International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(2) 2026 pp. 1237-1258
https://doi.org/10.70917/ijcisim-2026-4019
Article

AGILE SOFTWARE DEVELOPMENT RISK ASSESSMENT: A MONTE CARLO-BASED SIMULATION METHOD USING JIRA SCRUM AND KANBAN DATASETS

Neha Choudhary[1], Mahaveer Kumar Sain[2]

[1]Maharishi Arvind Institute of Science and Management, Jaipur, RTU,Kota Rajasthan,India
ORCID 0009-0004-6605-448X, Email Id: neha2861990@gmail.com
[2]Maharishi Arvind Institute of Science and Management, Jaipur, RTU,Kota Rajasthan,India
ORCID 0009-0006-3759-6145, Email Id: mahaveersain@gmail.com

Abstract: Agile Software Development (ASD) has turned out to be the foundation of the present-day software development practices with the provision of iterative processes, such as Scrum or Kanban, that emulate focus on flexibility, cooperation, and continuous deployment. Nonetheless, it is always a cumulative issue to deliver on schedule when uncertainty strikes because the teams have changing loads, backlogs, and lack visibility on potential risks. Common planning methods, like average-based velocity or fixed sprint length, are unable to reflect real variation of the world and thus are subject to poor forecasts and lack of confidence by the stakeholders. To overcome this, the paper has suggested simulation-based risk analysis model which will rely on Monte Carlo Simulation (MCS) and will use the real empirical data of three practical software projects including GFG (Scrum), and User grid and Aurora (using Kanban). The time a particular historical issue was resolved and data on sprint performance were processed in order to calculate a velocity distribution which was in turn applied to predict the backlog completion timeline over 10,000 MCS iterations. Further improvement of simulations with SimPy further decreased the variability of the forecast and produced single estimate of project completion with a median (P50) of 3.45 sprints with a narrow 95% confidence interval of 3.45 to 4.60 sprints as opposed to the broader estimate of 16.67 sprints of the baseline MCS. Histograms, burn-up charts, and cumulative distribution graphs along with other visual solutions, ensured smooth intuitive performance of risk communication and comparisons between Scrum and Kanban. The framework also enables probabilistic planning (versus a static one) in as far as uncertainty is measured in the form of percentile-based estimates (as P10, P50, P90). The use of accessible technologies like Python and Jira REST APIs ensures the framework remains cost-effective and practical for Agile teams. This research advances Agile risk assessment by integrating statistical modeling with real project data, improving forecasting accuracy, stakeholder alignment, and delivery confidence—ultimately providing a scalable solution for managing delivery uncertainty in dynamic Agile environments.
Keywords: Software Development, Agile methodology, Risk Assessment, Scrum, Kanban, Monte Carlo Simulation (MCS).

1. INTRODUCTION

In today's fast-paced software engineering landscape, software development projects are under increasing pressure to deliver high-quality products rapidly while staying responsive to changing customer requirements and market dynamics (Dingsøyr et al., 2012; Prajapati,2025). Software development methodologies seek to improve customer satisfaction, lower development costs, accelerate time to market, and boost team efficiency. To accomplish the aforementioned objectives, agile software development

(Spagnoletti et al., 2022; Goyal, 2023), Often referred to as agile development, it has progressively sparked public debate since the 1990s (Modalavalasa, 2021). It supports evolutionary development and adaptive planning, has the same principles as the software process, and promotes early delivery and continuous improvement to enable quick

and adaptable reactions to changes (Alsaqqa et al., 2020). Agile development places a strong emphasis on human-centered collaboration, face-to-face communication, self-organization and management, quick development, and modest planning. Software companies use extensive agile methods (Islam & Ferworn, 2020; Edison et al., 2022), to reproduce, at the organizational level, the effectiveness of agile methodologies on team projects. The agile software development approach has many drawbacks that might result in project failures, despite its ability to adapt to changing requirements, address issues efficiently and rapidly, boost production, and reduce the overall development time. For instance, the first Scrum issue is requirement identification and preliminary planning (Sherif et al., 2023). The disregard for design is another issue with Scrum. During the analysis and design stage, Scrum gives requirement engineering less consideration. Furthermore, its inability to track papers, files, and configuration management might result in project failure or worse product quality (Dada & Sanusi, 2022).

To tackle this complexity and uncertainty, many organizations have embraced Agile methodologies, which offer flexible and adaptive frameworks to manage evolving project needs (Kumar et al., 2019; Kalava, 2024). Agile methodologies are a family of principles and practices applied in software development (and in other industries) that posits greater flexibility, collaboration, feedbacks by customers (or, the user), and speed to delivery of functionally usable software. Agile methodologies are iterative, incremental methods of project management and software development, whose aim is to deliver small of workable parts of a project on a regular basis, with a view to the teams responding in quick, frequent iterations to changes and new requirements (Chan & Thong, 2009). Agile Software Development is a practical approach based on the principles of the Agile Manifesto that are about customer satisfaction, tight collaboration, and the skills to respond promptly to change. Through the iterative and incremental delivery cycles, Kanban et al would allow development teams to deliver workable software frequently and regularly, solicit ongoing feedback (Zayat & Senvar, 2020). Reorder priorities that are consistent with the expectations of the stakeholders. This flexibility is the strongest asset of agile in the current software development projects (Abioye et al., 2020; Goyal, 2022). Agile methodology of risk management is concerned with constant improvement, communication, and adaptive planning in order to execute the delivery of projects in a fast and effective manner (Singh et al., 2023). It is a project management process that is considered to be iterative and incremental and focuses on early and frequent delivery as well as the flexibility toward evolving requirements and customer demands. The traditional methods of risk management tend to fail when used in the dynamic environments of agile, since a chunk of these methods is involved in the thorough initial planning (Alencar et al., 2018). The increasing difficulty places more emphasis on the need to establish efficient agile software development risk assessment and risk management techniques, which are congruent with the iterative characteristics of the agile processes (Prajapati,2025). Improving it by incorporating risk management in agile practices means that the teams will be able to find, explore, and manage the risks on a continuous basis, making sure that the risks will be mitigated in time and the projects will not lose their agility (Omar et al., 2024; Dos Santos et al., 2023). Effective risks lead to success of the project directly by protecting the deliverables, budgets and schedules (Goyal, 2024).

Software Development and Simulation-based methods that can be integrated well into the agile frameworks are garnering interest because it improves this capability (Ghane,2017). One possible answer is the use of Monte Carlo simulation which has the potential to create uncertainties related to risk and to forecast the consequence of risks with actual project data. Teams can make informed decisions, through proper use of Jira Scrum and Kanban datasets (Goyal,2023), to allow them to simulate various scenarios to provide useful insight in the probable risk outcomes that may occur during the lifetime of the project. It is a result-oriented paper that investigates Monte Carlo-Based Simulation Method Using Jira Scrum and Kanban Datasets as an advanced form of the agile risk assessment.

1.1 Risks of software development in agile

Although agile software development stimulates the concepts of flexibility, adaptability, and efficient delivery, it is also a set of risks that may affect a successful project, particularly in the dynamic environment where Scrum and Kanban are performed (Garcia et al., 2022; Salin & Lundgren,2022). The major risks include:

- **Uncertainty in Delivery Timelines:** Based upon the changing demands as well as alterations in priorities/intentions there is uncertainty in determining the delivery time, when all factors are not constant (team velocities are variable).
- **Inconsistent Sprint Velocities:** Agile teams can have a variable rate of productive since it can take into consideration the availability of the personnel, technical blockers, varying difficulty of tasks, and this can complicate proper planning.

- **Inadequate Risk Buffers:** A conventional estimation does not have in-built risk buffers and these margins are more likely to make their schedule seem optimistic than not and, in most cases, end up exceeding their schedules.
- **Poor Visibility into Backlog Progress:** Without quantitative forecasting, teams will find it difficult to visualize the expected effect on the backlog caused by changes to the backlog itself or to the team (and its available resources).
- **Over-Reliance on Point Estimates:** Excessive rates of point estimates are mostly reflective of affected Agile teams which use constant rates to estimate the estimation (average velocity or constant story points) in a manner that does not reflect real-life unpredictability and uncertainty.
- **Difficulty Comparing Methodologies:** Different Agile teams that belong to different frameworks (i.e., Scrum versus Kanban) will find it hard to make a consistent approach of comparing risks across the processes.

1.2 Problem Statement

Although it is often overlooked in actual projects, software risk management is essential for effective project management. Project managers frequently lack efficient software risk management tools and procedures, which is one of the primary causes. This is due to the iterative and flexible approach to software development, including the principles of being customer-centric features of agile methodologies especially Scrum and Kanban which have become very popular. Nevertheless, such techniques are fraught with great uncertainty and variation especially with relation to when a project will be completed and the risk associated with the same. In both of these cases, tools like Jira provide access to ample historical data, yet many Agile teams still resort to subjective, or otherwise too simplistic estimation methods, (e.g., average velocity, fixed sprint planning, etc.), which do not reflect the complexity and uncertainty of real-life development cycles. This absence of solid practices in forecasting causes a divergence in planning and real delivery which in most instances results in missed deadlines, unconsidered risks as well as a lack of stakeholder assurance. The Agile projects are characterized by dynamic and incremental aspects that do not fit well into conventional risk management models and this calls urgently a need to have flexible and software development methods that can provide probabilistic values regarding the outcome of projects and the possibility of schedule delays. The fundamental issue as far as this study is concerned is:

RQ1: In what ways Monte Carlo simulation methods, in particular, can be applied as a solution to evaluate the delivery risk of Agile software development projects based on past performance of the Scrum and Kanban projects?

In order to solve this, the paper employs three real project data (GFG, Usergrid, and Aurora) to come up with sprint velocity distributions, simulate backlog completion, and achieve delivery schedules at different confidence levels. The method is a means of equipping the Agile practitioners with a more realistic and feasible risk forecasting and decision making tool.

1.3 Motivation of this Paper

Approaches to software development, such as Scrum and Kanban, which are based on agile principles, are ardently necessary in the contemporary software development process, allowing products to be delivered at greater speed and with constant progressive development. But Agile projects nevertheless remain at high risks and uncertainty because of changing requirements, performance of changing teams and unpredictable delays which the conventional static planning approaches are not able to point out correctly. Although tracking of work through tools such as Jira has become commonplace, teams infrequently use past data to deliver predictions of risks or schedules in a statistical rigorous way. The inspiration of carrying this study is the continued problem that Agile teams have when it comes to properly predicting when a project will be delivered in a high uncertainty environment and with rapidly changing requirements and variable team performances. Although Agile tools such as Jira can provide deep historical data, the majority of Agile teams still use non-realistic planning patterns that base plans on intuition and offer no measure of variability in the actual world. This result in unrealistic schedules, uncontrolled risks, and lower confidence of the stakeholders. Taking into consideration the necessity to have a more reliable, software development strategy, the objective of the current research is aimed at bridging this inequality by means of empirical Jira Scrum and Kanban databases supplemented with Monte Carlo simulation. It is expected to allow Agile practitioners to measure uncertainty levels, evaluate their risks probabilistically, and base planning decisions on an intelligent and confident deduction that improves project results and prevents delays.

1.4 Significance and Contribution

The relevance of this work is that it seeks to fill a major loophole in Agile software development which is attaining better control of uncertainty and risk in delivery schedules. The unstable nature of project outcomes because of insufficient forecasting solutions, changing requirements, and irregular performance of team members are some of the factors that agile teams are finding difficult to deal with. This study also puts special emphasis on the need to abandon the artificial estimates and instead adopt a data-driven approach towards improving the comprehension of delivery risks and the subsequent approach to control them. This work emphasizes the real application of historical performance data in order to help a team complete a project to evaluate the variability of the project and predict possible delays before it can influence the project. It strengthens decision-making, improves stakeholder trust through transparent communication of project expectations, and enhances overall project reliability. The framework's adaptability across both Scrum and Kanban methodologies further reinforces its value in modern, fast-paced development environments, promoting more confident planning and risk-aware execution. This study makes a significant contribution by bridging the gap between traditional Agile project planning and modern software development risk management practices. The following research contributions of this work are:

- To introduce a novel simulation-based framework that integrates empirical Jira project data with Monte Carlo simulation for assessing and forecasting Agile project delivery risks under uncertainty.
- To provide a practical method for computing sprint velocities from historical issue resolution data, enabling consistent modeling and comparison across different Agile methodologies such as Scrum and Kanban.
- To generate probabilistic delivery estimates (P10, P50, P90) that serve as a more accurate and risk-aware alternative to traditional static average-based planning in Agile project forecasting.
- To enhance risk visibility and communication through the use of visual tools like cumulative distribution plots, histograms, box plots, and burn-up charts, making risk insights more interpretable for Agile stakeholders.
- To enable empirical comparison of risk and predictability between Scrum and Kanban workflows, supporting Agile teams in selecting or refining methodologies based on real-world performance trends.
- To demonstrate a low-cost, tool-driven implementation using Python and Jira REST APIs, making the risk assessment framework easily adoptable for Agile teams without the need for specialized or expensive software solutions.

The overall contribution of this work lies in providing a practical, software development framework that enhances Agile project planning through accurate, probabilistic risk forecasting. It empowers teams to make informed decisions, manage uncertainties, and improve delivery confidence using accessible tools.

1.5 Organization of the Paper

The following structure of paper is: Section II presents the literature review for software development. Then Section III provides the proposed methodology with Jira framework and novelty and justification. Section IV provides the experimental results of proposed system with risk solutions, advantages, Limitations and future directions, at last Section V concludes this work with key findings and analysis.

2. LITERATURE REVIEW

In this section, Table 1 discusses the previous studies conducted in the form of papers in risk assessment area in agile software development methods. Risk assessment is the identification, management, and prioritization of risks followed by organization of resources to monitor the probability and/or the impact of regrettable events. Many risk assessment approaches followed in agile software development, because of their great significance in the success of projects.

Javed et al. (2025) focus on identifying, analyzing, and mitigating risks while ensuring clear communication protocols, realistic project timelines, appropriate tools, and skilled team members. The framework is validated through expert feedback, after results and analysis, the proposed framework demonstrates a 66.7% comprehensive rate. Over 75% of industry specialists support its implementation, with more than 55% considering it feasible for GSD projects. This approach has proven effective in minimizing risks and ensuring successful project delivery in complex, distributed development environments (Javed et al., 2025).

Ibraigheeth (2024) supports decision-makers in making critical early decisions to prevent potential events that could lead to project failure. Behavior and sensitivity analyses were conducted using eight virtual projects to observe the impacts of different factors. The exploratory analysis of the proposed ensemble model revealed critical insights

regarding project failure thresholds. In the testing dataset, it was found that all projects with a failure probability of 0.45 or lower were successful, while a score of 0.6 indicated a mixed outcome, with three failed and two successful projects. Consequently, a failure probability threshold of 0.6 was established, indicating that projects with probabilities at or above this value are likely to fail (Ibraigheeth, 2024).

Shaout and Kyer (2024) present a new tripartite software model (VSK) that has integrated agile methodologies, the Scrum and the Kanban methodologies, along with the traditional methodology the V-model. Since the V-model is commonly used in the automotive space for systems engineering activities, this was selected as the traditional framework for this new tripartite model. The bottom of the V will contain two separate Agile development methodologies that will run in parallel. A scrum methodology will be utilized for new feature development and release planning activities. In parallel to the Scrum methodology, a Kanban methodology will be used to track bugs and performance issues (Shaout & Kyer, 2024).

Seffe, Odzaly and Samah (2024) involved two phases in developing the proposed conceptual model: 1) problem assessment and research study and 2) data-driven strategies to enhance risk management performance. Then, the model is evaluated based on its ability to provide accurate and useful information about risk factors such as deadlines for the project, resource allocation, and stakeholder feedback. Besides, gaining expert validation shows the model's effectiveness in real-world software development environments. In conclusion, based on the developed model, they can ensure it will significantly ameliorate risk management performance in software development (Seffe et al., 2024).

Sheikh and Singh (2023) examine the current literature that establishes the implicit security needs using formal security approaches, security modelling techniques, and security compliance mechanisms. It covers a variety of methods, including Attack Tree Analysis (ATA), System Theoretic Process Analysis (STPA), Failure Modes and Effects Analysis (FMEA), and Fault Tree Analysis (FTA). In order to cut expenses and effort throughout the development process, a secure agile development framework is also suggested. Their framework's main components depend on rigorous training, the creation of prototypes, planning and replanning throughout the prototyping stage, including testing into each iteration, and implementation in accordance with security requirements (Sheikh & Singh, 2023).

Khurana and Wassay (2023) after presenting the research of risk management concerns, perspectives on these issues are further developed, expanding on the characteristics of potential solutions. Ultimately, each challenge's answers are provided depending on the characteristics Agile-based software development approaches rely heavily on the work completed here. The following five phases were identified based on the ideas drawn from how each of the challenges has been addressed in the literature that was studied: A gathering of potential risk variables, Knowledge transmission, risk escalation, risk mitigation, and risk refinement. As a starting step, the conclusion presented here might help project managers and developers structure their projects with cybersecurity risk management in mind, encouraging less overworked agile procedures and stakeholder perspectives on relevant topics (Khurana & Wassay,2023).

Table 1. Summary of recent studies on risk management in agile software development

Reference	Key Focus	Approach	Key Findings	Limitations	Recommendations
Javed et al. (2025)	Risk identification, analysis, mitigation; clear communication and skilled teams for GSD	Framework validated by expert feedback and practical analysis	66.7% comprehensive rate; >75% industry support; >55% feasibility for GSD projects	Limited to expert feedback; lacks large-scale real-world validation	Apply the framework on diverse real projects to verify effectiveness
Ibraigheeth (2024)	Early decision-making to prevent project failure	Behavior & sensitivity analysis on 8 virtual projects; ensemble model testing	Projects ≤0.45 failure probability succeed; ≥0.6 likely to fail	Limited dataset (virtual projects only); threshold may vary in real scenarios	Test with real-world project data to adjust thresholds
Shaout and Kyer (2024)	New tripartite software model (VSK): V-model + Scrum + Kanban	Integrates V-model with parallel Agile methods (Scrum for new features,	Supports parallel handling of new dev & maintenance	Applicability limited to automotive	Validate model in other domains beyond automotive

		Kanban for bugs)		domain; not tested broadly	
Seffe, Odzaly and Samah (2024)	Conceptual model to enhance risk management	Two phases: problem assessment & data-driven strategy; expert validation	Accurate risk info on deadlines, resources, stakeholder feedback	Conceptual level only; lacks detailed implementation results	Implement and test in diverse software dev environments
Sheikh and Singh (2023)	Secure Agile Development Framework; implicit security requirements	Combines FTA, FMEA, STPA, ATA, security modeling; secure agile framework	Integrated security reduces costs/effort via continuous training, prototyping	Practical implementation details not discussed	Develop practical tools and training material for adoption
Khurana and Wassay (2023)	Challenges in Agile risk management & solutions	proposes 5-step solution (risk collection, refinement, mitigation, knowledge transfer, escalation)	Guides developers/managers in structuring cybersecurity risk management	No empirical validation; general guidelines only	Conduct case studies to test effectiveness in live Agile projects
Khanna, Popli and Chauhan (2022)	Risk Management in Distributed Agile Software Development (DASD)	Literature review; identifies 71 risk factors grouped in 11 categories	Timely management can reduce uncertainty in DASD	No practical framework or tools proposed	Develop practical risk management tools for DASD projects

Khanna, Popli and Chauhan (2022) examine recent research and outline the difficulties facing risk management in the context of distributed agile software development (DASD). Additionally, 71 risk variables linked to DASD are identified in this article and analyzed according to their origins and causes. Additionally, these risk variables are divided into eleven distinct groups. In the context of DASD, prompt risk management might lower the likelihood of project failure (Khanna et al.,2022)

2.1 Research Gaps

The reviewed literature highlights significant efforts in enhancing risk assessment within Agile software development by focusing on various aspects such as risk identification, early failure prediction, hybrid methodology integration, conceptual modeling, and security-focused frameworks. While existing studies have contributed models that aid in decision-making, categorize risk factors, and offer solutions tailored to distributed or security-sensitive environments, they often lack dynamic, data-driven simulation capabilities that reflect real-time project variability. Most approaches rely heavily on qualitative feedback or static risk classifications, failing to capture the probabilistic nature of delivery uncertainty across different Agile workflows like Scrum and Kanban. Additionally, few frameworks provide

visual, empirical comparisons between methodologies or utilize actual sprint data for forecasting. The proposed system addresses these research gaps by introducing a simulationbased risk assessment model using Monte Carlo techniques on real Jira project data. It overcomes limitations of prior models by quantifying uncertainty through percentile-based forecasts, integrating velocity trends, and offering enhanced visual tools for risk communication, ultimately enabling practical, scalable, and low-cost implementation for improved risk transparency and delivery confidence in Agile environments.

3. METHODOLOGY

The aim of this study is to develop a simulation-based risk assessment framework for Agile project delivery using empirical data extracted from real-world Jira software projects. As illustrated in Figure 1, the proposed

methodology starts with data collection from three Agile Jira projects using the REST API. Extracted issue-level data is preprocessed to compute metrics like velocity and resolution time. These metrics inform an empirical model used in a Monte Carlo simulation to estimate sprint requirements for completing a fixed backlog, incorporating real-world variability and risk. Outputs include percentile-based forecasts and confidence intervals. Visualizations such as histograms, CDFs, and cumulative flow diagrams support risk assessment and enhance delivery predictability for Agile teams.

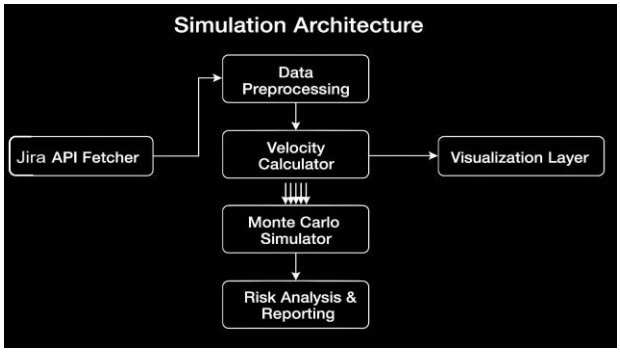

Figure 1. Proposed Simulation Architecture for Agile Software Development Risk Assessment

Following flowchart of the proposed system for Agile delivery risk analysis using real-world Jira data and Monte Carlo simulation is illustrated in Figure 1. The implementation steps are discussed below:

3.1 JIRA Framework

In this research, Jira Atlassian served as a critical data source for performing a simulation-based risk analysis of Agile project delivery. Through Jira API, stories, bugs, and tasks of past issue data in three projects in three real-life scenarios (one Scrum and two Kanban) were scraped successfully. Important performance indicators including velocity (number of issues completed in sprint or pseudosprint), resolution time (how long it takes to fix issues), and backlog size were also calculated. These metrics constituted the main corpus of a Monte Carlo simulation model that simulated numerous delivery situations and the measurement of risk on the diverse Agile workflows. With this method, the study was able to extend the conventional static planning to contemplating dynamics of how the real project is executed and it therefore gives the project delivery schedules a probabilistic perspective. The attendant forecasts such as percentile-based completion projections and cumulative distribution graphs provided a software development comparison of predictability between Scrum and Kanban and provided an avenue of making a manageable step towards incremental Agile planning. Besides, by generating visualization graphs based on Jira project data, exploratory data analysis (EDA) was also carried out to find out the trends, outliers, and performance patterns across project teams. Jira bar charts and cumulative flow diagrams facilitated the generation of insights and early risk identification in the projects.

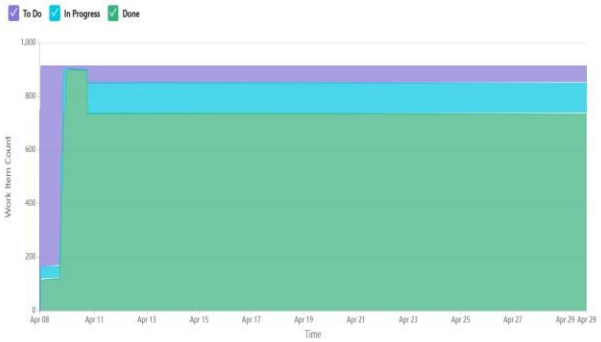

Figure 2. Cumulative Flow Diagram UserGrid Project

Figure 2 represents the Cumulative Flow Diagram (CFD) of the UserGrid Project, which demonstrates how the work items are distributed evenly throughout three workflow states (To Do purple, In Progress blue, and Done green) at a period of time. The chart shows a sudden escalation in all categories of work items around April 10, which must have been an import or a planning exercise. After this peak, the parallelism of the bands representing each state is

fairly constant and over time, it can see no significant variation in the pattern, indicating just a stable workflow without significant bottlenecks. The wide area of Done means that throughput is healthy, and the fact that the In-Progress area is relatively narrow and that the To Do area is flat means that there is effective WIP control and an even pace of completion. On the whole, the CFD indicates a controlled process of Kanban with the proper flow of tasks and few interruptions. Moreover, this diagram can be a solid basis of deep analysis like determination of cycle time and throughput rates, bottlenecks, and feeding of historical data into Monte Carlo simulations to get predictions. The insights can assist in making continuous process improvements, enhanced resource allocation, and risk-aware planning of delivery.

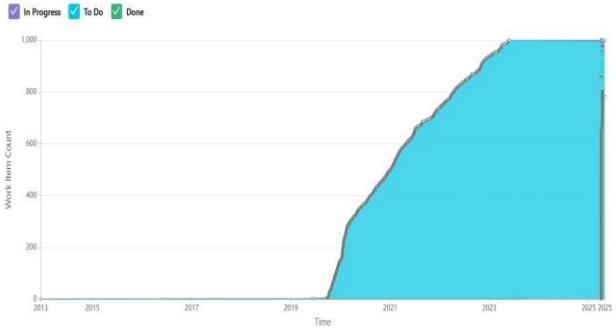

Figure 3. Cumulative Flow Diagram GFG Project

Figure 3 represents the Cumulative Flow Diagram (CFD) of the GFG Project, which showcases the trends of work items that are developed over a prolonged period of several years (2019-2025). The chart demonstrates that the In Progress (blue) state increased dramatically and steadily since 2019, which means that there is active continuous development with a high rate of growth towards 2023. The high compactness and consistency of the blue segment indicate heavy volume of business being launched and carried out in the past. Nevertheless, nearly unchanging widths of To Do (green) and Done (purple) close to upper right corner show how the backlog replacement process and completion success might have been stagnant, which could be interpreted as possible bottlenecks or delays in getting items to be completed. The broad time range and cumulative addition of work result in this CFD being an important addition to long-term trend analysis, determining capacity constraints, and the use of advanced forecasting methods such as Monte Carlo simulation to determine the potential risks of deliveries in the future and to give the proper throughput.

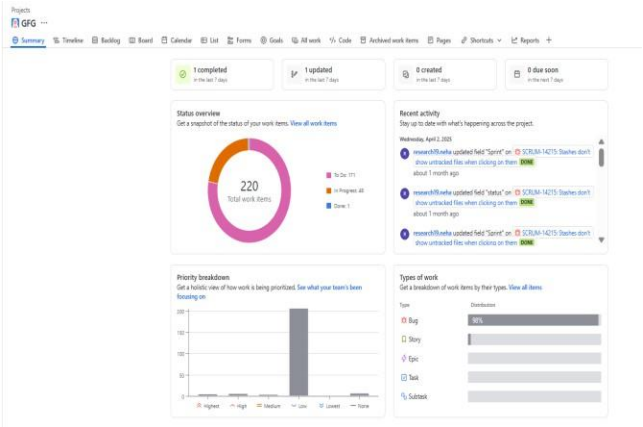

Figure 4. Jira Dashboard for the GFG Project

In Figure 4 below, the Jira Dashboard of the GFG Project that provides a complete overview of the project activity, status, and priority of work. According to the Status Overview, there are 220 work items, most of which are at the To Do phase (171), then there are at the in Progress (48), and then there is only 1 piece that is marked as Done, which is a sign of progress being made or progress being impeded. On the Priority Breakdown, there is an extensive emphasis toward medium-priority tasks indicating a balanced workload control. The most recent activity logs include the changes regarding sprint fields and issues, actively cooperating. In the Types of Work section, it becomes clear that almost all the items of work are in the responsibility of the type Bug which indicates the adjustment or bugs

fixation stage. All in all, this dashboard can be seen as a place where project health can be monitored and can be regarded as the central interface where the overall picture of the project can be viewed in real-time, allowing to assess its status almost immediately, informing decisions made by Agile teams.

3.2 System Design for Risk Assessment

The proposed research introduces a risk analysis approach that considers Jira data and Monte Carlo simulation to predict the delivery of Agile projects with uncertainty. Data collection It starts at the data gathering phase, collecting information on three problems managed in Jira (GFG, Usergrid, and Aurora) using REST API to retrieve information at the issue level. Sprint velocities are then calculated and it is empirically modeled, after the preprocessing, such as converting timestamps, calculating the resolution time, and structuring the sprints. This set of velocity distributions drives a Monte Carlo simulation (10,000 iterations) to predict how many iterations of sprinting will be required to go through a set backlog and are incorporating risk padding and the potential variability of sprint duration. The simulation yields percentilebased forecasts (P10, P50, P90) and confidence intervals, visualized through histograms or CDFs. This enables Agile teams to make data-driven, risk-aware planning decisions with greater confidence.

3.3 Data Collection

To create credible simulation model, data collection stage plays the pivotal role because by this stage the empirical ground on which all other analysis are based was layed. In this study, historical data was collected from three real-world software projects managed through Atlassian Jira, a widely used Agile project management platform. The selected projects include:

- GFG (following the Scrum methodology)
- Usergrid and Aurora (following the Kanban methodology)

Data was retrieved using the Jira REST API, which allows programmatic access to issue-level records from project repositories. The API interaction included appropriate authentication mechanisms (such as API tokens or OAuth) to ensure secure data access. For each issue retrieved from the Jira system, the following attributes were collected:

- **Issue Key:** A unique identifier for each task.
- **Issue Type:** The classification of work (e.g., Bug, Task, Story).
- **Created Date:** The time at which the issue was created.
- **Resolution Date:** The timestamp of issue closure or completion.
- **Assignee:** The team member responsible for the task.
- **Status:** The current or final state of the issue (e.g., Done, In Progress, Closed).

These areas will help give an idea about what the work is like as well as its lifecycle, which will be the basis of further velocity and risk analysis. The fact that both Scrum and Kanban-based projects are included grants an opportunity to compare the patterns of delivery and risk profiles of the two approaches.

3.4 Data Preprocessing

Once the raw data was obtained, a number of preprocessing operations were used to prepare the data, clean it, and transform it into a form which can be simulated. This harmonizes the information and obtains substantial variables that are needed in modeling velocity and analyzing risk. These are the data preprocessing steps that have been undertaken:

1) Timestamp Conversion

Timestamp conversion refers to the act of converting date and time data, in text or string format in fields to standard form as denoted by the programming environment in the form of datetime objects. To this end, the Created Date and Resolution Date columns in each of the issues were originally stored as strings in this study. Each of these fields was then transformed into the form of a python datetime object in the form of the pandas.to_datetime(). With

this transformation, it is possible to do correct time-based operations like subtracting dates in order to calculate duration and dividing data into fixed quantities of time in order to create pseudo sprints.

2) Resolution Time Calculation

Resolution time is the time that it takes all together to do work and that is calculated between the date on which the work is created and the date it is resolved. The resolution time per issue was computed by the program by the Created Date and the Resolution Date. The outcome was tabulated in terms of days, which gave a clear indication of the rate at which things were done. This metric was later used to evaluate team performance trends and to visualize issue resolution patterns across time.

3) Filtering Unresolved Issues

Filtering unresolved issues involves removing tasks that have not yet been completed from the dataset used for performance analysis. Issues without a Resolution Date were considered open or in-progress and were excluded from further analysis. This step is crucial because including unresolved issues could distort sprint velocity metrics, leading to inaccurate forecasts. Only fully completed work items were retained for velocity modeling and Monte Carlo simulations.

4) Sprint Structuring

Sprint structuring refers to organizing work items into time-boxed intervals (sprints) to facilitate consistent measurement of team output (velocity).

- For the Scrum-based GFG project, the Jira dataset contained well-defined sprints. Each issue was already tagged with a sprint ID, enabling direct grouping of resolved tasks by sprint.
- For the Kanban-based Usergrid and Aurora projects, where traditional sprints are absent, the timeline was divided into fixed 14-day intervals, simulating pseudosprints. All issues resolved within each interval were grouped as a sprint equivalent. This allowed for uniform velocity computation across both Scrum and Kanban methodologies and enabled meaningful comparisons.

5) Velocity Calculation

Sprint velocity is defined as the number of completed work items within a sprint or sprint-like period and is a key performance indicator in Agile projects (Huss et al.,2023). Once the issues are sorted into the real- or pseudo-sprints, the number of issues resolved in each time frame was summed up. This resulted in a series of sprint speeds, which is historical team throughput. This set of values of velocity was matched in a list and used as starting input in the Monte Carlo model of simulation so that probabilistic delivery forecasts could be generated.

Comprehensively, these preprocessing procedures yield a massive boost in the predictiveness and visibility of the simulation results, thereby facilitating better decision-making in the planning of Agile projects.

3.5 Velocity Distribution Modeling

Velocity distribution modeling is a decisive step in forecasting (Koch, 2013), The Agile delivery timelines with the help of probabilistic simulations after the preprocessing. Here, the research measures the amount of issues that a team closes in a given sprint (or pseudo-sprint) and the learned information is applied so as to construct an empirical distribution of velocity. This distribution models the variance of a team in its delivery performance that forms a statistical basis of simulation-based forecasting. After the cleaning, and structure of the data are completed, modeling of the distributions of sprint velocities follows.

1) Sprint Velocity

Sprint Velocity is the count of the issues that were resolved during a particular sprint. It is a measure of the speed that a

team can get through their work in a set amount of time, computed by the number of issues that were completed in each sprint or pseudo-sprint. Mathematically, velocity in sprint i is given by Equation (1):

□□ = $\sum_{j=1}^{N_i} 1$(□□□□□□ ∈ □□□□□□□□∧= □□□□□□□□□) (□) Where:

- □□=is the velocity of sprint i.

- □□=is the total number of issues.
- 1=is the indicator function that equals 1 if the issue is resolved in that sprint.

2) Velocity Extraction for Scrum-Based Project (GFG)

For the Scrum-based GFG project, the sprint data is explicitly available, including sprint identifiers linked to each issue. The velocity is computed by counting all issues resolved within each actual sprint period. This yields a series of velocity values □1, □2, … … □□ , that can be considered historical data points. Such values enable the formation of a trend and variance in performance of the team with time.

3) Velocity Extraction for Kanban-Based Projects (Usergrid and Aurora)

As opposed to Scrum, Kanban avoids explicit sprints. The paper simulates pseudo-sprints (fixed 14-day periods for the purpose of generating comparable data). All the problems that were fixed in one 14-day period are bundled together, and velocity is determined the same way. The pseudo-sprint i

velocity, in Kanban, is Equation (2):

□□ = □□□□□ □□ □□□□□□□□ □□□□□□ □□□*h*□□ 14 −

□□□ □□□□□□□□ □

This method allows for uniform comparison between different project types and ensures consistency in modeling team performance across frameworks.

4) Construction of Empirical Velocity Distribution

Once the velocities between sprints or pseudo-sprints are calculated, the information will be viewed as an empirical sample of the delivery capacity of the team. The study does not make any normal or theoretical distribution-based assumption but rather uses Empirical Distribution Function (EDF) or non-parametric methods, such as histograms and Kernel Density Estimation (KDE). The probability mass function for each velocity level is estimated using Equation (3): (□ = □) = □□□□□□ □□ □□□□□□□ □□□*h* □□□□□□□□□ □ (□)

□□□□□ □□□□□□ □□ □□□□□□□

This kind of empirical modeling does not make any assumption about underlying distributions, it more accurately reflects the behavior that the team actually observes.

Finally, this can be used to improve the effectiveness of simulation-based delivery predictions, as project managers have the ability to make bolder, more knowledgeable and transparent planning decisions.

3.6 Monte Carlo Simulation Setup in Agile Forecasting

One such very powerful statistical technique that could be used to simulate uncertainty and unpredictability in Agile project delivery is Monte Carlo simulation. Monte Carlo simulation in the context of this study calculates the number of sprints necessary to be able to finish a fixed backlog of work items based on the past sprint velocities (e.g., 100 issues). The method offers the probability of possible project completion schedules to offer planning on a data-driven basis in uncertainty. The following is executed in this regard:

1) Simulation Configuration and Iteration Process

The simulation has been set to have a total of more than 10,000 iterations which is adequate to reach statistical stability. In iteration, a sample of sprint velocities are chosen with replacement of the existing empirical velocity already modeled. This method of sampling guarantees that each of the iterations can be considered a realistic way of project development as per them given performance of the team historically. The elapsed sample set of velocities is incrementally monitored to have an estimation of the number of sprints required, in order to fix 100 issues. In case the velocities sampled in an iteration sum up to or larger than the backlog in a particular iteration, then that iteration sprint count is marked as a valid outcome.

2) Sprint Velocity Sampling and Backlog Completion

Let backlog size be B=100 issues. In each simulation iteration □ sprint velocities □1, □2, … … □□ are drawn from the historical data, and cumulative velocity is computed as Equation (4):

$$\sum_{i=1}^{n} x_i \geq T \quad (5)$$

The minimal n where this inequality is true is equal to the number of sprints that had to be made in that simulation round. This operation is done 10 to 000 times and gives a distribution of the necessary sprints. The likelihood of finishing the backlog in a range of sprints is considered using this distribution since it reflects the reality that not all sprints take the same amount of time and team performance naturally varies.

3) Incorporation of Risk Buffers and Sprint Variability

To further the degree of realism, the simulation features risk buffers that are configurable, percentage additions to the estimated number of sprints to be completed to account for possible delays, technical blockers or disruptions to the team. As an example, a 10 per cent risk buffer changes the number of required sprints as follows calculate Equation (5):

$$\text{adjusted\_sprints} = n \times (1 + \text{risk\_buffer}) \quad (6)$$ with the estimated number of sprints given by the simulation denoted by n. Further, the length of a sprint is randomized between a range (usually between 7 to 21 days) between iterations. This is to cover different team cadences or flexibility within the length of sprints either because of holidays, organizational changes, or release cycles. The number of days of the total estimated delivery time per each iteration is then calculated as Equation (6):

$$D = \sum_{i=1}^{n} s_i \quad (7)$$ where D is the total time of delivery, s_i is the amount of sprint during iteration, i is the sprint seconds (chosen randomly among 7 and 21 days).

This will give a project manager a quantifiable guideline to making decisions, as one could visualize the distribution of risk and have visions on the likely time of delivery, rather than using single point estimations. Including a historical team behavior, real-world uncertainty and able-to-be adjusted risk factors, the Monte Carlo simulation generates actionable and evidence-based forecasts. Such predictions enable Agile teams to profile iterations, releases, and resource assignments with much more precision and confidence of the stakeholders.

3.7 Statistical Analysis and Risk Estimation

After executing the Monte Carlo simulation (typically over 10,000 iterations), the raw output consists of a large distribution of possible sprint counts or delivery durations required to complete the defined backlog. This raw output must be interpreted using statistical measures to generate actionable insights. The goal of this step is to quantify uncertainty, characterize risk, and communicate outcomes in a manner that is meaningful to both technical and nontechnical stakeholders.

1) Percentile-Based Risk Estimates (P10, P50, P90)

Three key percentiles are extracted from the simulation results to frame different risk perspectives:

- P10 (10th Optimistic): This value indicates an optimistic scenario, where only 10% of simulated outcomes are faster than this point. It suggests that in the best-case scenario, the project is likely to finish at least this early.
- P50 (50th percentile/median): This is the most likely estimate, meaning 50% of outcomes are faster and 50% are slower. It represents a balanced, average case used for baseline planning.
- P90 (90th Pessimistic): This value reflects a pessimistic or conservative scenario, with 90% of outcomes completing before this point. It accounts for worst-case planning, helping teams prepare for delays.

2) Confidence Intervals and Variability

In addition to percentile markers, a 95% Confidence Interval (CI) is typically computed to describe the range within which the true delivery time is expected to lie with high probability. This is often calculated as Equation (7):

$$\text{CI}_{95\%} = [\mu - 1.96 \times \sigma, \mu + 1.96 \times \sigma] \quad (8)$$

Where:

- μ = mean of simulated sprint durations
- σ = standard deviation of simulated durations

The narrower the confidence interval, the lower the uncertainty in project delivery; a wider interval indicates greater risk and variation in team performance.

3) Visualizing and Interpreting the Distribution

The distribution of simulated sprint counts or delivery days is often plotted as a histogram or cumulative density function (CDF). This allows stakeholders to visually assess the spread and skewness of delivery timelines. For example:

- A left-skewed distribution may indicate high consistency but potential for rare delays.
- A right-skewed distribution suggests frequent delays or team inefficiency.

Overlaying the P10, P50, and P90 values on these plots helps contextualize where the current delivery forecast falls within the overall risk profile.

3.8 Justification and Novelty of this methodology

The justification for this methodology stems from a fundamental limitation in traditional Agile planning, where teams often rely on simplistic or static estimates—such as average velocity or fixed sprint planning—that fail to capture the inherent variability and uncertainty present in real-world software projects. Despite widespread usage of tools like Jira, most Agile teams underutilize the wealth of historical performance data they generate, resulting in forecasting practices that are more subjective than empirical. To address this gap, the proposed methodology integrates actual Jira issue-level data—such as resolution times and backlog size— with Monte Carlo simulation, a proven statistical technique for modeling uncertainty. This combination allows for a realistic, repeatable simulation of project completion timelines by considering a wide range of possible team performance scenarios through velocity sampling and risk buffer modeling.

The novelty of this work lies in its ability to transform routine Agile project data into meaningful risk intelligence through a structured, simulation-based approach. Whereas the current practice in Agile planning and many other frameworks fails to capture the value of the historical delivery patterns, this study presents a novel aspect of using this data to generate probabilistic evidence on the future of the project results. This is not a relatively new way of doing so but the distinctive feature of this approach is that it wants to measure and tolerate delivery uncertainty, based on percentile-based forecasts, actual project dynamics, instead of making hard or frozen assumptions. Moreover, the compatibility of the framework with both Scrum and Kanban processes makes it adaptable to different Agile contexts. The combination of empirical project data and risk-aware forecasting is such an introduction to overcoming this issue and introduces a new, evidence-based aspect to Agile project management, increasing the accuracy of the planning process, instilling transparency, and finally supporting more resilient decision-making.

4. RESULT ANALYSIS AND DISCUSSION

This section presents the simulation results of the designed system. This paper will examine the risk in Agile delivery by performing data-driven simulations with the use of Jira data. Pandas, NumPy, Matplotlib, and Seaborn Python libraries were used to implement the present study on a Windows 10 operating system with an Intel Core i7 and 16 GB of memory. Historical data was collated at the issue-level through the Jira REST API, and pre-processed using Jira velocity estimation before Monte Carlo simulation was applied. The outcomes are the velocity tendencies, the proportion of completion times, and most critical percentile-based estimates (P10, P50, P90) that help to describe the volatility and uncertainty in the project delivery. The findings are further explained with the use of visualizations in the form of the histogram, cumulative distribution plot, and box plot.

Table 2. Issues In Jira Scrum Project

S.No.	Key	fields.issuetype.n ame/	resolution_time_ days
0	SCRUM-14221	Bug	674
1	SCRUM-14215	Bug	705
2	SCRUM-14211	Bug	680
3	SCRUM-14210	Bug	7

4	SCRUM-14209	Bug	7

Table 2 showcases an example of an issue data consisting of such attributes as issue key, issue type, and days to resolve. Each issue key (e.g., SCRUM-14221) uniquely identifies a task, while the issue type, such as "Bug," classifies the nature of the work. Resolution times vary widely from as little as 7 days to over 600 days, reflecting significant inconsistency in how long tasks take to complete. This variability highlights the inherent uncertainty in Agile workflows and serves as a crucial factor in calculating sprint velocity, which directly influences the accuracy and robustness of the Monte Carlo risk simulation.

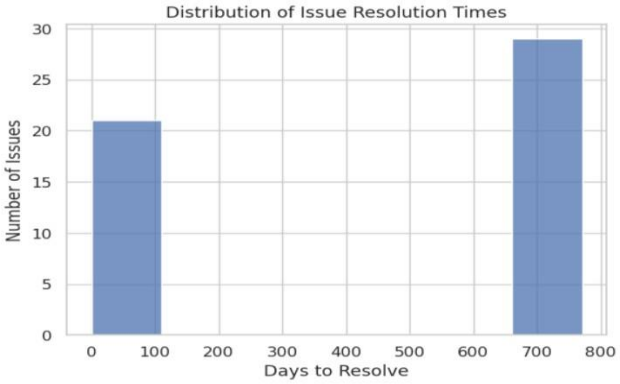

Figure 5. Histogram Distribution of Issue Resolution Times

Figure 5 shows the histogram distribution of issue resolution times that reveals a striking bimodal pattern, indicating that issues within the Jira Scrum project tend to be resolved in two distinct timeframes. The first cluster shows that approximately 21 issues were resolved quickly within 0 to 100 days, while the second, more prominent cluster indicates that about 29 issues took significantly longer between 650 and 800 days to be resolved. This clear separation, with very few issues resolved in the intermediate range (100 to 650 days), suggests inconsistencies in issue prioritization or handling. Identifying such a pattern is crucial for building more accurate and risk-aware simulations, as it highlights the need to account for both fast-moving and delayed tasks when forecasting sprint velocity and project timelines. Incorporating this variability into Monte Carlo simulations enhances the realism of delivery risk models and helps teams better anticipate and manage delays.

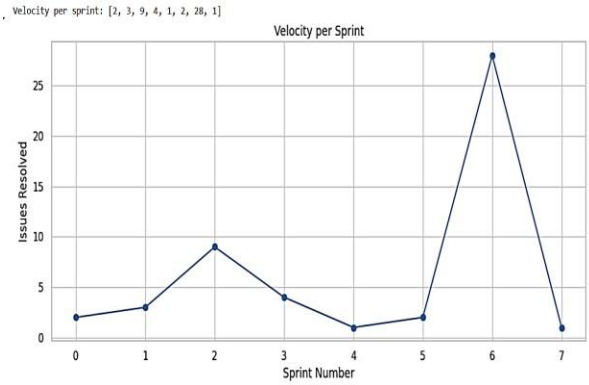

Figure 6. Line Graph of Agile Sprint Velocity

Figure 6 shows a line graph depicting sprint-wise velocity based on the number of issues resolved across eight sprints, with data points [2, 3, 9, 4, 1, 2, 28, 1]. The x-axis is the number of sprints followed by the y-axis as the number of issues that have been resolved. The graph also demonstrates the high level of oscillation of the performance of the team with significantly increased and decreased rates including the sharp upswing to 28 issues in Sprint 6 and the even stronger downswing to 1 issue in Sprint 7. This sporadic trend indicates the absence of consistency in how work is performed, perhaps due to the dynamics of the team, unequal work distribution, or rearranging priorities. Inclusion of such variability is powerful to Agile planning because it reminds against the understanding of pure velocity to make forecasts. With this actual variability entered into a Monte Carlo simulation, project managers will

be in a much better position to plan delivery schedules, design-targeted goals, and create schedules that are more open to any disruptions in team performance.

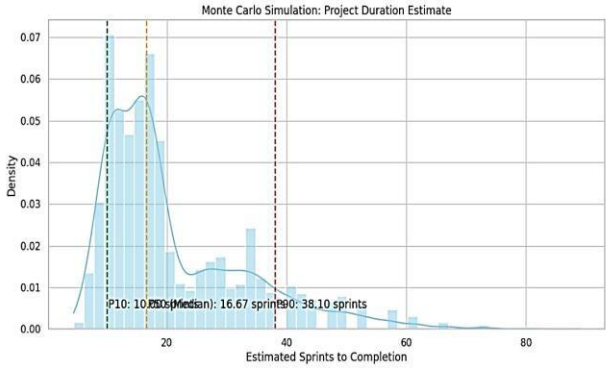

Figure 7. Histogram of MCS: Project Duration Estimate

Figure 7 presents a histogram combined with a kernel density estimate (KDE) plot derived from the Monte Carlo simulation results for project duration, based on historical sprint velocities and a fixed backlog size of 100 issues. The distribution is right-skewed, suggesting that while most simulations predict completion within 15 to 20 sprints, there is a notable probability of longer durations due to inherent variability in team performance and task complexity. Vertical dashed lines mark key percentiles: the 10th percentile (P10) at 9.88 sprints indicates an optimistic scenario with only a 10% chance of completing the project this quickly; the 50th percentile (P50) at 16.67 sprints represents the most likely outcome or median expectation; and the 90th percentile (P90) at 40.00 sprints reflects a pessimistic case accounting for delays and risk factors. This probabilistic view of project timelines equips Agile teams with a deeper understanding of delivery uncertainty, enabling them to create more resilient plans, allocate resources effectively, and communicate realistic expectations to stakeholders.

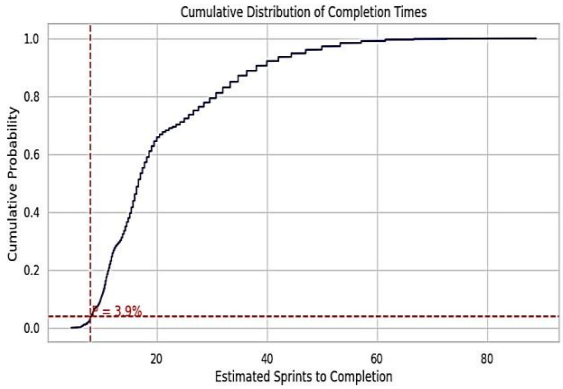

Figure 8. Line Curve of Cumulative Distribution of Completion Times

Figure 8 shows the line curve of cumulative distribution of completion times presents the probability of completing a project within a given number of sprints, based on Monte Carlo simulation results using historical velocity data and a backlog of 100 items. The curve rises sharply between 5 and 20 sprints, indicating that most projected outcomes fall within this range, while it gradually levels off beyond 40 sprints, approaching a probability of 1.0. A point that is highlighted at 7.9 sprints presents it that there is a 3.9% chance of meeting the sprints and sprints completion with an aggressive timeline is very low. This visualization will provide accurate information on uncertainties in delivery and can be used in more accurate planning since the team will know the likelihood of hitting certain sprint goals. It assist teams to be on the same page with regards to expectations, make sound decisions on scheduling and decreases chances of over promising delivery dates.

Table 3. Summary of monte carlo risk

Metric	Value (Sprints)

Optimistic (P10)	9.88
Median (P50)	16.67
Pessimistic (P90)	38.10
95% Confidence Interval	7.77 – 53.33

Table 4. Completion Probability For Target Sprints

Target Sprints	Completion Probability
6	0.2%
8	3.9%
10	11.1%
12	25.8%

Table 3 and Table 4 give an in-depth look at the results of the Monte Carlo analysis of simulated sprint outcomes. The important statistic indicators which are used to explain the delivery predictions revealed in Table 3 are that 50 percent of the simulations complete the project in around 16.67 sprints, 10 percent do it within less than 9.88 sprints, and 90 percent of them complete the project in 38.10 sprints. The confidence interval of 95 percent between 7.77 and 53.33 sprint represents the large range that can be taken between possible completion times as a result of project uncertainty. Table 4 also uses these results to provide practical values in making a decision by giving the chance of hitting certain sprint objectives. To illustrate, at 6 sprints, the possibility of delivery completion is as low as 0.2 percent and at 12 sprints, it is 25.8 percent, which suggests that the shortened deadline can prove to be excessively optimistic. These values demonstrate how historical data, and probabilistic forecasting techniques enable the project managers not to make errors on underestimating, categorizing risk buffers as priority, and in setting project milestones more wisely- leading to more realistic program schedules and an improved performance in the realization of projects.

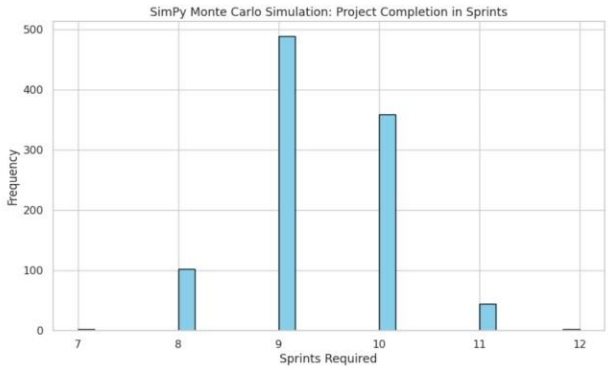

Figure 9. Histogram of SimPy MCS: Project Completion in Sprints

Figure 9 illustrates the histogram generated from the SimPy Monte Carlo simulation, depicting the distribution of project completion times in sprints. Simulation results indicate that project completion is most frequently observed between 9 and 10 sprints, with 9 sprints appearing in nearly 500 simulations and 10 sprints in over 350 cases. Minimum spread indicated by very few simulations at 7 or 12 sprints evidences a stable and controlled delivery environment even further. This clustering is further supported by percentile marks-P10 (8.00), median (P50) 9.00 and P90 (10.00) therefore indicating a highly concentrated and predictable range of delivery. This predictability decreases the uncertainty in planning which enables teams to plan better, have no or less overcommitment of the resources and make promise delivery on time. Such data-driven ideality enables project managers to shrink padding of the schedule milestones and be sure about projecting mid-range sprint goals. It also reduces the likelihood of overestimating or underusing team capacity, making sprint planning more accurate, resource allocation more efficient, and communication with the stakeholders with respect to what is expected of them in terms of delivering products or features to be more efficient.

Table 5. Enhanced monte carlo simulation summary using simpy

Metric	Value (Sprints)
Optimistic (P10)	3.45
Median (P50)	3.45
Pessimistic (P90)	3.45
95% Confidence Interval	3.45 – 4.60

Table 5 shows the result of the accelerated Monte Carlo simulation with SimPy that tries out sprint-based completion of a project with diverse team speeds. Remarkably, all three values, P10, P50, and P90, to the plan of completion have a common point, 3.45 sprints, which ensures the high rate of probability in the theoretical plan completion time. Simulation stability is bolstered by an accounting of the project duration as very predictable and practically feasible on a consistent basis at the narrow confidence interval of 95% (3.45 to 4.60 sprints). Such high accuracy favors good planning, limits risk, and enables best use of resources and stakeholder assurance.

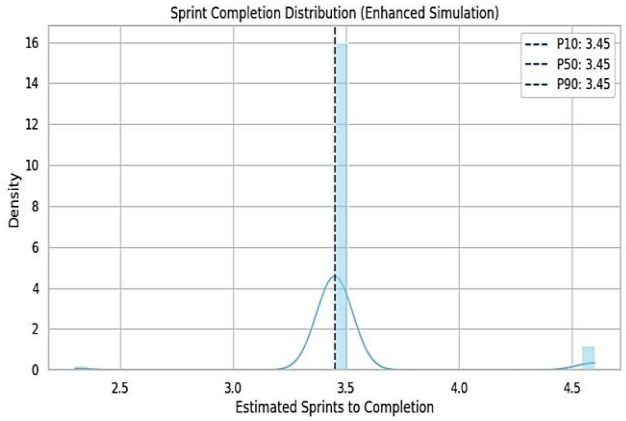

Figure 10. Histogram of Sprint completion distribution (enhanced simulation)

The histogram and KDE plot in Figure 10 display a very focused distribution of sprint completions with its center point being 3.45 sprints in the enhanced Monte Carlo simulation. The narrow high peak gives evidence of a remarkably consistent project completion estimate that is devoid of large variance, which is strengthened by the fact that P10, P50, and P90 rest at the same point. This indicates that the enhanced simulation model yields highly reliable forecasts, reducing ambiguity in planning and enabling confident scheduling decisions. Although a few outliers appear around 2.25 and 4.75 sprints, their negligible frequency further validates the robustness and predictability of the model's output. This level of precision significantly improves stakeholder confidence and facilitates accurate resource allocation. The impact of this simulation lies in its ability to minimize project delays and support agile teams in making timely, data-driven decisions.

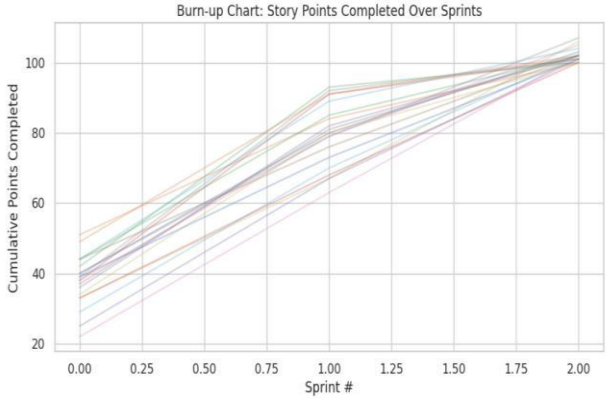

Figure 11. Line graph of burn-up chart: story points complete overprints

Figure 11 presents a burn-up chart illustrating the cumulative progress of simulated sprint outcomes. Each faint trajectory line represents a possible path from the Monte Carlo simulation, highlighting variations in team velocity and project advancement. Despite differences in early sprint performance, most lines converge between 100 and 110 cumulative story points by the end of Sprint 2, indicating a strong likelihood of completing all required work within this timeframe. This convergence not only validates the estimated story point total but also confirms the model's reliability in forecasting delivery timelines. The visual clarity and predictive power of this chart offer project managers a strategic advantage—enabling better monitoring of actual versus forecasted progress, timely interventions if deviations occur, and enhanced confidence in making delivery commitments.

4.1 Discussion

The results presented through multiple visualizations and tables reveal critical insights into delivery variability, risk, and forecasting accuracy in Agile software projects using Jira data. The significant spread in resolution times (Table 2, Figure 5) and fluctuating sprint velocities (Figure 6) emphasize the inherent unpredictability of Agile workflows, underlining the inadequacy of static planning models. Monte Carlo simulations, both standard and enhanced (Figures 7, 10, 11; Tables 3–5), effectively capture this uncertainty by providing percentile-based estimates that quantify delivery risks. For instance, the standard simulation estimated project completion between 7.77 and 53.33 sprints with a P50 of 16.67 sprints (Table 3), while the enhanced SimPy simulation delivered much tighter results with a P10, P50, and P90 all converging at 3.45 sprints and a narrow 95% confidence interval of 3.45–4.60 (Table 5), demonstrating superior consistency and reliability. Figure 8 shows only a 3.9% chance of completion within 8 sprints, reinforcing the need for probabilistic forecasting over unrealistic static plans.

The advantages of this approach lie in its ability to model real-world variability, visualize uncertainty clearly through CDFs and burn-up charts (Figures 8 and 12), and produce actionable metrics like completion probability and sprint buffers (Table 4). Unlike traditional estimation methods that produce a single deterministic value, this system empowers project managers with probabilistic insights to make informed, risk-aware decisions. The impact of these results is significant: Agile teams can improve estimation accuracy, reduce the likelihood of missed deadlines, optimize resource allocation, and increase stakeholder trust through data-driven transparency. Compared to conventional systems that rely on average sprint velocity or expert judgment, this method demonstrates superior forecasting precision and adaptability, especially in unpredictable environments. Its integration of accessible tools like Python and Jira ensures that these benefits can be realized by teams of all sizes without expensive or proprietary software, making it a highly scalable and impactful solution in the Agile project management landscape.

4.2 Solutions: Risk Assessment of Software Development in Agile

The research proposes a Monte Carlo-based simulation framework using real-world Jira Scrum and Kanban datasets to quantify and address these risks in Agile projects. Key solutions offered include:

- **Data-Driven Simulation of Uncertainty:** By sampling historical sprint velocity data, the Monte Carlo simulation accounts for variability and generates a probability distribution of delivery outcomes. This replaces risky single-point estimates with percentilebased forecasts (P10, P50, P90).
- **Backlog Completion Forecasting:** The simulation models the number of sprints required to complete a given backlog (e.g., 100 items), enabling teams to assess timelines under best-case, average, and worstcase scenarios.
- **Enhanced Visual Risk Communication:** Tools like cumulative distribution plots, box plots, and violin plots are used to communicate the range and likelihood of possible outcomes, making it easier for stakeholders to understand project risks.
- **Scrum vs. Kanban Risk Comparison:** The framework allows for empirical comparison of risk and predictability across Agile methodologies. For instance, Scrum showed wider variability than Kanban, informing decisions on process improvement or framework selection.
- **Incorporation of Risk Buffers and Sprint** Variability: Adjustable risk buffers and flexible sprint lengths are introduced into the simulation model to reflect real-world conditions and reduce the impact of uncertainty.

- **Probability-Based Planning Support:** The model answers planning questions like "What is the probability of completing the backlog within X sprints?"—helping project managers set realistic expectations and mitigate delivery risk proactively.

By integrating Monte Carlo simulation into Agile risk assessment using real Jira data, this methodology shifts Agile planning from guesswork and averages to probabilistic, datadriven forecasting. It enables teams to anticipate delays, manage uncertainty, and enhance confidence in delivering software projects on time and within scope—regardless of whether they use Scrum or Kanban.

4.3 Agile Risks Mitigation through Proposed SimulationBased Framework

The proposed simulation-based Agile risk assessment framework directly addresses and mitigates multiple categories of risks typically encountered in Agile software development, as present in Table 6.

4.4 Advantages of this Work

This study offers a practical, software development approach to Agile risk forecasting by integrating Jira project data with Monte Carlo simulation, providing percentile-based delivery estimates and visual tools that improve

Table 6. Agile software development risks to mitigation strategies by propose framework

S. N0	Risk Category	Description	Mitigation by Proposed Work
1	Organizational Risks	Poor schedule definition, inexperienced PMs, multitasking impacts.	Data-driven forecasts using empirical Jira data and Monte Carlo simulation produce realistic timelines (P10, P50, P90). Visual dashboards improve schedule clarity, resource planning, and team alignment.
2	Budget and Financial Risks	Unexpected costs, scope creep, poor budget planning.	Risk buffers and confidence intervals highlight likely overruns early. Percentile-based forecasts support contingency fund planning for scope changes or security upgrades.
3	Technology Risks	Outdated tools, technical gaps, potential quality losses.	CFDs and resolution trends reveal bottlenecks, delays, and technical debt. Early detection supports timely upgrades and process adjustments.
4	Security Risks	Data breaches, poor security controls, misuse of systems.	Continuous backlog monitoring identifies unresolved security tickets. Clear Jira audit trails support compliance and traceability for secure development workflows.
5	Human Risks	Skill gaps, staff turnover, conflicts, multitasking.	Historical team throughput trends reveal under-resourcing or burnout risks. Supports informed work balancing, realistic capacity planning, and early response to staffing changes.
6	Requirement Change Risks	Frequent scope changes from clients or teams; misunderstood requirements.	Flexible simulation reruns update forecasts when backlogs change. Risk buffers quantify impacts of scope changes on delivery timelines and budget.
7	Market and Business Risks	Unpredictable market conditions, shifting competition, ROI threats.	Improved predictability supports product launches aligned with market windows. Confidence intervals inform go-to-market timing, profit planning, and investment risk reduction.
8	Performance Risks	Delays in execution by internal teams or third parties; delivery failures.	Probabilistic modeling over 10,000 iterations quantifies worst-case scenarios (P90). Supports proactive fallback plans for third-party dependencies or capacity shortfalls.
9	Legal Risks	Contract disputes, compliance failures, legislative changes.	Clear delivery forecasts and Jira traceability help meet contractual timelines. Audit trails support compliance documentation and evidence in case of legal review.
10	Environmental Risks	Natural disasters, remote work disruptions, supply chain delays.	Risk buffers and variable sprint durations account for unforeseen delays. Probabilistic planning builds in realistic buffers for uncontrollable external factors.

planning accuracy. It enhances decision-making, identifies delivery uncertainty, and supports comparative analysis between Scrum and Kanban. The framework is scalable, cost-effective, and suitable for real-world Agile environments. The following key advantages are:

- Improves decision-making by providing Agile teams with software development insights into project risk and delivery timelines.
- Reduces delivery uncertainty through accurate and realistic forecasting based on historical project behavior.
- Builds the trust of stakeholders due to the visual, transparent presentation of the available options and delays.
- Balances agile planning and allows the flexibility of Agile environment like Scrum and Kanban.
- Minimizes project delays by enabling proactive identification of bottlenecks and incorporating risk buffers.
- Promotes cost-effective scalability as the framework leverages commonly used tools (Jira, Python), requiring no additional investment.

4.5 Limitations with Future Work

While the proposed Monte Carlo-based framework offers valuable insights into Agile project risk forecasting, it is not without limitations. The accuracy of the simulation heavily depends on the quality and completeness of historical Jira data; missing or inconsistent entries can affect the reliability of velocity distributions and resulting forecasts. Additionally, the model assumes that past team performance will remain indicative of future behavior, which may not hold true in dynamic team environments affected by personnel changes or shifting priorities. The approach also abstracts complex factors such as interdependencies between tasks, varying story point weights, or team learning curves, which are not fully captured in issue-level data alone. Lastly, while the framework is accessible and cost-effective, it still requires a baseline level of technical expertise in data extraction, Python programming, and statistical interpretation—potentially limiting adoption by non-technical Agile practitioners. These limitations can be addressed through the proposed future directions:

- Expand compatibility with other Agile tools (e.g., Azure DevOps, ClickUp, Monday.com) to test generalizability of the framework.
- Integrate machine learning models (e.g., Random Forest, XGBoost) to predict sprint velocity trends based on richer feature sets using Scikit-learn or TensorFlow.
- Incorporate story point weighting and task complexity to refine velocity estimation via Jira API enhancements and task classification algorithms.
- Model team dynamics and learning effects over time — using time-series forecasting tools like Prophet or
- ARIMA.
- Account for inter-task dependencies and parallelism in simulations — using advanced workflow modeling tools such as NetworkX or SimPy extensions.
- Develop a user-friendly dashboard for non-technical stakeholders — built with tools like Streamlit or Dash for real-time visualization and decision support.
- Validate across diverse project types and domains to generalize findings — by expanding datasets from public repositories and cross-tool integrations (e.g., Trello, Asana).
- As future work, it also expected that this study will be an input for further research that seeks the construction of a method that is aware of security throughout the software construction process and that is focused on inexperienced teams or developers.

5.CONCLUSION

Agile software development often prioritizes speed and flexibility over rigorous planning, making traditional risk management approaches difficult to implement. This study introduces a robust, simulation-based risk assessment framework that bridges this gap by integrating real-world Jira project data with Monte Carlo simulation techniques. Through detailed modeling of sprint velocity and backlog resolution, the research delivers clear, percentile-based delivery forecasts, providing Agile teams with realistic expectations of project timelines. Notably, a backlog of 100 issues yielded a median (P50) estimate of 16.67 sprints, while the enhanced SimPy simulation refined this prediction to a concentrated 3.45 sprints with high confidence. This work significantly enhances Agile planning accuracy, promotes software development decision-making, and fosters transparency in stakeholder communication. It empowers teams to manage uncertainty proactively, minimize over-promising, and allocate resources more efficiently,

all using accessible, lowcost technologies like Python and Jira's REST API. The framework stands out for its adaptability, scalability, and relevance across Scrum and Kanban methodologies. Looking ahead, this framework can be extended with AI-based predictive models, dynamic backlog prioritization, and realtime dashboard integration. These enhancements will further refine delivery forecasting and make advanced risk management more accessible to Agile teams across industries.

References:

1. Abioye, T. E., Arogundade, O. T., Misra, S., Akinwale, A. T., & Adeniran, O. J. (2020). Toward ontology-based risk management framework for software projects: An empirical study. *Journal of Software: Evolution and Process*, *32*(12). doi:10.1002/smr.2269
2. Alencar, T., Cortés, M., Veras, N., & Magno, L. (2018). A proactive approach to support risk management in software projects using multi-agent systems. In *Proceedings of the 20th International Conference on Enterprise Information Systems* (pp. 415–424). SCITEPRESS. doi:10.5220/0006703804150424
3. Alsaqqa, S., Sawalha, S., & Abdel-Nabi, H. (2020). Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, *14*(11). doi:10.3991/ijim.v14i11.13269
4. Chan, F. K. Y., & Thong, J. Y. L. (2009). Acceptance of agile methodologies: A critical review and conceptual framework. *Decision Support Systems*, *46*(4), 803–814. doi:10.1016/j.dss.2008.11.009
5. Dada, O. A., & Sanusi, I. T. (2022). The adoption of software engineering practices in a Scrum environment. *African Journal of Science, Technology, Innovation and Development*, *14*(6), 1429–1446. doi:10.1080/20421338.2021.1955431
6. Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, *85*(6), 1213-1221. doi:10.1016/j.jss.2012.02.033
7. dos Santos, J. C., Coelho, V. P. A., Fonseca, F. M., & Mendonça, D. S. (2023, November). Implementing a simplified risk management approach in agile software development: An experience report. In *Proceedings of the XXII Brazilian Symposium on Software Quality* (pp. 254–263). New York, NY: ACM. doi:10.1145/3629479.3629516
8. Edison, H., Wang, X., & Conboy, K. (2022). Comparing methods for large-scale agile software development: A systematic literature review. *IEEE Transactions on Software Engineering*, *48*(8), 2709–2731. doi:10.1109/TSE.2021.3069039
9. Garcia, F., Hauck, J., & Hahn, F. (2022, July). Managing risks in agile methods: A systematic literature mapping. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering* (pp. 394–399). doi:10.18293/SEKE2022-123
10. Ghane, K. (2017). Quantitative planning and risk management of agile software development. In *2017 IEEE Technology and Engineering Management Society Conference (TEMSCON)*. doi:10.1109/TEMSCON.2017.7998362
11. Goyal, A. (2022). Optimising software lifecycle management through predictive maintenance: Insights and best practices. *International Journal of Science and Research Archive*, *7*(2), 693–702.
12. Goyal, A. (2023). Reducing defect rates with AI-powered test engineering and JIRA automation in agile workflows. *International Journal of Current Engineering and Technology*, *13*(6), 554–560. doi:10.14741/ijcet/v.13.6.7
13. Goyal, A. (2024). Integrating blockchain for vendor coordination and agile scrum in efficient project execution. *International Journal of Innovative Science and Research Technology*, *9*(12).
14. Huss, M., Herber, D. R., & Borky, J. M. (2023). Comparing measured agile software development metrics using an agile model-based software engineering approach versus scrum only. *Software*. doi:10.3390/software2030015
15. Ibraigheeth, M. (2024, October). Assessment tool for software failure and risk mitigation in project management. In *2024 IEEE International Conference on Data and Software Engineering (ICoDSE)* (pp. 49–55). IEEE. doi:10.1109/ICoDSE63307.2024.10829890
16. Islam, A. K. M. Z., & Ferworn, D. A. (2020). A comparison between agile and traditional software development methodologies. *Global Journal of Computer Science and Technology*, 7–42. doi:10.34257/GJCSTCVOL20IS2PG7
17. Javed, M., Alam, M. H., Sohail, M., & Arif, F. (2025). Enhancing risk mitigation in global software development: Leveraging agile practices and seven risk management principles. In *2025 International Conference on Communication Technologies (ComTech)* (pp. 1–6). doi:10.1109/ComTech65062.2025.11034462
18. Kalava, S. P. (2024). Enhancing software development with AI-driven code reviews. *North American Journal of Engineering Research*, *5*(2), 1–7.
19. Khanna, E., Popli, R., & Chauhan, N. (2022). Identification and classification of risk factors in distributed agile software development. *Journal of Web Engineering*. doi:10.13052/jwe1540-9589.2164
20. Khurana, S. K., & Wassay, M. A. (2023). Towards challenges faced in agile risk management practices. In *6th International Conference on Inventive Computation Technologies (ICICT)*. doi:10.1109/ICICT57646.2023.10134188
21. Koch, A. S. (2013). *Agile software development evaluating the methods for your organization*. Artech House.
22. Kumar, R., Maheshwary, P., & T. Malche. (2019). Inside agile family software development methodologies. *International Journal of Computer Science and Engineering*. doi:10.26438/ijcse/v7i6.650660
23. Modalavalasa, G. (2021). The role of DevOps in streamlining software delivery: Key practices for seamless CI/CD. *International Journal of Advanced Research in Science, Communication and Technology*, *1*(12), 258–267. doi:10.48175/IJARSCT-8978C

24. Omar, M., et al. (2024). Enhancing agile project success: A comprehensive study of risk management approaches among Malaysian practitioners. *Journal of Software: Evolution and Process*, *36*(9). doi:10.1002/smr.2681
25. Prajapati, V. (2025). Advances in software development life cycle models: Trends and innovations for modern applications. *Journal of Global Research in Electronics and Communication*, *1*(4), 1–6.
26. Salin, H., & Lundgren, M. (2022). Towards agile cybersecurity risk management for autonomous software engineering teams. *Journal of Cybersecurity and Privacy*, *2*(2), 276–291. doi:10.3390/jcp2020015
27. Seffe, N. S., Odzaly, E. E., & Samah, K. A. F. A. (2024, December). Data-driven strategies for enhanced risk management performance in software development perspective: An agile implementation. In *2024 IEEE 22nd Student Conference on Research and Development (SCOReD)* (pp. 122–127). IEEE. doi:10.1109/SCOReD64708.2024.10872763
28. Shaout, A., & Kyer, T. (2024). VSK - A tripartite development model based on the V-model, the Scrum, and the Kanban methodologies. In *2024 25th International Arab Conference on Information Technology (ACIT)* (pp. 1–6). doi:10.1109/ACIT62805.2024.10876952
29. Sheikh, Z. A., & Singh, Y. (2023). Minimizing cost, effort, and implementation complexity for adopting security requirements in an agile development process for cyber-physical systems. In *Agile Software Development: Trends, Challenges and Applications* (pp. 87–100). doi:10.1002/9781119896838.ch6
30. Sherif, E., Helmy, W., & Galal-Edeen, G. H. (2023). Proposed framework to manage non-functional requirements in agile. *IEEE Access*, *11*, 53995–54005. doi:10.1109/ACCESS.2023.3281195
31. Singh, S., Madaan, G., Singh, A., Sood, K., Grima, S., & Rupeika-Apoga, R. (2023). The AGP model for risk management in agile I.T. projects. *Journal of Risk and Financial Management*, *16*(2). doi:10.3390/jrfm16020129
32. Spagnoletti, P., Kazemargi, N., & Prencipe, A. (2022). Agile practices and organizational agility in software ecosystems. *IEEE Transactions on Engineering Management*. doi:10.1109/TEM.2021.3110105
33. Zayat, W., & Senvar, O. (2020). Framework study for agile software development via Scrum and Kanban. *International Journal of Innovation and Technology Management*, *17*(04). doi:10.1142/S0219877020300025