International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

Machine Learning and Explainable AI for Delivery-Risk Prediction in Software Engineering: A Survey of Issue-Tracking-Based Approaches

Fikret Zajmović[1]; Dino Kečo[2]; Sead Mašetić[3]

[1, 2, 3]Department of Information Technologies, International Burch University, Bosnia and Herzegovina
[1] fikret.zajmovic@stu.ibu.edu.ba; [2] dino.keco@ibu.edu.ba; [3] sead.masetic@stu.ibu.edu.ba

DOI: https://doi.org/10.47760/ijcsmc.2026.v15i11.002

Abstract: Software teams rely on issue-tracking systems to plan, coordinate, and record their work, and these systems accumulate structured, temporal, textual, and relational data about every task, bug, and feature request. Machine learning has been applied to this data for bug-fix time estimation, issue lifetime prediction, delay prediction, issue success prediction, and blocking-bug identification, but the resulting literature is fragmented across prediction targets, datasets, feature types, models, and evaluation practices. At the same time, explainable artificial intelligence (XAI) has become important because practitioners need to understand why a model labels a work item as risky before they act on it. This structured narrative review examines issue-tracking-based approaches published between 2007 and 2025, including recent transformer- and large language model (LLM)-based work, through the lens of delivery risk, defined here as the likelihood that a work item will not be completed within the expected time or will disrupt the planned delivery process. The review organizes prior work along six dimensions: prediction targets, data sources, feature categories, learning methods, evaluation metrics, and explainability techniques. It finds that the building blocks for delivery-risk prediction are well developed but rarely combined. Bug-fix time research is mature but limited to defects; delay prediction is closest to delivery risk but comparatively small; public Jira data is rich in workflow information but underused; and XAI in software engineering remains concentrated on defect prediction. Few studies combine public issue-tracking data, early workflow signals, textual information, and explainability in one framework, and few evaluate how early risk can be detected without information leakage. Recent deep learning and LLM-based approaches improve performance on related tasks but widen the interpretability gap. The paper closes with a research agenda for early, explainable, and reproducible delivery-risk prediction.

Keywords: delivery risk prediction, issue tracking systems, mining software repositories, machine learning, explainable AI, Jira, issue lifetime prediction, software project management

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

I. INTRODUCTION

Modern software engineering is data-rich. Development teams coordinate their work through issue trackers, version control, continuous integration, code review tools, and communication platforms. Among these sources, issue-tracking systems are especially important because they document software work at the level of individual tasks, bugs, improvements, feature requests, and maintenance activities. A typical issue record combines structured attributes such as type, priority, status, assignee, component, and resolution with temporal and textual information such as creation and resolution dates, status transitions, descriptions, comments, and links to related issues. Issue trackers have therefore become a central data source in Mining Software Repositories (MSR) research [1], [2].

Researchers have used historical issue data not only to describe what has happened in a project but also to predict what will happen. Prior studies have addressed bug-fix time estimation, issue lifetime prediction, issue success prediction, priority prediction, blocking-bug prediction, and software project delay prediction [3]–[10]. More recent work applies deep learning, transformers, and large language models (LLMs) to bug-fixing time, sprint deliverability, story point estimation, and issue labeling [11]–[14]. Together these studies show that issue data carries meaningful signals about future process outcomes. This matters for delivery management, where teams need to identify risky work items before delays or bottlenecks become visible.

In this paper, *delivery risk* is the possibility that a work item will not be completed within the expected time or will create problems that affect the planned delivery process. In practice, such problems are usually handled reactively. A work item may look normal when it is created and only later become delayed because of unclear requirements, dependencies, repeated status changes, reassignments, communication problems, or unexpected complexity. By the time the problem appears on a dashboard or in a status report, the opportunity for early intervention may already be gone. This motivates predictive models that use early issue-tracking signals to flag work items with a high probability of delay.

Delivery-risk prediction is broader than bug-fix time prediction. Issue trackers contain many kinds of work items besides defects, and tasks, improvements, features, and user stories all affect delivery. A delivery-risk perspective must therefore consider structured metadata together with workflow signals such as status transitions, issue links, reassignments, and comment activity, and with textual signals in summaries, descriptions, and comments that may reveal uncertainty, urgency, or coordination difficulty [8], [15]. It also requires explanations: a prediction that an issue is “high risk” is far more useful when it states why, for example because of high transition activity or many linked issues. XAI research in software engineering has shown that interpretability is essential for trust and adoption [16]–[18], yet most of that work concerns defect prediction and maintenance rather than management-oriented problems such as delivery risk.

Several surveys cover neighbouring areas: repository mining in general [1], [2], issue-tracker mining [19], and XAI for software engineering [18], [20], [21]. None of them examines issue-level prediction from the perspective of delivery risk, and none connects prediction targets, features, models, evaluation, and explainability into a single picture for this problem. This review addresses that gap. Its purpose is not to propose a new model but to organize and critically analyze prior work so that it is clear what is known and what remains insufficiently studied. It makes four contributions:

- It organizes prior research according to its prediction targets (bug-fix time, issue lifetime, delay, issue success, blocking issues, and workflow patterns) and relates each to delivery risk.
- It compares the data sources and feature categories used in existing studies, covering structured metadata, workflow, dependency, textual, and communication features.

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

- It reviews the learning methods and evaluation practices used in issue prediction, with particular attention to class imbalance and early prediction.
- It analyzes the role of explainable AI and derives a research agenda for early, interpretable, and reproducible delivery-risk prediction.

The remainder of the paper is organized as follows. Section II describes the review method. Section III introduces the background. Sections IV to IX review prediction targets, data sources, features, learning methods, evaluation metrics, and explainability. Section X synthesizes the findings and answers the research questions, Section XI presents research gaps and future directions, Section XII discusses the limitations of the review, and Section XIII concludes.

II. Review Methodology

This paper is a structured narrative review. It does not claim the exhaustive coverage of a systematic literature review in the sense of established software engineering guidelines [22]. Instead, it applies explicit search terms, inclusion and exclusion criteria, and a classification framework to the identified literature, and it synthesizes that literature by theme rather than paper by paper. The method is reported so that readers can judge the basis and the limits of the review's findings.

A. Research Questions

Four research questions structure the review:

- **RQ1** — Which prediction targets and data sources have been used in issue-tracking-based prediction research, and how do they relate to delivery risk?
- **RQ2** — Which feature categories and machine learning methods have been applied, and what are their strengths and limitations for delivery-risk prediction?
- **RQ3** — How are such models evaluated, and how has explainable AI been used to make their predictions interpretable?
- **RQ4** — What research gaps remain for early and explainable delivery-risk prediction, and which directions should future work take?

B. Literature Identification

The literature was searched in Google Scholar, IEEE Xplore, the ACM Digital Library, ScienceDirect, SpringerLink, and arXiv. These sources were chosen because the topic spans software engineering, data mining, machine learning, and explainable AI. The search strings, grouped into four thematic clusters, are listed in Table I. Backward and forward citation tracking from the most relevant papers was used to find foundational studies and later work that built on them.

TABLE I
SEARCH TERM CLUSTERS

Cluster	Search strings
Delivery risk and delay	"software delivery risk prediction" AND "machine learning"; "issue delay prediction" AND "software engineering"
Issue resolution time	"issue lifetime prediction" AND "GitHub"; "bug fixing time prediction" AND "software engineering"; "issue success prediction" AND "software projects"

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

Cluster	Search strings
Issue-tracking data and text	"Jira" AND "machine learning" AND "issue prediction"; "public Jira dataset" AND "software engineering"; "issue tracking" AND "text mining" AND "software engineering"; "process mining" AND "Jira" AND "software engineering"
Explainability	"explainable AI" AND "software engineering"; "SHAP" AND "software defect prediction"; "explainable software engineering prediction"

C. Inclusion and Exclusion Criteria

Each candidate study was assessed against the criteria in Table II. The criteria keep the review focused on studies that use repository or issue-tracking data for a prediction, estimation, or explanation task that can be connected to issue-level delivery outcomes.

TABLE II
INCLUSION AND EXCLUSION CRITERIA

Criterion	Inclusion	Exclusion
Data	Software repositories, issue-tracking systems, or software project data	General project management without repository or issue-tracking data
Task	Prediction, classification, estimation, monitoring, or explanation related to issue resolution, delay, risk, or delivery outcomes	Organizational theory without a link to predictive analytics
Contribution	Describes a dataset, method, framework, or analytical approach connectable to issue-level prediction	No clear method or dataset; insufficient methodological detail
Source	Peer-reviewed journal or conference paper, preprint, or dataset record	Inaccessible records and duplicates
Language	English	Other languages

D. Classification Framework

Included studies were classified along the six dimensions shown in Fig. 1: prediction target, data source, feature groups, learning method, evaluation approach, and explainability. The dimensions follow the pipeline of a typical prediction study, from the data that is collected to the explanation that is shown to a practitioner, and they structure Sections IV to IX. Table III summarizes the main primary studies along these dimensions.

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

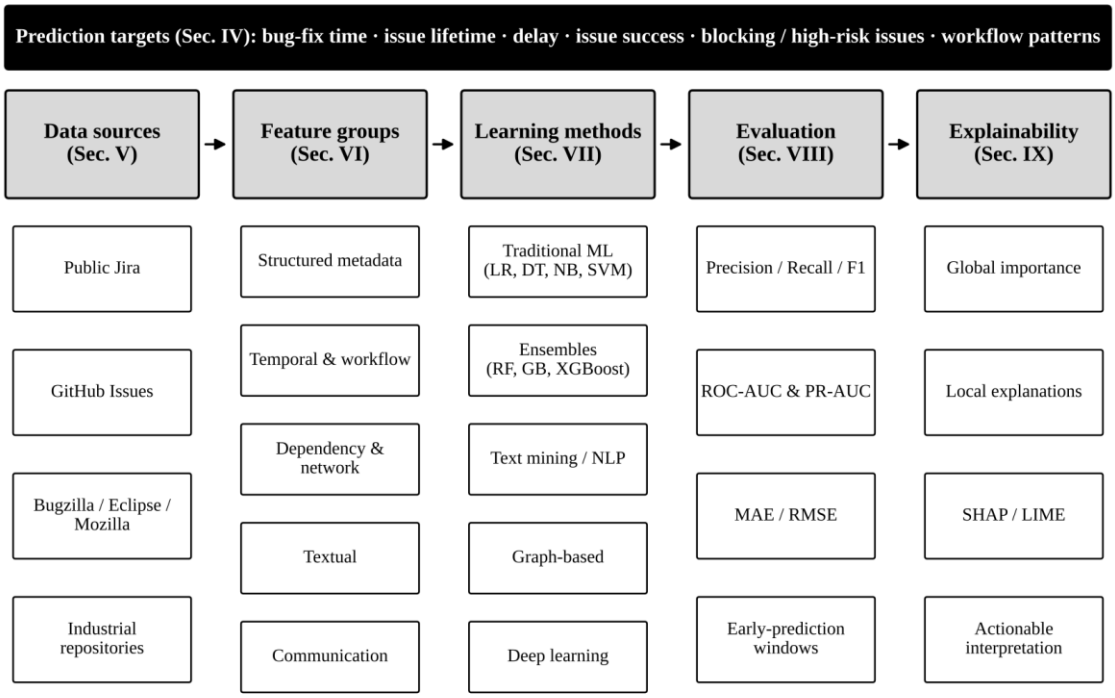


Fig. 1 Classification framework of the review

III. BACKGROUND

A. *Mining Software Repositories*

MSR is a research area within empirical software engineering that extracts knowledge from development artifacts such as version histories, bug reports, issue trackers, mailing lists, code review discussions, and commit messages. Its premise is that software projects continuously leave digital traces, and that these traces can be analyzed to understand software evolution, developer behaviour, quality, and maintenance processes, and to anticipate future outcomes. Kagdi et al. [1] provided one of the foundational surveys of the area, showing how repository mining supports evolution analysis, traceability, and change understanding. Jung et al. [2] later described MSR as an interdisciplinary field that combines software engineering, data mining, and information retrieval. Over time the field has moved from describing what happened in a repository towards predicting what is likely to happen, including management-oriented questions such as which issues will be delayed. Delivery risk fits this tradition, but it is not a raw variable in any dataset: it must be inferred from issue histories, timestamps, status changes, text, and dependencies.

B. *Issue-Tracking Systems as Data Sources*

Issue trackers represent software work at the level of individual work items. Unlike source code repositories, which mainly capture code changes, they record the planning, discussion, prioritization, coordination, and resolution of work. A typical record contains an identifier, project, type, priority, status, component, assignee,

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

reporter, creation and resolution dates, and resolution status, together with free text and, on many platforms, a full change history of status transitions, reassignments, priority changes, and issue links. Montgomery et al. [19] describe issue trackers as complex socio-technical systems that support many lifecycle activities, in which developers, testers, product owners, and users coordinate their work. This is relevant for delivery risk because delays often arise from the interaction between technical and coordination complexity: an issue may be late not only because it is hard but also because its requirements are unclear, it depends on other work, or ownership keeps changing.

The combination of data types in one system is what makes issue trackers useful for machine learning. Metadata describes basic properties, workflow histories show how an item moved through the process, text carries semantic and communication signals, and links express dependencies. Each of these can capture a different facet of delivery risk: priority reflects perceived importance, transitions reflect process instability, comments reflect communication intensity, and links reflect dependency burden.

C. Delivery Risk as a Prediction Problem

In the literature, delivery-related outcomes appear under several names: bug-fix time, issue lifetime, delay, issue success, and blocking bugs. These targets differ, but they share the assumption that historical issue data contains signals about future outcomes. Issue trackers capture both the creation and the evolution of a work item, which makes it possible to distinguish *static* predictors, available when the issue is created, from *dynamic* predictors, which emerge as the issue changes over time. Dynamic and contextual features have been shown to improve prediction compared with static information alone [7].

This distinction has a practical consequence. A model that uses information from the whole lifecycle can explain why an issue was late, but it cannot warn anyone in time. A model restricted to information available early in the lifecycle can support decisions such as adding resources, clarifying requirements, resolving dependencies, or escalating coordination problems. Because issue records contain human-readable fields and interpretable process events, predictions can also be linked back to understandable factors, which connects delivery-risk prediction with explainable AI.

IV. PREDICTION TARGETS IN EXISTING RESEARCH

Issue-tracking-based prediction research addresses several related targets. They are rarely called “delivery risk”, but all of them ask whether a work item is likely to become difficult, delayed, unresolved, or disruptive.

A. Bug-Fix Time Prediction

Bug-fix time prediction is one of the oldest and most established streams. Its goal is to estimate how long a reported bug will take to resolve, which affects planning, maintenance, release management, and customer satisfaction. Giger et al. [3] showed that bug report attributes can distinguish quickly fixed from slowly fixed bugs, establishing time-related outcomes as a prediction target. Bhattacharya and Neamtiu [4] examined whether richer repository information could produce more accurate models and found the problem harder than expected. Zhang et al. [23] showed that the task is relevant in commercial projects, and the replication study of Akbarinasaji et al. [24] highlighted that model behaviour depends strongly on the dataset and project context. Recent work continues to analyse bugs with the aim of predicting resolution time [25]. Noyori et al. [11] trained a convolutional neural network on more than 36,000 Bugzilla reports and used Grad-CAM to extract the word sequences behind each prediction, finding that reports of quickly fixed bugs tend to be concrete while those of slowly fixed bugs tend to be abstract, an early example of combining deep learning with explanation in this stream. For delivery risk, this stream provides the methodological foundation: historical issue data can predict whether a work item will take longer than expected. Its limitation is that it is restricted to defects.

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

B. Issue Lifetime Prediction

Issue lifetime prediction applies the same idea to all issue types and asks how long an issue will stay open or whether it will be closed within a given period. Kikas et al. [26] showed that GitHub issues follow distinct lifecycle patterns, and later used dynamic and contextual features to predict issue lifetime, finding that information about how an issue evolves improves prediction over static metadata [7]. Rees-Jones et al. [27] argued that simpler, more interpretable models can be competitive for this task. Nastos et al. [28] recently proposed an interpretable, per-project analysis of issue resolution time that separates actual development effort from administrative delay, combining topic modelling of issue text with metadata such as components, labels, priority, type, and assignees. Two lessons transfer to delivery risk: future issue duration is predictable from issue data, and dynamic features matter. A work item that looks harmless at creation can become risky as transitions, comments, reassignments, and dependencies accumulate.

C. Software Project Delay Prediction

Delay prediction is the stream most directly related to delivery risk, because it asks whether work will be completed later than planned rather than how long it will take. Choetkiertikul et al. [5] characterized and predicted issue-related risks, framing issue-level prediction in terms of risk rather than duration. They then proposed networked classification for predicting delays, showing that relationships between tasks improve predictive performance because delays propagate through dependencies [6]. The same group predicted the delay of issues with due dates [29] and the delivery capability of iterations in agile projects [30], moving the analysis from single issues to sprint-level delivery. Sprint2Vec [12] extends this line with deep learning: it combines sprint, issue, and developer attributes with features learned by recurrent networks to predict whether a sprint will deliver its committed work and how much rework it will cause, outperforming baselines on five open-source projects. These studies are the closest conceptual precedent for delivery-risk prediction, yet the stream remains small compared with defect and bug-fix time research.

D. Issue Success and Resolution Outcome Prediction

Another group of studies predicts whether an issue will reach a successful outcome. Ramírez-Mora et al. [8] used issue descriptions and comments to predict issue success and showed that text adds meaningful predictive value. In a follow-up study they examined the communication functions of comments during bug fixing, showing that comments serve explanation, clarification, negotiation, and coordination [15]. For delivery risk, this suggests that risk may be visible in how an issue is written and discussed, and it supports the use of textual and communication features. Related work on predicting the objective and priority of issue reports from their text [9] shows that the same textual signals can recover how important and what kind of work an issue represents.

E. Blocking Bugs and High-Risk Issues

Blocking bugs prevent other work from progressing and are therefore particularly disruptive. Valdivia Garcia and Shihab [10] characterized and predicted blocking bugs in open-source projects, showing that issue data can identify items with a disproportionate effect on progress. This broadens the notion of delivery risk beyond duration: an item can be risky because it blocks other work, causes rework, or requires coordination across contributors.

F. Process and Workflow Mining

A complementary stream analyses the workflow recorded in issue histories rather than predicting a single outcome. Coremans et al. [31] applied process mining to Jira issues at a large company and showed that status histories reveal handoffs, loops, and bottlenecks. Wang et al. [32] mined micro-process patterns of issue

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

resolution in open-source projects and used them for prediction, and Reszka et al. [33] combined several repository dimensions to improve software project monitoring. This work is mostly descriptive, but it supplies workflow-aware features such as time in status and handoff counts, and it supports the idea of "delivery intelligence": using repository data to anticipate problems rather than only to report status.

G. Related Issue-Level Tasks: Effort, Labels, and Links

Several adjacent tasks supply methods that transfer directly to delivery-risk prediction. Story point estimation predicts the effort of a work item from its text: GPT2SP [13] fine-tunes a GPT-2 transformer and reports clear gains over earlier deep learning baselines both within and across projects. Colavito et al. [14] showed that GPT-like LLMs can classify issue reports with little or no labelled training data, which lowers the cost of deriving text-based features such as issue intent. Lüders et al. [34] analysed more than 600,000 links in 15 public Jira repositories and showed that link types beyond duplication can be predicted, which matters because dependency features are only as good as the recorded links. Together these studies show that transformers and LLMs are now standard tools for issue text, although they have not yet been applied to delivery risk itself.

TABLE III
OVERVIEW OF REPRESENTATIVE PRIMARY STUDIES

Study	Prediction target	Data source	Main features	Method / contribution
Giger et al. [3]	Bug-fix time (fast/slow)	Bugzilla (Eclipse, Mozilla, Gnome)	Initial and post-submission bug attributes	Decision trees; fix time is predictable from report data
Bhattacharya & Neamtiu [4]	Bug-fix time	Bugzilla (open source)	Bug report and reporter attributes	Multivariate regression; limited predictive power
Zhang et al. [23]	Bug-fixing time	Commercial projects	Bug report attributes	Empirical study in industry
Akbarinasaji et al. [24]	Bug-fixing time	Open-source project	Bug report attributes	Replication; results are context-dependent
Kikas et al. [7]	Issue lifetime	GitHub Issues	Static, dynamic, and contextual features	Random forest; dynamic features improve accuracy
Rees-Jones et al. [27]	Issue lifetime	GitHub Issues	Small set of issue features	Simpler models are competitive
Choetkiertikul et al. [5]	Issue-related risk	Jira (open source)	Issue attributes	Risk characterization and prediction
Choetkiertikul et al. [6]	Issue delay	Jira (open source)	Issue attributes and task relations	Networked classification
Choetkiertikul et al. [30]	Iteration delivery capability	Jira (agile projects)	Iteration and issue features	Predicts sprint-level delivery
Ramírez-Mora et al. [8]	Issue success	Issue trackers	Descriptions and comments	Text-based classification
Valdivia Garcia & Shihab [10]	Blocking bugs	Open-source bug trackers	Bug report attributes	Classification of disruptive bugs
Coremans et al. [31]	Workflow behaviour	Jira (industrial)	Status histories	Process mining; descriptive

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

Study	Prediction target	Data source	Main features	Method / contribution
Noyori et al. [11]	Bug-fixing time	Bugzilla	Bug report text	CNN with Grad-CAM explanations
Nastos et al. [28]	Issue resolution time	Issue trackers (multiple projects)	Topics and metadata	Interpretable per-project analysis
Choetkiertikul et al. [12]	Sprint deliverability and rework	Jira (5 open-source projects)	Sprint, issue, and developer features	Sprint2Vec; LSTM-learned features
Fu & Tantithamthavorn [13]	Story points (effort)	16 open-source projects	Issue text	GPT-2 transformer
Colavito et al. [14]	Issue type labels	GitHub issues	Issue text	GPT-like LLMs, few or no labels
Lüders et al. [34]	Issue link types	15 public Jira repositories	Issue text and links	Link-type analysis and prediction
Montgomery et al. [35]	— (dataset)	16 public Jira repositories	Issues, changes, comments, links	Public dataset enabling reproducible research

V. DATASETS AND DATA SOURCES

The choice of dataset shapes what can be predicted. Because delivery-risk prediction depends on how work items evolve, the dataset must allow the history of each item to be reconstructed. Prior work has relied on four kinds of sources, compared in Table IV.

A. Public Jira Datasets

Jira is widely used for project management, sprint planning, and workflow coordination, and Jira data typically includes issue types, priorities, statuses, workflow transitions, comments, assignees, components, issue links, and resolutions. Montgomery et al. [35] released a dataset of 16 public Jira repositories covering roughly 2.7 million issues together with their change histories, comments, and links, which makes Jira-based research reproducible without private company data. Its limitations are practical: the data is large and costly to process, it lacks organizational context such as team structure and business priorities, and personal information is anonymized. Public Jira data is therefore strong for issue-level and workflow-level prediction but unsuitable for studying individual contributors.

B. GitHub Issues

GitHub Issues is accessible and widely used in open-source projects, offering labels, comments, timestamps, participants, and closing dates. It has been used to study issue dynamics and lifetime [7], [26]. However, GitHub workflows are often simpler and less standardized than Jira workflows, and detailed status transitions and typed dependencies are frequently unavailable. GitHub data suits lifetime and collaboration research, while Jira is better suited to workflow-based delivery analysis.

C. Bugzilla, Eclipse, and Mozilla Repositories

Bug repositories of projects such as Mozilla and Eclipse underpin much of the bug-fix time literature [3], [4], [24]. They contain severity, priority, component, reporter, status, comments, and resolution dates, and they provide a strong empirical basis for studying defect resolution. Their main limitation for delivery risk is that they are bug-focused and do not represent tasks, features, and improvements.

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

D. Private and Industrial Datasets

Industrial datasets reflect real organizational environments and may include sprint, release, team, and code review data [23], [31]. Such context could be highly valuable for delivery-risk prediction, but the data is usually confidential, which limits reproducibility, and results may be tied to one company's processes. Public issue-tracking datasets therefore remain essential for reproducible academic research.

TABLE IV
COMPARISON OF DATA SOURCES USED IN ISSUE-TRACKING-BASED PREDICTION

Source	Typical data	Strengths	Limitations	Relevance to delivery risk
Public Jira dataset	Issues, comments, status histories, changes, issue links	Large, public, workflow-rich, reproducible	Very large; needs filtering; personal data anonymized	High: contains workflow, textual, and dependency signals
GitHub Issues	Issues, labels, comments, timestamps, participants	Accessible, widely used in open source	Less structured workflow than Jira	Medium: suited to lifetime and collaboration analysis
Bugzilla / Eclipse / Mozilla	Bug reports, severity, priority, comments, resolution dates	Strong basis for bug-fix time research	Bug-focused; not representative of all work	Medium: methodological precursor
Industrial datasets	Internal issues, sprint, release, and team data	Realistic, rich organizational context	Confidential, not reproducible, limited access	High in practice; low for reproducible research

No single source is ideal. For delivery-risk research, the most useful dataset combines structured metadata, temporal information, workflow history, and text. Public Jira data offers the best balance between realism, richness, and reproducibility, which makes it a natural basis for future work that combines early workflow signals, textual information, and explainability.

VI. FEATURE TYPES USED FOR PREDICTION

Prediction quality depends as much on how issue data is turned into variables as on the learning algorithm. Existing studies use five broad feature groups, summarized in Table V.

A. Structured Metadata Features

Structured metadata such as type, priority, severity, component, project, labels, reporter, and assignee is the most basic and most common input. It is easy to extract and encode and usually easy to interpret, which is why early bug-fix time studies relied on it [3], [4]. On its own, however, metadata misses process behaviour: an issue with ordinary priority can still be delayed by dependencies or repeated reassignment. Metadata is therefore a baseline that needs to be complemented by richer features.

B. Temporal and Workflow Features

Temporal and workflow features describe how an issue moves through its lifecycle: issue age, time to first response, number of status transitions, time in each status, reopenings, reassignments, and activity in the first days. Lifetime and delay studies show that such dynamic information improves prediction [6], [7], and process mining studies show that it reveals bottlenecks [31], [32]. For delivery risk the key requirement is temporal validity. Features computed over the full lifecycle may describe a delay that has already happened, whereas features restricted to an early window, such as first status changes, early comments, initial reassignment, or

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

early links, support proactive decisions. Future studies should state explicitly which features are available at prediction time.

C. Dependency and Network Features

Work items depend on each other, block each other, share components, and belong to larger features or releases. Dependency features include the number of linked issues, blocking relationships, duplicates, parent-child relations, dependency depth, and network centrality. Choetkiertikul et al. [6] showed that modelling task relations improves delay prediction over treating each issue in isolation. In Jira, links such as "blocks", "is blocked by", and "relates to" make such features directly available. Their quality, however, depends on how consistently teams record links, and some dependencies exist only informally. Link-type prediction on public Jira data [34] offers a way to recover or correct missing relations before they are used as features.

D. Textual Features

Summaries, descriptions, and comments can reveal uncertainty, unclear requirements, technical complexity, urgency, or the need for investigation, none of which is captured by structured fields. Text has been shown to predict issue success [8] and issue priority [9]. Common representations range from bag-of-words and TF-IDF through topic models [28] to word embeddings and transformer-based encoders [13], and LLMs can now produce text-derived labels with little training data [14]. Simpler representations are cheaper and easier to interpret, while embeddings capture meaning better at the cost of computation and transparency. All of them require careful preprocessing.

E. Communication and Interaction Features

Communication features describe how people interact around an issue: comment count, number of participants, discussion length, time to first comment, and comment frequency. They can be computed from logs without understanding the text. Comments are a functional part of problem solving rather than noise [15], but their interpretation is ambiguous: many early comments may signal unclear requirements or disagreement, or simply healthy collaboration. These features are most informative in combination with workflow indicators and model explanations.

TABLE V
FEATURE CATEGORIES USED IN ISSUE-LEVEL PREDICTION

Category	Example features	Purpose	Main limitation
Structured metadata	Type, priority, severity, component, labels, project	Basic properties and initial context	Misses process complexity and later evolution
Temporal	Issue age, time to first response, resolution duration	Time-related behaviour	Some values are known only after the outcome (leakage risk)
Workflow	Status transitions, reassignments, reopenings, time in status	Process instability and workflow complexity	Requires detailed change history
Dependency / network	Linked issues, blockers, duplicates, centrality	Coordination burden and relations between items	Links may be incomplete or inconsistent

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

Category	Example features	Purpose	Main limitation
Textual	TF-IDF terms, topics, embeddings of summary, description, comments	Semantics, complexity, uncertainty	Preprocessing cost; higher model complexity
Communication	Comment count, participants, discussion length, response time	Collaboration and interaction intensity	High activity may mean complexity or good collaboration

Because delivery risk can emerge from any of these sources, a model that relies on one group is likely to miss part of the risk profile. Combining groups is therefore preferable, and it also enables more useful explanations: if a model uses several groups, explanation methods can show whether a prediction is driven by workflow instability, dependency burden, textual complexity, or communication intensity.

VII. MACHINE LEARNING METHODS

Issue prediction has been formulated as classification, regression, ranking, text mining, and network learning. Because delivery risk usually results from several interacting factors, the choice of method should follow the data and the purpose of the prediction.

A. Traditional Classification and Regression Models

Logistic regression, decision trees, naive Bayes, and support vector machines (SVMs) are widely used because they are simple, efficient, and suitable for structured data. Logistic regression is the usual interpretable baseline: its coefficients show whether a feature increases or decreases risk, but it cannot capture non-linear interactions such as the joint effect of many comments, many transitions, and many links. Decision trees express non-linear rules in readable form but are unstable and prone to overfitting. Naive Bayes is efficient for high-dimensional text but assumes feature independence, and SVMs perform well on text but are less transparent and require tuning. These models remain important as baselines against which the added complexity of other methods must be justified.

B. Ensemble Learning Methods

Ensembles combine many base learners and capture non-linear relationships and interactions in mixed, noisy data. Random forests [36] are robust, require little tuning, and provide feature importance scores; Kikas et al. [7] used them for issue lifetime prediction. Gradient boosting, and XGBoost [37] in particular, often achieves the best results on tabular data. The price is reduced transparency, which is precisely why ensembles are usually paired with post-hoc explanation methods such as SHAP or LIME in practical settings.

C. Text Mining and Natural Language Processing

Text mining turns issue text into features. TF-IDF weights terms that are distinctive for a document and can surface words associated with blocking, investigation, migration, or failure; topic models group issues into themes such as performance, deployment, or dependency conflicts; embeddings and transformer models capture semantic similarity even when different words are used, and fine-tuned transformers now outperform earlier approaches on issue-level estimation tasks [13]. LLMs can be applied in a zero- or few-shot fashion [14], which is attractive for projects without labelled history but raises cost, reproducibility, and data privacy concerns. Text has proven useful for predicting issue success and priority [8], [9], but it adds dimensionality, preprocessing effort, and interpretability challenges, so its contribution should be measured against a non-textual baseline.

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

D. Network-Based and Graph-Based Approaches

When relations between issues matter, issues can be modelled as nodes and links as edges. Networked classification improved delay prediction because delays propagate through task networks [6]. Graph features such as link counts, centrality, and blocking relations quantify coordination complexity. Their effectiveness depends on the completeness of recorded links.

E. Deep Learning Approaches

Deep learning can represent issue text, model sequences of issue events, and combine heterogeneous inputs, and has been applied to bug-fixing time [11] and to sprint-level delivery prediction [12], [30]. It typically requires large datasets, considerable computation, and careful tuning, and it reduces interpretability. In delivery management, stakeholders may prefer a slightly less accurate but understandable model to an opaque one, so interpretable traditional and ensemble methods remain a sensible starting point.

F. Implications for Model Selection

No method is universally best. Traditional models provide interpretable baselines, ensembles usually perform best on structured and mixed features, NLP is needed when text is used, graph methods help when dependencies matter, and deep learning suits large text-rich datasets. A robust study design therefore compares several model families, for example logistic regression, random forest, and gradient boosting, optionally extended with text and dependency features. When the goal is decision support rather than accuracy alone, the preferred model is the one that balances predictive performance, interpretability, and practical usefulness.

VIII. Evaluation Metrics

Evaluation determines how results are interpreted. Bug-fix time is often treated as regression, evaluated with Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE). Delivery-risk prediction is more often treated as classification, where precision, recall, F1-score, ROC-AUC, and PR-AUC are appropriate. A recurring difficulty is class imbalance: in most projects, only a minority of issues are substantially delayed.

A. Accuracy and Its Limitations

Accuracy is the share of correct predictions. If only 15% of issues are delayed, a model that labels every issue "on time" reaches 85% accuracy while detecting no risky item at all. Since missing a risky item is usually more costly than a false alarm, accuracy should only be reported alongside more informative metrics.

B. Precision, Recall, and F1-score

Precision measures how many issues flagged as risky are in fact risky; low precision produces false alarms that erode trust. Recall measures how many risky issues are detected; low recall undermines the purpose of an early-warning system. The F1-score, their harmonic mean, balances the two, but the right trade-off depends on context: a manager for whom missed delays are costly will favour recall, while a team overwhelmed by warnings will favour precision.

C. ROC-AUC and PR-AUC

ROC-AUC measures how well a model ranks risky issues above non-risky ones across all thresholds, which is useful when risk scores are used for prioritization. Under strong class imbalance, however, ROC-AUC can be overly optimistic, and the precision-recall curve and its area (PR-AUC) are more informative about performance on the minority class [38], [39]. Reporting both gives a more complete picture.

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

D. Regression Metrics for Time-Based Prediction

When resolution time is predicted directly, MAE and RMSE measure the gap between predicted and actual durations. Regression supports planning but must be thresholded to yield actionable risk categories. Conversely, classification labels are often derived from historical resolution-time thresholds. The two formulations answer different questions: "how long will this issue take?" versus "is this issue likely to be delayed?"

E. Early Prediction and Information Leakage

The timing of a prediction is as important as its accuracy. Early prediction can be evaluated by training and testing models on information available at fixed points, for example at creation, after 24 hours, and after 72 hours, and by comparing performance across these windows. Many studies do not state when each feature becomes available, and some use values that are known only after resolution. This causes information leakage: the model performs well in evaluation but would fail in a real forecasting setting. Delivery-risk studies should therefore restrict features to the prediction window and use time-aware data splits, training on earlier issues and testing on later ones. Evidence from defect prediction shows why: under time-aware evaluation, performance varies considerably between periods, so results obtained with random splits may not hold for the next release [40].

In summary, a sound evaluation of delivery-risk models reports precision, recall, and F1-score for the risky class, ROC-AUC and PR-AUC for ranking under imbalance, a confusion matrix, regression metrics where duration is predicted, and performance at several prediction windows. Because predictions are meant for project managers and engineering leads, evaluation should also consider whether the outputs can be explained, which is the subject of the next section.

IX. Explainable AI in Software Engineering Prediction

Machine learning supports many software engineering tasks, including defect prediction, effort estimation, bug triage, and issue classification. Its practical value is limited if developers and managers cannot understand how predictions are produced, because these predictions inform planning, prioritization, and resource allocation. Tantithamthavorn and Jiarpakdee [16] argue that practitioners need explanations that help them act, not only accurate predictions. For delivery risk this is essential: a manager who sees that an issue is risky needs to know whether the cause is dependencies, frequent status changes, an unclear description, intense discussion, or repeated reassignment.

A. Global and Local Explanations

Global explanations describe a model's overall behaviour, for instance that status transitions, link count, and description length are the strongest predictors of delay across a project. Local explanations describe a single prediction, for instance that one particular issue was flagged because of several dependencies, high early comment activity, and repeated status changes. Global explanations support strategic understanding of risk patterns, while local explanations support day-to-day decisions about specific work items; a delivery-risk system should offer both.

B. Common Explainability Techniques

Feature importance in tree ensembles is simple and widely used, but it describes the model as a whole rather than individual predictions. LIME [41] explains a single prediction by fitting a simple surrogate model around it; it works with any classifier, but its explanations depend on sampling and parameter choices. SHAP [42], based on Shapley values from cooperative game theory, assigns each feature a contribution to each prediction and can be aggregated into global importance. Rule-based local techniques such as PyExplainer [43] go a step further and express explanations as conditions that would change the prediction. For delivery risk, SHAP can

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

show whether a high transition count, a long description, many links, or high comment activity raised or lowered the predicted risk of a given issue.

C. Explainability in Defect Prediction and Maintenance

Most XAI research in software engineering concerns defect prediction and maintenance. Jiarpakdee et al. evaluated model-agnostic techniques for explaining individual defect predictions [44] and found that practitioners consider visual explanations useful and understandable [17]. Tantithamthavorn et al. [45] argue that analytics should go further and tell practitioners what to do, turning predictions into guidance on what to change and what to avoid. Systematic reviews confirm that XAI in software engineering is growing but unevenly distributed: Mohammadkhani et al. [18] found that most studies address maintenance tasks, especially defect prediction, and Cao et al. [20] and Arora et al. [21] describe a fragmented, methodologically diverse field in which management-oriented phases of the lifecycle receive comparatively little attention. Explanations for delivery risk would also need to differ in content, focusing on workflow signals, dependencies, textual complexity, and coordination patterns rather than on code metrics.

D. Challenges and Limitations

Explanations are not always stable: Shin et al. [46] showed that explanations of defect prediction models can vary with model choice, preprocessing, and settings, and Rajbahadur et al. [47] found that different feature importance methods often produce different rankings for the same classifier, particularly when features interact. XAI output should therefore be treated as an interpretive aid rather than ground truth. Different stakeholders also need different explanations; a statement that "transition count increased risk" is useful to a data scientist, whereas a manager needs an actionable reading such as "ownership or workflow may need clarification". Explanations reveal associations, not causes: comment count may correlate with delay because difficult issues need more discussion, not because comments cause delay. Finally, issue data contains traces of individual behaviour. Even with anonymized public data, research should focus on issue-, workflow-, and project-level signals and avoid profiling individuals.

E. Explainability for Delivery-Risk Prediction

For delivery risk, explainability should be part of the modelling framework rather than an afterthought, because the purpose of prediction is to support action. If a high-risk prediction is driven mainly by dependency features, the appropriate response may be dependency management or cross-team coordination; if it is driven by reassignments, ownership may need clarifying; if it is driven by textual complexity or many early comments, the requirements may need refinement. Explanations thus turn a risk score into delivery insight. This kind of management-oriented XAI remains underdeveloped compared with defect prediction.

X. DISCUSSION

This section synthesizes the findings and answers the research questions. Table VI summarizes the reviewed research directions.

A. RQ1: Prediction Targets and Data Sources

Issue-tracking research has produced several related but largely separate streams. Bug-fix time prediction is the most mature, but it is limited to defects and mostly based on Bugzilla-style repositories. Issue lifetime prediction covers all issue types and established the value of dynamic features, mainly on GitHub data. Delay prediction is the closest to delivery risk, and is the only stream that reaches sprint-level delivery [12], [30], but it is a small body of work. Issue success, blocking-bug, and process mining studies add textual, disruption, and

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

workflow perspectives. Public Jira data, which uniquely combines workflow histories, links, and text in a reproducible form [35], has been used far less than its richness would suggest.

B. RQ2: Features and Learning Methods

Feature use has shifted from static metadata towards temporal, workflow, textual, and relational features, reflecting the recognition that risk emerges as an issue evolves. Models range from interpretable baselines to ensembles, NLP, graph methods, and deep learning. Ensembles are the pragmatic default for mixed tabular data. Since about 2020, deep learning, transformers, and LLMs have become common for issue text [11]–[14], which improves performance but makes explanation even more necessary. Few studies integrate all feature groups in a single model.

C. RQ3: Evaluation and Explainability

Evaluation practice is inconsistent. Regression and classification formulations coexist, accuracy is still reported without metrics that account for imbalance, and the time at which features become available is rarely stated, so leakage cannot be ruled out. XAI is well studied for defect prediction, where model-agnostic techniques, rule-based explanations, and practitioner studies exist [43]–[45], but has seldom been applied to delay or delivery prediction; interpretable resolution-time analysis is only beginning to appear [28].

TABLE VI
SUMMARY OF REVIEWED RESEARCH DIRECTIONS

Direction	Target	Typical data	Common features	Common methods	Contribution to delivery risk	Main limitation
Bug-fix time [3], [4], [11], [23], [24]	Time to fix a bug	Bugzilla, Eclipse, Mozilla, industrial	Severity, priority, component, comments, reporter	Regression, classification, traditional ML	Time outcomes are predictable from repository data	Limited to bugs
Issue lifetime [7], [26]–[28]	Time an issue stays open	GitHub Issues	Issue age, comments, labels, dynamic context	Classification, ensembles	Value of dynamic and contextual features	Not framed as delivery risk
Delay prediction [5], [6], [12], [29], [30]	Whether work is late	Jira issue and sprint data	Task relations, dependencies, issue attributes	Networked classification, ML, deep learning	Direct delivery framing; role of relations	Small body of work
Issue success [8], [15]	Successful resolution	Issue descriptions and comments	Text, comments, metadata	NLP, classification	Text carries predictive signals	Outcome definitions vary
Blocking issues [10]	Whether an issue blocks progress	Open-source bug trackers	Severity, priority, links, comments	Classification	Identifies disruptive items	Narrow issue type
Workflow mining [31]–[33]	Process patterns	Jira, issue histories	Transitions, time in status, handoffs	Process mining, repository mining	Workflow-aware view of risk	Mostly descriptive
XAI in SE [16]–[18], [43], [44], [46]	Explaining predictions	Defect and SE datasets	Feature importance, model outputs	SHAP, LIME, visual explanations	Interpretable, actionable output	Rarely applied to delivery risk

Overall, the field has the necessary building blocks but rarely combines them. Bug-fix time research contributes time-based modelling, lifetime research contributes dynamic features, delay research contributes the delivery framing and relational modelling, text-based research contributes semantic and communication signals, process mining contributes workflow awareness, and XAI contributes interpretability. The main opportunity lies

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

in integrating them into one framework for early and explainable delivery-risk prediction on public issue-tracking data. RQ4 is answered in the next section.

XI. Research Gaps and Future Directions

A. Delivery Risk Is Rarely a Direct Target

Most studies address narrower problems such as bug-fix time, lifetime, success, or blocking. Delivery risk also concerns whether a work item disrupts planning, creates coordination problems, or requires intervention. Future work should define delivery risk explicitly, operationalize it with project-relative thresholds rather than one universal cut-off, and study bugs, tasks, improvements, and features together.

B. Early Prediction Is Under-Evaluated

Models trained on full-lifecycle data explain delays but cannot prevent them. Future studies should evaluate performance at defined windows, such as at creation and after 24 hours, 72 hours, and one week, should restrict features to each window, and should use time-ordered splits. This would show how early meaningful signals appear, which matters most in agile settings where risk must be detected within a sprint.

C. Public Jira Data Is Underused

Public Jira repositories [35] provide workflow transitions, links, and comments that GitHub and Bugzilla data often lack, yet they are rarely used for delay prediction. Their size requires careful filtering and documentation of project selection. A valuable contribution would be curated benchmark subsets with standardized features and target definitions, allowing models to be compared across studies.

D. Feature Groups Are Rarely Integrated

Studies typically use one or two feature groups. Integrated frameworks combining metadata, workflow, dependency, textual, and communication features should be explored, together with ablation studies that quantify the contribution of each group. Integration would also allow explanations to indicate the *type* of risk involved.

E. Explainable AI Is Underused for Delivery-Oriented Prediction

Delivery-risk models should explain why an item is risky, not only output a probability. Explanations should be presented as decision-support evidence, checked for stability across models and samples, and not interpreted as causal claims.

F. Transformers and LLMs Are Untested for Delivery Risk

Transformers and LLMs now perform well on issue labeling, effort estimation, and resolution-time tasks [11], [13], [14], but none of the reviewed studies uses them to predict delivery risk. Future work should test whether LLM-derived text features or zero-shot risk assessments add value over simpler representations, under the same early-window and time-aware evaluation, and should report their cost, reproducibility, and privacy implications. Because such models are less transparent, they strengthen rather than replace the case for explainability.

G. Practical Decision Support Is Missing

Performance metrics do not say what a team should do when an issue is flagged. Future studies should link explanations to candidate actions, such as clarifying dependencies, ownership, or requirements, and evaluate

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

predictions with practitioners, for example in backlog refinement, sprint planning, or delivery reviews. Models should support human decision-makers rather than make project decisions automatically.

H. Cross-Project Generalization and Reproducibility

Projects differ in issue types, workflows, priority conventions, and communication habits, so a model trained on one project may not transfer to another; a "high priority" label does not mean the same everywhere. Cross-project and within-project evaluation should be compared, and studies should document dataset selection, preprocessing, feature construction, parameters, and evaluation procedures, and share code where possible. Throughout, research should respect privacy and focus on issue- and process-level rather than personal characteristics.

XII. Limitations of This Review

This review is a structured narrative review, not a systematic literature review. Its search relied on keyword queries and citation tracking across several databases. The number of records retrieved and screened at each stage was not recorded, so the selection process cannot be reported quantitatively or reproduced exactly, and the review may have missed relevant work, particularly studies that frame delivery-related outcomes in other terms or that are published outside the indexed venues. Search engine rankings also change over time. Studies were selected on the basis of relevance to the research questions; no formal quality-assessment instrument was applied. The review is limited to English-language publications and to issue-tracking data; related sources such as version control, continuous integration, and communication platforms are discussed only where they inform issue-level prediction. Finally, because the reviewed studies use different datasets, targets, and metrics, their reported performance figures cannot be compared directly, and the synthesis is therefore qualitative.

XIII. Conclusion

This review examined machine learning and explainable AI for issue-tracking-based prediction in software engineering from the perspective of delivery risk. Issue trackers are valuable data sources because they combine structured metadata, workflow histories, text, and relations between work items, which makes it possible to anticipate as well as describe project outcomes.

Several research streams provide foundations for delivery-risk prediction. Bug-fix time prediction shows that time-based outcomes are predictable from historical data; issue lifetime prediction shows the value of dynamic and contextual features; delay prediction connects most directly to delivery and highlights the role of dependencies; studies of issue success and comments demonstrate the value of text; and process mining shows that issue histories reveal bottlenecks. Methods range from interpretable baselines to ensembles, NLP, graph, and deep learning approaches, and the most promising choice for decision support is the one that balances performance with interpretability. Evaluation should account for class imbalance and, critically, for how early risk can be detected without information leakage.

Explainable AI is central to making such predictions actionable, yet it has been applied far more often to defect prediction than to delivery-oriented problems. The main gap identified by this review is integration: few studies combine public issue-tracking data, early workflow signals, textual and dependency features, and explainability in one reproducible framework. Closing this gap could move software project monitoring from a reactive to a predictive and decision-support-oriented practice, helping teams identify risky work items earlier, understand why they are risky, and act in time.

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

REFERENCES

[1]. H. Kagdi, M. L. Collard, and J. I. Maletic, "A survey and taxonomy of approaches for mining software repositories in the context of software evolution," *J. Softw. Maint. Evol.: Res. Pract.*, vol. 19, no. 2, pp. 77–131, 2007.
[2]. W. Jung, E. Lee, and C. Wu, "A survey on mining software repositories," *IEICE Trans. Inf. Syst.*, vol. E95.D, no. 5, pp. 1384–1406, 2012.
[3]. E. Giger, M. Pinzger, and H. Gall, "Predicting the fix time of bugs," in *Proc. 2nd Int. Workshop Recommendation Syst. Softw. Eng. (RSSE)*, Cape Town, South Africa, 2010, doi: 10.1145/1808920.1808933.
[4]. P. Bhattacharya and I. Neamtiu, "Bug-fix time prediction models: Can we do better?" in *Proc. 8th Work. Conf. Mining Softw. Repositories (MSR)*, Honolulu, HI, USA, 2011, doi: 10.1145/1985441.1985472.
[5]. M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Characterization and prediction of issue-related risks in software projects," in *Proc. IEEE/ACM 12th Work. Conf. Mining Softw. Repositories (MSR)*, 2015, pp. 280–291.
[6]. M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Predicting delays in software projects using networked classification," in *Proc. 30th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, 2015, pp. 353–364.
[7]. R. Kikas, M. Dumas, and D. Pfahl, "Using dynamic and contextual features to predict issue lifetime in GitHub projects," in *Proc. 13th Int. Conf. Mining Softw. Repositories (MSR)*, Austin, TX, USA, 2016, doi: 10.1145/2901739.2901751.
[8]. S. L. Ramírez-Mora, H. Oktaba, and H. Gómez-Adorno, "Descriptions of issues and comments for predicting issue success in software projects," *J. Syst. Softw.*, vol. 168, Art. no. 110663, 2020.
[9]. M. Izadi, K. Akbari, and A. Heydarnoori, "Predicting the objective and priority of issue reports in software repositories," *Empir. Softw. Eng.*, vol. 27, no. 2, 2022, doi: 10.1007/s10664-021-10085-3.
[10]. H. Valdivia Garcia and E. Shihab, "Characterizing and predicting blocking bugs in open source projects," in *Proc. 11th Work. Conf. Mining Softw. Repositories (MSR)*, Hyderabad, India, 2014, doi: 10.1145/2597073.2597099.
[11]. Y. Noyori, H. Washizaki, Y. Fukazawa, K. Ooshima, H. Kanuka, and S. Nojiri, "Deep learning and gradient-based extraction of bug report features related to bug fixing time," *Front. Comput. Sci.*, vol. 5, Art. no. 1032440, 2023, doi: 10.3389/fcomp.2023.1032440.
[12]. M. Choetkiertikul, P. Banyongrakkul, C. Ragkhitwetsagul, S. Tuarob, H. K. Dam, and T. Sunetnanta, "Sprint2Vec: A deep characterization of sprints in iterative software development," *IEEE Trans. Softw. Eng.*, vol. 51, no. 1, pp. 220–242, 2025.
[13]. M. Fu and C. Tantithamthavorn, "GPT2SP: A transformer-based agile story point estimation approach," *IEEE Trans. Softw. Eng.*, vol. 49, no. 2, pp. 611–625, 2023, doi: 10.1109/TSE.2022.3158252.
[14]. G. Colavito, F. Lanubile, N. Novielli, and L. Quaranta, "Leveraging GPT-like LLMs to automate issue labeling," in *Proc. IEEE/ACM 21st Int. Conf. Mining Softw. Repositories (MSR)*, Lisbon, Portugal, 2024, pp. 469–480, doi: 10.1145/3643991.3644903.
[15]. S. L. Ramírez-Mora, H. Oktaba, H. Gómez-Adorno, and G. Sierra, "Exploring the communication functions of comments during bug fixing in open source software projects," *Inf. Softw. Technol.*, vol. 136, Art. no. 106584, 2021.
[16]. C. Tantithamthavorn and J. Jiarpakdee, "Explainable AI for software engineering," in *Proc. 36th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, 2021, pp. 1–2.
[17]. J. Jiarpakdee, C. K. Tantithamthavorn, and J. Grundy, "Practitioners' perceptions of the goals and visual explanations of defect prediction models," in *Proc. IEEE/ACM 18th Int. Conf. Mining Softw. Repositories (MSR)*, 2021, pp. 432–443.
[18]. A. H. Mohammadkhani, N. S. Bommi, M. Daboussi, O. Sabnis, C. Tantithamthavorn, and H. Hemmati, "A systematic literature review of explainable AI for software engineering," *arXiv preprint* arXiv:2302.06065, 2023. [Online]. Available: https://arxiv.org/abs/2302.06065
[19]. L. Montgomery, C. Lüders, and W. Maalej, "Mining issue trackers: Concepts and techniques," in *Handbook on Natural Language Processing for Requirements Engineering*, A. Ferrari and G. Ginde, Eds. Cham, Switzerland: Springer, 2025.
[20]. S. Cao et al., "A systematic literature review on explainability for ML/DL-based software engineering," *ACM Comput. Surv.*, vol. 58, no. 4, Art. no. 95, 2025, doi: 10.1145/3763230.

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

[21]. L. Arora, S. S. Girija, S. Kapoor, A. Raj, D. Pradhan, and A. Shetgaonkar, "Explainable artificial intelligence techniques for software development lifecycle: A phase-specific survey," in *Proc. IEEE 49th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, 2025, pp. 2281–2288.
[22]. B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele Univ. and Durham Univ., Tech. Rep. EBSE-2007-01, 2007.
[23]. H. Zhang, L. Gong, and S. Versteeg, "Predicting bug-fixing time: An empirical study of commercial software projects," in *Proc. 35th Int. Conf. Softw. Eng. (ICSE)*, 2013, pp. 1042–1051.
[24]. S. Akbarinasaji, B. Caglayan, and A. Bener, "Predicting bug-fixing time: A replication study using an open source software project," *J. Syst. Softw.*, vol. 136, pp. 173–186, 2018.
[25]. H. Y. Ozkan, P. E. Heegaard, W. Kellerer, and C. Mas-Machuca, "Bug analysis towards bug resolution time prediction," *arXiv preprint* arXiv:2407.21241, 2024. [Online]. Available: https://arxiv.org/abs/2407.21241
[26]. R. Kikas, M. Dumas, and D. Pfahl, "Issue dynamics in GitHub projects," in *Product-Focused Software Process Improvement (PROFES)*, Cham, Switzerland: Springer, 2015, pp. 295–310.
[27]. M. Rees-Jones, M. Martin, and T. Menzies, "Better predictors for issue lifetime," *arXiv preprint* arXiv:1702.07735, 2017. [Online]. Available: https://arxiv.org/abs/1702.07735
[28]. D.-N. Nastos, T. Diamantopoulos, D. Tosi, M. Tropeano, and A. L. Symeonidis, "Towards an interpretable analysis for estimating the resolution time of software issues," in *Proc. 29th Int. Conf. Eval. Assess. Softw. Eng. (EASE)*, Istanbul, Türkiye, 2025, doi: 10.1145/3756681.3757031.
[29]. M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Predicting the delay of issues with due dates in software projects," *Empir. Softw. Eng.*, vol. 22, no. 3, pp. 1223–1263, 2017.
[30]. M. Choetkiertikul, H. K. Dam, T. Tran, A. Ghose, and J. Grundy, "Predicting delivery capability in iterative software development," *IEEE Trans. Softw. Eng.*, vol. 44, no. 6, pp. 551–573, 2018.
[31]. B. Coremans, A. L. Klomp, S. A. Rukmono, J. Krüger, D. Fahland, and M. R. V. Chaudron, "Process mining from Jira issues at a large company," in *Proc. IEEE Int. Conf. Softw. Maint. Evol. (ICSME)*, 2023, pp. 425–435.
[32]. Y. Wang, J. Cao, and D. Lo, "Mining and predicting micro-process patterns of issue resolution for open source software projects," in *Proc. 32nd Int. Conf. Softw. Eng. Knowl. Eng. (SEKE)*, 2020, pp. 477–482, doi: 10.18293/SEKE2020-031.
[33]. Ł. Reszka, J. Sosnowski, and B. Dobrzyński, "Enhancing software project monitoring with multidimensional data repository mining," *Electronics*, vol. 12, no. 18, Art. no. 3774, 2023.
[34]. C. M. Lüders, A. Bouraffa, and W. Maalej, "Beyond duplicates: Towards understanding and predicting link types in issue tracking systems," in *Proc. 19th Int. Conf. Mining Softw. Repositories (MSR)*, Pittsburgh, PA, USA, 2022, doi: 10.1145/3524842.3528457.
[35]. L. Montgomery, C. Lüders, and W. Maalej, "An alternative issue tracking dataset of public Jira repositories," in *Proc. 19th Int. Conf. Mining Softw. Repositories (MSR)*, Pittsburgh, PA, USA, 2022, doi: 10.1145/3524842.3528486.
[36]. L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[37]. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (KDD)*, 2016, pp. 785–794.
[38]. J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 233–240.
[39]. T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, no. 3, Art. no. e0118432, 2015.
[40]. A. A. Bangash, H. Sahar, A. Hindle, and K. Ali, "On the time-based conclusion stability of cross-project defect prediction models," *Empir. Softw. Eng.*, 2020, doi: 10.1007/s10664-020-09878-9.
[41]. M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (KDD)*, 2016, pp. 1135–1144.
[42]. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 30, 2017, pp. 4765–4774.
[43]. C. Pornprasit, C. Tantithamthavorn, J. Jiarpakdee, M. Fu, and P. Thongtanunam, "PyExplainer: Explaining the predictions of just-in-time defect models," in *Proc. 36th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, 2021, pp. 407–418, doi: 10.1109/ASE51524.2021.9678763.
[44]. J. Jiarpakdee, C. Tantithamthavorn, H. K. Dam, and J. Grundy, "An empirical study of model-agnostic techniques for defect prediction models," *IEEE Trans. Softw. Eng.*, vol. 48, pp. 166–185, 2022.

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden

International Journal of Computer Science and Mobile Computing
A Monthly Journal of Computer Science and Information Technology

Fikret Zajmović *et al*, Volume 15, Issue 11, November 2026, pg. 23-43
(An Open Accessible, ISI Indexed, Fully Refereed and Peer Reviewed Journal)
ISSN: 2320-088X
Impact Factor: 7.056

[45]. C. Tantithamthavorn, J. Jiarpakdee, and J. Grundy, "Actionable analytics: Stop telling me what it is; please tell me what to do," *IEEE Softw.*, vol. 38, no. 4, pp. 115–120, 2021, doi: 10.1109/MS.2021.3072088.

[46]. J. Shin, R. Aleithan, J. Nam, J. Wang, and S. Wang, "Explainable software defect prediction: Are we there yet?" *arXiv preprint* arXiv:2111.10901, 2021. [Online]. Available: https://arxiv.org/abs/2111.10901

[47]. G. K. Rajbahadur, S. Wang, G. A. Oliva, Y. Kamei, and A. E. Hassan, "The impact of feature importance methods on the interpretation of defect classifiers," *IEEE Trans. Softw. Eng.*, vol. 48, no. 7, pp. 2245–2261, 2022, doi: 10.1109/TSE.2021.3056941.

© 2026, IJCSMC All Rights Reserved, ZAIN Publications, Fridhemsgatan 62, 112 46, Stockholm, Sweden