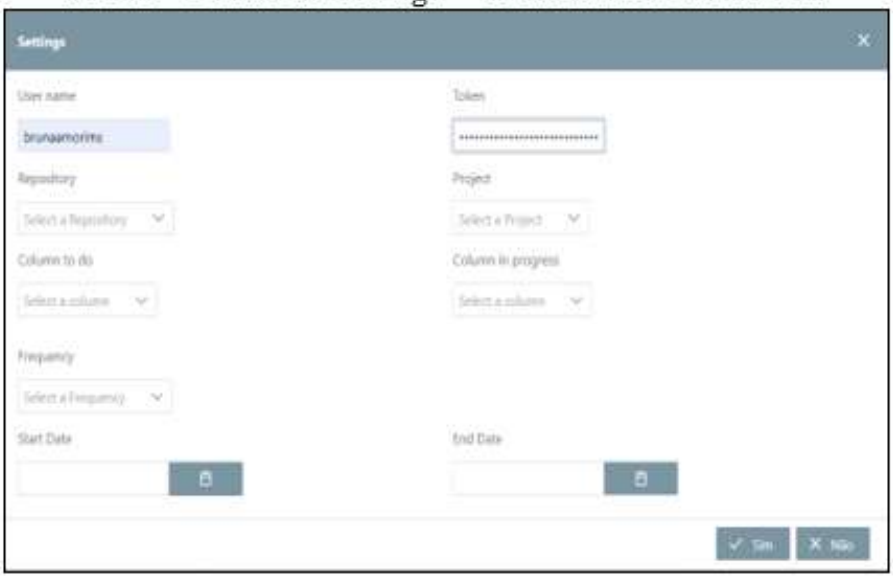
Visual Studio Code was used to develop the front-end of the tool and the Eclipse IDE for the back end. The analysis and testing of the GitHub Rest API was performed with the help of the Postman application, which allows the execution of requests in any API.

Prototypes

In this section, we present the usage flow of the tool, that was named GIRO, with images of the developed functionalities.

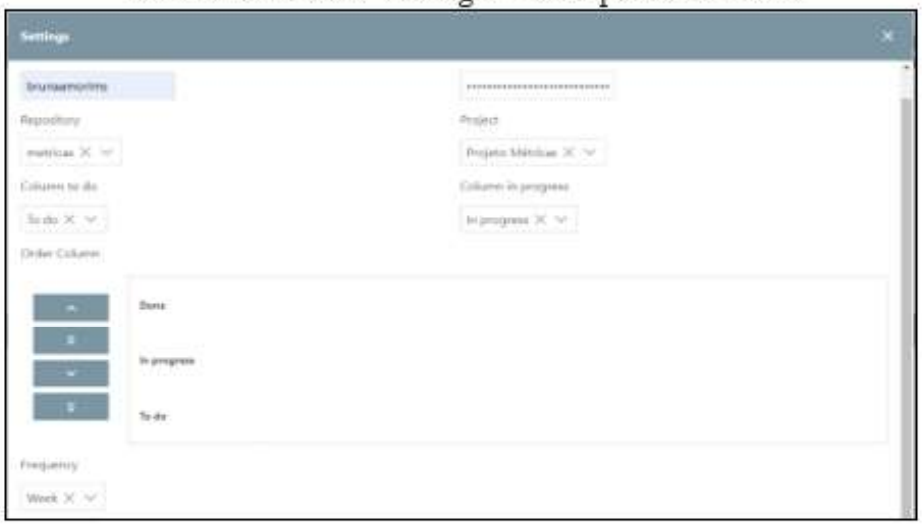
1. Enter the user name and token for authentication in the tool (Figure 2)

FIGURE 2. Screen "Settings" – Authenticate GitHub user



2. After the user authentication, they should select the repository and project they wish to view metrics from, the viewing frequency (day, week or month), enter the column on the board that corresponds to tasks to do and tasks in progress, and indicate the order of the columns (finished tasks column/ tasks in progress column/ tasks to do column; Figure 3).

FIGURE 3. Screen "Settings" – Tool parametrization



3. Next, the user should enter the period for viewing the metrics, i.e., the start and end date for searching issues, as shown in Figure 4.

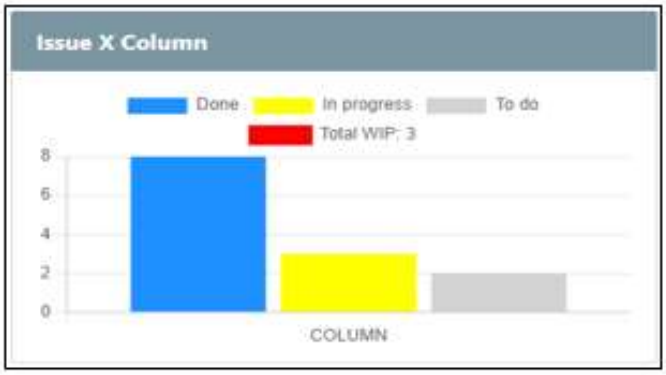
FIGURE 4. Screen "Settings" – Issue start and end date

4. With all the information filled in the "Settings" option, the tool will calculate and display the metrics Lead Time, Cycle Time, Cumulative Flow Diagram, Work in Progress, and Throughput. Figure 5 shows the display panel for all metrics.

FIGURE 5. Metrics Exhibition

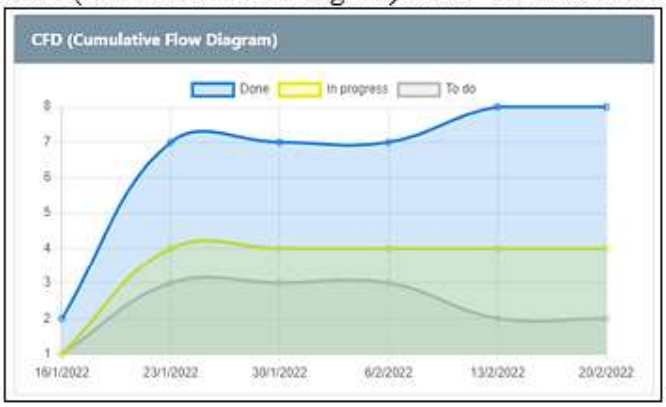
The "Issue x Column" chart (Figure 6) presents the number of issues in each column of the project board. It allows us to visualize the number of issues in progress through the WIP (Work In Progress) metric.

FIGURE 6. Issue x Column chart – WIP metric



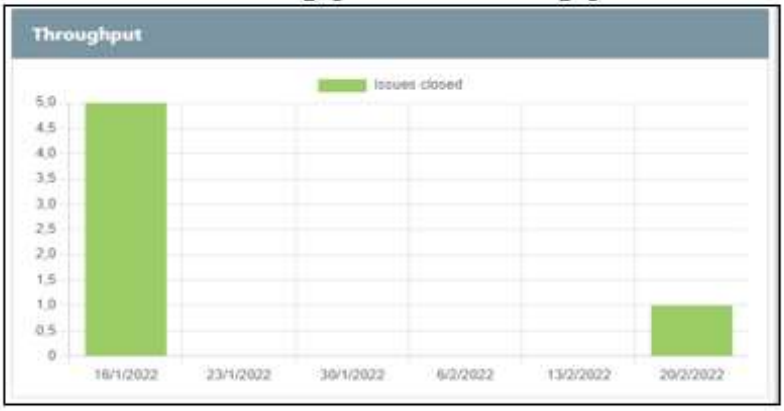
The CFD chart graphically represents the evolution of work through the Cumulative Flow Diagram, as shown in Figure 7

FIGURE 7. CFD (Cumulative Flow Diagram) chart – Cumulative Flow Diagram



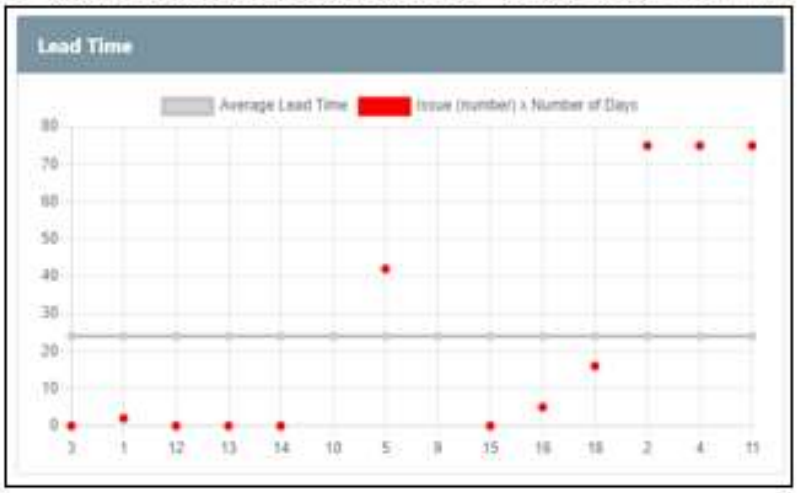
The Throughput chart indicates the number of items delivered in a given period (Figure 8).

FIGURE 8. Throughput chart – Throughput metric



The Lead Time chart indicates the average time that a request takes to go through the entire process, from start to finish. The chart also displays the average Lead Time of issues (Figure 9).

FIGURE 9. Lead Time chart – Lead Time Metric



Finally, the Cycle Time chart displays the total number of days from the start of development of an issue until its completion (Figure 10).

FIGURE 10. Cycle Time Chart – Cycle Time metric

Testing and Evaluation of the tool

After using the tool, a questionnaire was applied to evaluate five quality criteria. The first one was usability, and the tool was considered simple and easy to use, with graphics that were easy to understand and added value to the user.

In terms of performance efficiency, the surveyed users found that the tool provided results in an appropriate time frame. Regarding system reliability, no unexpected errors were found during testing. As for the fourth requirement, security, users noted that only authenticated users could view information from their respective repositories and projects. Lastly, in terms of functional suitability, respondents found that the tool allowed users to plan, estimate, and track projects and tasks by visualizing agile software development metrics.

Based on the evaluation responses, it can be concluded that users found the tool to be simple, useful, and practical for agile software development teams. The displayed graphics were easy to understand and analyse, with reliable information and appropriate loading times. The tool was deemed secure, and the available metrics were considered sufficient to carry out planning, estimation, project and task tracking, and identifying bottlenecks and instabilities in the workflow, which are important activities in the daily routine of teams.

CONCLUSIONS AND IMPLICATIONS

The purpose of this study was to develop a tool that provides the necessary metrics for agile team management, using the information available on GitHub as a basis. To develop the tool, requirements were initially gathered through an interview to collect information from people with knowledge and experience in

agile software development projects. Next, the responses were analysed, and an initial list of requirements was defined. Lastly, a brainstorming session was held to collaborate on ideas, discuss existing needs and problems, and validate and define functional requirements for the tool.

With the identified functionalities and needs, it was possible to model the scenarios and user interaction, define the overall architecture, and develop the web tool. The tool was tested and evaluated according to the main objective of the work, with the aid of a questionnaire constructed according to the product quality model defined in ISO/IEC 25010. It was found that the tool had positive results in all quality criteria, and the existing metrics were considered relevant and easy to analyse and visualize, allowing for project and task tracking on a daily basis.

It is believed that the developed tool can be used by managers and development teams in agile projects, as it is easy to use and contributes to reducing wasted time in consolidating the necessary information for project monitoring, facilitating analysis and visualization of results, and consequently improving team relationships, as the tool brings transparency to the development process. It can also be considered to improve decision-making, as a consolidated view of the overall project situation is available, and continuous improvement.

Finally, regarding literature, it is believed to contribute by delivering a tool that highlights the perception of end-users' needs regarding the main metrics necessary for agile project management. Although the study presented relevant results, it is recognized that there are some limitations to be addressed in future research. An example of this is the need for validation of the tool with a larger number of managers to further refine the results derived from this study. For future studies, it is considered important to conduct a practical test on a group of users with a proposal for full manipulation of the application's functionalities to understand whether the tool really meets the needs presented on a daily basis.

Conflict of Interest

The authors declare no potential conflict of interest regarding the publication of this work.

REFERENCES

Abrahamsson, P., Baskerville, R., Conboy, K., Fitzgerald, B., Morgan, L., & Wang, X. (Eds.) (2008). Agile Processes in Software Engineering and Extreme Programming: 9th International Conference, XP 2008, Limerick, Ireland, June 10-14, 2008, Proceedings. Springer. Lecture Notes in Computer Science Vol. 9 https://doi.org/10.1007/978-3-540-68255-4

Albino, R. D. (2017). Métricas Ágeis: Obtenha melhores resultados em sua equipe. Casa do Código.

Al-Samarraie, H., & Hurmuzan, S. (2018). A review of software techniques in higher education. Thinking Skills and Creativity, 27, 78–91. https://doi.org/10.1016/j.tsc.2017.12.002

Bhasin, T., Murray, A., & Storey, M. A. (2021). Student Experiences with GitHub and Stack Overflow: An Exploratory Study. Proceedings - 2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2021, 81–90. https://doi.org/10.1109/CHASE52884.2021.00017

Beer, B. (2018). Introducing GitHub: A Non-Technical Guide. I. O'Reilly Media, Ed.

Beh, H. C., Jusoh, Y. Y., Abdullah, R., & Hassan, S. (2022). Dimensions in Measuring Performance of Agile Software Development Projects: A Literature Review. In 2022 Applied Informatics International Conference (AiIC) (pp. 83-87). Serdang, Malaysia. doi: 10.1109/AiIC54368.2022.9914025

Budacu, E., & Pocatilu, P. (2018). Real Time Agile Metrics for Measuring Team Performance. Informatica Economica, 22(4/2018), 70–79. https://doi.org/10.12948/issn14531305/22.4.2018.06

Caroli, P. (2020). Diagrama de Fluxo Cumulativo: Uma ferramenta valiosa para melhorar o fluxo de trabalho (Caroli).

Chemuturi, M. (2012). Requirements Engineering and Management for Software Development Projects. Springer, Ed.

Choraś, M., et al. (2020). Measuring and Improving Agile Processes in a Small-Size Software Development Company. IEEE Access, 8, 78452-78466. doi: 10.1109/ACCESS.2020.2990117

Dawson, C., & Straub, B. (2016). Building Tools with GitHub: Customize Your Workflow (B. MacDonald & M. Blanchette, Eds.).

Eisty, N. U., Thiruvathukal, G. K., & Carver, J. C. (2018). A survey of software metric use in research software development. Proceedings - IEEE 14th International Conference on EScience, e-Science 2018, 212–222. https://doi.org/10.1109/eScience.2018.00036

El Sheikh, A., & Alnoukari, M. (2012). Business Intelligence and Agile Methodologies for Knowledge - Based Organizations : Cross - Disciplinary Applications. Business Science.

Fenton, N., & Bieman, J. (2014). Software Metrics: A Rigorous and Practical Approach, Third Edition. CRC Press.

Fernandes, J., & Machado, R. (2017). Requisitos em Projetos de Software e de Sistemas de Informação. Abril.

Fernández-Diego, M., Méndez, E. R., González-Ladrón-De-Guevara, F., Abrahão, S., & Insfran, E. (2020). An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review. IEEE Access, 8, 166768-166800. doi: 10.1109/ACCESS.2020.3021664

GitHub. (2024). Let's build from here. Retrieved July 10, 2024, from https://github.com/

GitHub Resources. (2024). Resources to help enterprise teams do their best work. Retrieved July 10, 2024, from https://resources.github.com/

Hazzan, O., & Dubinsky, Y. (2007). Agile Software Engineering. Springer.

Hemalatha, C., Sankaranarayanasamy, K., & Durairaaj, N. (2021). Lean and agile manufacturing for work-in-process (WIP) control. Materials Today: Proceedings, 46, 10334–10338. https://doi.org/10.1016/j.matpr.2020.12.473

InfoQ. (2024). Agile Epics in GitHub with ZenHub Epics. Retrieved July 10, 2024, from https://www.infoq.com/news/2016/04/zenhub-epics-github-agile/

Islam, K. (2013). Agile Methodology for developing e measuring learning: Training development for today's world. AuthorHouse.

ISO/IEC 25010. (2022). ISO 25000 software and data quality. Retrieved from https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

Keng, S., & Terry, H. (2000). Unified Modeling Language: Systems Analysis, Design and Development Issues: Systems Analysis, Design and Development Issues. Idea Group Inc (IGI), Ed.

Kupiainen, E., Mäntylä, M. v., & Itkonen, J. (2015). Using metrics in Agile and Lean software development - A systematic literature review of industrial studies. In Information and Software Technology (Vol. 62, Issue 1, pp. 143–163). Elsevier B.V. https://doi.org/10.1016/j.infsof.2015.02.005

LABIUTIL - Laboratório de Utilizabilidade da Universidade Federal de Santa Catarina. (2018). Usabilidade: Uma versão atualizada da ErgoList desenvolvida pela UFSC. Retrieved from https://usabilidade.github.io/

Laplante, P. A., & Kassab, M. H. (2022). Requirements Engineering for Software and Systems: Requirements Engineering for Software and Systems. CRC Press, Ed.; 4th ed.

López, L., Burgués, X., Martínez-Fernández, S., Vollmer, A. M., Behutiye, W., Karhapää, P., Franch, X., Rodríguez, P., & Oivo, M. (2022). Quality measurement in agile and rapid software development: A systematic mapping. Journal of Systems and Software, 186. https://doi.org/10.1016/j.jss.2021.111187

Mishra, J., & Mohanty, A. (2011). Software Engineering (Pearson Education India, Ed.).

Niemi, T., Gallay, O., & Hameri, A.-P. (2021). Technical Note: Mean Lead-Time as a Real-Time Key Performance Indicator Subject Areas: Finite Observation Time Window, Lead-Time Analysis, and Performance Measure. In Decision Sciences (Vol. 52).

Ortu, M., Destefanis, G., Graziotin, D., Marchesi, M., & Tonelli, R. (2020). How do you Propose Your Code Changes? Empirical Analysis of Affect Metrics of Pull Requests on GitHub. IEEE Access, 8, 110897–110907. https://doi.org/10.1109/ACCESS.2020.3002663

Patkai, N. (2022). Web Usability Checklist. Https://Teamsuccess.Io/UX.

Ram, P., Rodriguez, P., Oivo, M., & Martinez-Fernandez, S. (2019). Success factors for effective process metrics operationalization in agile software development: A multiple case study. Proceedings - 2019 IEEE/ACM International Conference on Software and System Processes, ICSSP 2019, 14–23. https://doi.org/10.1109/ICSSP.2019.00013

Sage, A. P., & Rouse, W. B. (2014). Handbook of Systems Engineering and Management: Wiley series in systems engineering and management, 2nd ed. John Wiley & Sons.

Sarrab, M., & M. Hussain Rehman, O. (2014). Empirical study of open-source software selection for adoption, based on software quality characteristics. Advances in Engineering Software, 69, 1–11.

Şeker, A., Diri, B., & Arslan, H. (2021). New developer metrics for open-source software development challenges: An empirical study of project recommendation systems. Applied Sciences (Switzerland), 11(3), 1–26. https://doi.org/10.3390/app11030920

Singh, S. R. (2007). Information System Management. APH Publishing, Ed.

Subramanian, V. N., Rehman, I., Nagappan, M., & Kula, R. G. (2022). Analyzing First Contributions on GitHub: What Do Newcomers Do? IEEE Software, 39(1), 93–101. https://doi.org/10.1109/MS.2020.3041241

Suresh, Y., Pati, J., & Rath, S. K. (2012). Effectiveness of Software Metrics for Object-oriented System. Procedia Technology, 6, 420–427.

User Effect. (2009). 25-point Website Usability Checklist. https://Drpete.Co/Pdf/Checklist.Pdf

Venkatesh, V., Thong, J. Y. L., Chan, F. K. Y., Hoehle, H., & Spohrer, K. (2020). How agile software development methods reduce work exhaustion: Insights on role perceptions and organizational skills. Information Systems Journal, 30(4), 733–761. https://doi.org/10.1111/isj.12282

Vijayasarathy, L., & Turk, D. (2012). Drivers of agile software development use: Dialectic interplay between benefits and hindrances. Information and Software Technology, 54(2), 137–148. https://doi.org/10.1016/j.infsof.2011.08.003