Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Advancing Research Software Engineering with AI: A Research Framework

Siamak Farshidi

siamak.farshidi@wur.nl

Wageningen University & Research

Kwabena Bennin

Wageningen University & Research

Önder Babur

Wageningen University & Research

June Sallou

Wageningen University & Research

Ayalew Kassahun

Wageningen University & Research

Bedir Tekinerdogan

Wageningen University & Research

Research Article

Keywords: AI for Research Software Engineering (AI4RSE), Research Software Engineering, AI for Software Engineering, Generative AI, Software Repository Mining, Research Software Maturity Assessment

Posted Date: August 7th, 2025

DOI: https://doi.org/10.21203/rs.3.rs-7178452/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Automated Software Engineering on May 4th, 2026. See the published version at https://doi.org/10.1007/s10515-026-00621-0.

Advancing Research Software Engineering with AI: A Research Framework

Siamak Farshidi[1*], Kwabena Bennin[1], Önder Babur[1], June Sallou[1], Ayalew Kassahun[1], Bedir Tekinerdogan[1]

[1*]Information Technology Group (INF), Wageningen University & Research, Wageningen, Netherlands.

*Corresponding author(s). E-mail(s): siamak.farshidi@wur.nl;
Contributing authors: kwabena.bennin@wur.nl; onder.babur@wur.nl; june.sallou@wur.nl; ayalew.kassahun@wur.nl; bedir.tekinerdogan@wur.nl;

Abstract

The rapid adoption of Artificial Intelligence (AI) and Generative AI (GenAI) tools is transforming the creation, maintenance, and dissemination of research software. Despite their growing prevalence, the implications of these technologies for Research Software Engineering (RSE) practices remain underexplored. This work introduces **AI4RSE**, an emerging research domain focused on the integration of AI into the development lifecycle of research software.

To investigate current trends in AI-augmented RSE, we conducted an empirical study of more than 1,500 open-source research software repositories hosted on Zenodo. Each repository was assessed using a quadrant-based typology defined by two key dimensions: software engineering maturity and the level of AI integration. Our analysis combined static and semantic code inspection, evaluation of alignment with the FAIR Principles for Research Software (FAIR4RS), and heuristic classification of generative AI usage and MLOps adoption.

Repositories are categorized into four development modes: *Exploratory Coding*, *Vibe Coding*, *RSE*, and *AI4RSE*, which reflect different levels of process rigor and AI tool integration. While many projects exhibit informal development patterns, a growing subset demonstrates mature, AI-assisted workflows. This landscape reveals key challenges, such as reproducibility risks and licensing ambiguity, while also highlighting emerging opportunities, including AI-assisted testing and intelligent documentation generation.

The findings support a research agenda for AI4RSE, outlining benchmarks, guidelines, and community standards to promote responsible, reproducible, and scalable adoption of AI in scientific software development.

Keywords: AI for Research Software Engineering (AI4RSE), Research Software Engineering, AI for Software Engineering, Generative AI, Software Repository Mining, Research Software Maturity Assessment

1 Introduction

Scientific research increasingly relies on complex software to collect data, conduct experiments, and analyze results [1, 2]. As the scale and complexity of scientific challenges grow, so does the burden on researchers to develop, maintain, and validate the software that underpins their work. At the same time, advances in artificial intelligence, particularly GenAI tools such as GitHub Copilot [3] and ChatGPT [4], are reshaping the way software is created and maintained. These tools offer the potential for faster development, more accessible programming for non-experts, and new forms of automation such as code generation, refactoring, and documentation synthesis [5, 6].

This technological transition presents both a significant opportunity and a substantial challenge for research software engineering. Traditional RSE focuses on principles such as maintainability, reproducibility, and sustainability, supported by practices like version control, testing, and structured documentation [7, 8]. However, the role of AI tools within these established workflows is still poorly understood. Several essential questions remain unresolved. In what ways are AI and GenAI tools currently being applied in research software development? Do these tools enhance software quality, or do they introduce new types of risk? What are the potential consequences for provenance, licensing, and reproducibility [9]? A central challenge is determining how this technological transition can be studied rigorously and directed through evidence-based practices.

To explore these questions, we define a new research domain called Artificial Intelligence for Research Software Engineering (AI4RSE). This domain examines how AI technologies influence the processes, practices, and outcomes of scientific software development. To investigate this domain empirically, we analyze a curated dataset of 1,512 research software repositories hosted on Zenodo and categorized using the IEEE Taxonomy 2025 [10]. This dataset enables a systematic assessment of current trends in AI integration and software engineering maturity.

Each repository is evaluated along two core dimensions: the level of software engineering maturity and the degree of AI integration. These dimensions form the basis of a quadrant model that captures four distinct development patterns observed in the dataset. To support this classification, we construct a multi-dimensional scoring framework that combines static and semantic code analysis, heuristic detection of GenAI-generated code, FAIR4RS compliance, and the presence of MLOps indicators.

Our contributions in this work are as follows:

- We formally define **AI4RSE** as a research domain at the intersection of RSE, AI/GenAI technologies, and open science practices.

- We curate and analyze a dataset of 1,512 Zenodo-hosted research software repositories, linking each to GitHub metadata to evaluate development practices, automation patterns, and AI involvement.
- We introduce a *Software Development Quadrant Model* that classifies projects along two axes: engineering maturity and AI integration. This model reveals four common development modes: *Exploratory Coding*, *Vibe Coding*, *RSE*, and *AI4RSE*.
- We design a *multi-dimensional scoring framework* that combines FAIR compliance, software engineering heuristics, generative AI inference, and MLOps activity to quantify project characteristics.
- We propose a research agenda organized around five thematic pillars: AI-augmented practices, FAIR-AI reproducibility, GenAI-driven collaboration, sustainable AI workflows, and community training and engagement.

By grounding AI4RSE in a broad empirical analysis of real-world projects, we highlight both the promise and the limitations of AI-assisted research software. Our findings indicate that although GenAI tools are starting to transform development workflows, their impact remains uneven and is often limited by weak engineering practices. The results underscore the need for systematic processes, robust standards, and new tools that support reproducibility and sustainability. In this context, AI4RSE provides a timely and practical framework for understanding and guiding the role of AI in the next generation of scientific software development.

This study is structured as follows. *Section 2* reviews the background and related work on RSE, AI, FAIR principles, and MLOps, and introduces the Software Development Quadrant Model. *Section 3* details our methodology for collecting, analyzing, and classifying over 1,500 research software repositories, including composite scoring for engineering maturity and AI integration. *Section 4* presents empirical findings on repository practices, quadrant distributions, FAIR compliance, AI tool usage, and metadata quality. *Section 5* reflects on the implications of AI4RSE for software tooling, automation, reproducibility, and responsible AI adoption. Furthermore, this section formalizes AI4RSE as a distinct interdisciplinary research field and proposes a roadmap for future inquiry. *Section 6* summarizes our findings and calls for collaboration across the SE, AI, and RSE communities.

2 Background and Related Work

2.1 Background

Scientific advancement increasingly relies on complex software [11]: from data analysis pipelines [12] to simulation frameworks [13], research software drives discoveries across disciplines. Unlike production-grade software, research software is often developed under tight timelines by domain experts lacking formal training in software engineering. This leads to challenges related to sustainability, maintainability, and

reproducibility [7]. To address these issues, the RSE community promotes best practices including version control, automated testing, structured documentation, and open licensing [7, 8].

In parallel, advances in AI and GenAI are reshaping the software development landscape [14, 15]. Tools such as GitHub Copilot and large language models can assist with code generation, automated refactoring, and documentation synthesis [5, 6], thereby accelerating development and lowering entry barriers for non-experts. However, these benefits come with risks: AI-generated code may introduce undocumented behavior, propagate licensing ambiguities, or compromise testability and transparency [16–18].

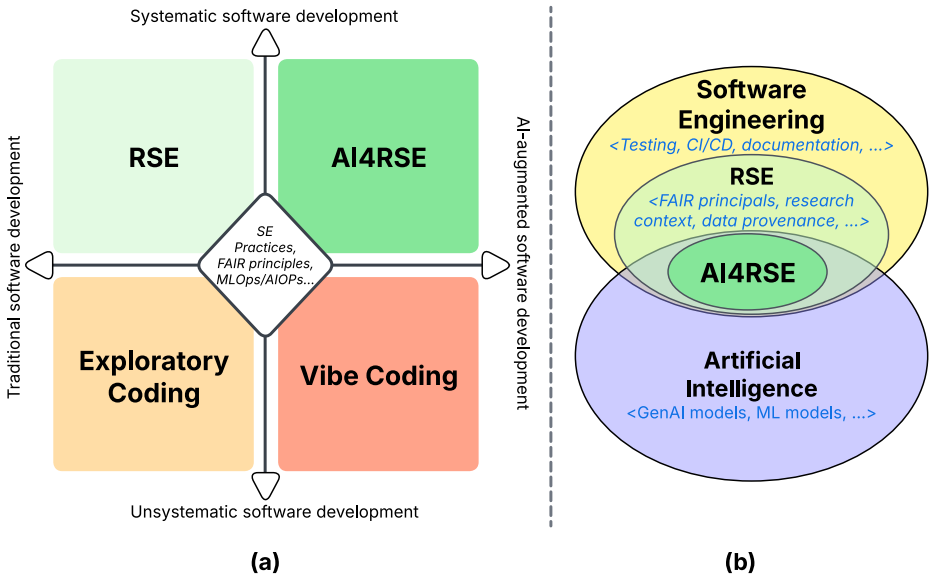


Fig. 1 (a) The Software Development Quadrant Model situates four research software development modes along two axes: engineering maturity (unsystematic to systematic) and AI integration (traditional to AI-augmented). (b) A conceptual Venn diagram illustrating how AI4RSE emerges at the intersection of Software Engineering, Artificial Intelligence, and Research Software Engineering.

To conceptualize the convergence of these trends, we introduce the *Software Development Quadrant Model* (Figure 1). This framework maps development practices along two key dimensions: *engineering maturity*, ranging from unsystematic to systematic, and *AI augmentation*, spanning from traditional to AI-driven development. These axes yield four quadrant-defined development modes:

- **Exploratory Coding:** Informal, ad hoc development, typically without structured documentation, testing, or versioning [19–21].
- **Vibe Coding:** Informal development enhanced by GenAI assistance. Developers in this mode frequently rely on AI-generated code, but lack engineering discipline, documentation, or reproducibility safeguards [22, 23].

- **RSE:** Systematic engineering practices applied in scientific contexts, including testing, modular design, collaboration, and version control [24–27].
- **AI4RSE:** An integrated, aspirational mode that combines rigorous software engineering practices with GenAI-driven automation, alongside adherence to open science and FAIR principles [28–32].

Figure 1(a) positions these modes along the axes of AI integration and engineering maturity. Exploratory and Vibe Coding reflect early-stage or unstructured development, whereas RSE and AI4RSE represent mature practices. AI4RSE extends traditional RSE by incorporating GenAI-based capabilities such as intelligent testing, auto-documentation, and MLOps-based automation [33, 34].

AIOps (Artificial Intelligence for IT Operations) refers to the application of AI techniques to automate and enhance traditional operations tasks such as anomaly detection, log analysis, self-healing systems, and infrastructure monitoring [35]. MLOps (Machine Learning Operations), in contrast, focuses on the operationalization of machine learning pipelines, including components like CI/CD for models, automated retraining, model deployment, experiment tracking, and data versioning [36]. While both paradigms involve automation and AI, AIOps emphasizes system resilience and observability, whereas MLOps centers on the reproducibility, scalability, and governance of ML model workflows.

Figure 1(b) complements this by illustrating the conceptual boundaries between related domains. RSE can be seen as a specialization within Software Engineering (SE), focused on scientific codebases and research contexts [24, 37]. AI4RSE, in turn, emerges at the intersection of SE, RSE, and AI. It combines the methodological rigor of SE (e.g., CI/CD, testing), the reproducibility and openness championed by RSE and the FAIR principles, and the assistive capabilities of AI/GenAI [30, 31, 37].

Developed initially for data stewardship and now extended to research software, the FAIR principles emphasize that software should be *Findable*, with clear identifiers and metadata; *Accessible*, through open or well-defined access protocols; *Interoperable*, using standardized formats and interfaces; and *Reusable*, supported by proper licensing, provenance, and documentation [38]. These principles support long-term sustainability and reproducibility by ensuring that scientific software can be discovered, understood, and reused across projects and disciplines.

The vertical axis of the quadrant model is grounded in maturity frameworks such as Capability Maturity Model Integration [39] and the Research Software Maturity Model [40], which evaluate structure, sustainability, and disciplined process execution. The horizontal axis captures the increasing adoption of AI tools in software development, from low-level automation to fully generative workflows [37]. Similar quadrant-based taxonomies have been adopted in industry to classify technological readiness and process integration, including Gartner's Magic Quadrants [41, 42] and MLOps maturity models [43–45].

Despite the potential of this convergence, current practices often treat these domains in isolation. While FAIR principles [30, 31] have been extended to software to promote openness and interoperability, and MLOps practices support reproducible model lifecycle management [46], few initiatives fully integrate them with GenAI and RSE processes.

Industry adoption of GenAI into robust CI/CD environments is progressing rapidly [33], but academic and research institutions frequently lack the engineering resources required to implement such pipelines [7]. As a result, AI tools are often deployed opportunistically, emphasizing short-term productivity at the expense of long-term sustainability and traceability.

We therefore propose *AI4RSE* as an integrated domain that synthesizes practices from SE, RSE, and AI-enabled automation. Through empirical analysis of over 1,500 Zenodo-hosted research software repositories, we investigate how AI influences development behaviors and identify both enablers and barriers to achieving sustainable, high-quality, AI-augmented research software ecosystems.

To clarify the scope of this work, it is helpful to distinguish AI4RSE from related areas. *AI4SE* refers to the use of artificial intelligence to assist general-purpose software engineering tasks, typically in commercial or industrial settings. *SE4AI* involves applying software engineering methods to the development and deployment of AI systems, such as model pipelines and testing infrastructure [47]. In research settings, *RSE4AI* focuses on using research software engineering practices to support machine learning experimentation, dataset curation, and model reproducibility. By contrast, our study centers on *AI4RSE*, which involves integrating AI and GenAI tools into the development, evaluation, and maintenance of research software itself. As illustrated in Figure 1(b), AI4RSE is conceptualized as a focused subdomain within research software engineering that intersects with both software engineering and artificial intelligence.

2.2 Related Work

To position our study within the broader landscape of RSE and the integration of AI and GenAI into scientific software development, we conducted a semi-structured literature review using a snowballing approach. Beginning with foundational works in RSE and AI in SE, we followed backward and forward citation trails and assembled a corpus of 39 publications spanning the years 2018 to 2025. Studies were included if they explicitly addressed at least one of the following: RSE practices, software automation, FAIR principles, MLOps workflows, or the use of AI/GenAI tools in software development or maintenance.

Each study was coded using a seven-dimensional schema that functions as a conceptual metamodel for comparing methodological and thematic scope. The coding assessed whether the study had a *conceptual or empirical focus*, the *methodology* used (such as case studies, interviews, surveys, repository mining, or theoretical synthesis), and the *scale or dataset* evaluated, including the number of projects, participants, or empirical units. It further analyzed each paper's engagement with *AI or GenAI involvement*, alignment with the *FAIR principles*, attention to *MLOps workflows* (e.g., deployment, CI/CD, monitoring), and coverage of *software engineering practices* such as modularity, testing, and automation.

As shown in Figure 2, empirical studies dominate the literature (18 out of 39), primarily using surveys, interviews, and field studies to explore development challenges and tooling in scientific environments [28, 32, 48]. Another 16 papers are conceptual, providing taxonomies, architectural models, or vision statements without collecting

Study	Year	Focus Area	Dataset / Scale	Empirical Studies	Conceptual and Commentary	Research Methods	AI/GenAI in SE	FAIR Principles Analysis	AI/MLOps Focus	SE Practice Evaluation
This study	2025	AI4RSE	1500+ Zenodo repositories	X		Repository analysis	X	X	X	X
Cao et al.	2025	Explainable AI in SE	108 studies	X		Literature review	X			
Ariamajd et al.	2025	RSE Automation Practices	N/A	X		Proof of concept		X	X	X
Hyrynsalmi et	2025	Interdisciplinary Publishing Challenges	73 survey	X		Survey				
Eisty et al.	2025	RSE Testing Practices	131 survey	X		Survey				X
Tenquist et al.	2025	RSE Inclusivity Initiatives	8 interviews	X		Expert interviews				
Felderer et al.	2025	RSE Development Improvement	N/A		X	Conceptual viewpoint	X	X		X
Goth et al.	2025	RSE Core Competencies	N/A		X	Workshop study		X		X
Tistelgrén	2024	AI Tool Utilization	37 survey	X		Survey	X		X	X
Milewicz et al.	2024	Evidence-Based Practices	6 reviews	X		Expert interviews				
Odeh	2024	AI Code Generation	N/A	X		Literature review	X			
Hasselbring et	2024	Software Categorization	5 examples		X	Conceptual framework		X		
Russo	2024	GenAI in RSE	280 participants	x		Survey	X			
Ali et al.	2023	Trustworthy AI Explainability	410 articles	X		Literature review	X			
Farshidi et al.	2023	FAIR Software Assessment	5 projects	X		Proof of concept		X		X
Malviya-Thakur	2023	Lab RSE Practices	1 case study	X		Case study research				X
Heroux	2023	Research Software Science	N/A		X	Conceptual framework				
Huerta et al.	2023	FAIR AI Adoption	N/A		X	Workshop study	X	X	X	
Felderer et al.	2023	RSE Research Improvement	N/A		X	Conceptual framework		X		X
Heroux et al.	2023	Scientific Software Science	124 papers, 264 participants		X	Workshop study	X			X
Leng	2023	RSE Profession Emergence	N/A		X	Conceptual viewpoint				
Felderer et al.	2023	RSE Improvement Methods	N/A		X	Conceptual viewpoint		X		X
Joran et al.	2023	Sustainable RSE	N/A		X	Conceptual viewpoint		X		X
Lamprecht et al.	2022	RSE Research Questions	27 workshop, 15 survey	X		Survey				
Serebrenik et	2022	AI Engineering Trends	N/A		X	Editorial overview				
Sochat et al.	2022	Software Framework	N/A	X		Conceptual framework				X
Hong et al.	2022	FAIR Principles for Research Software	250 participants	X		Survey		X		
Barker et al.	2022	FAIR Principles for Research Software	500 participants	X		Survey		X		
Trisovic et al.	2022	Research code quality	2109 Github repositories	X		Repository analysis				X
Carver et al.	2022	Sustainable RSE	1149 participants	X		Survey			X	X
Xu et al.	2021	AI Scientific Applications	8 disciplines	X		Survey	X			
Lu et al.	2021	Responsible AI Engineering	21 interviews	X		Expert interviews	X		X	X
Cohen et al.	2021	RSE Core Practices	N/A		X	Conceptual framework				X
Roscher et al.	2020	ML Explainability Science	N/A	X		Literature review	X			
Hasselbring et	2020	Open-Source Sustainability	5000+ Github repositories	X		Repository analysis				
Lamprecht et al.	2020	FAIR Principles for Research Software	N/A	X		Literature review		X		
Amershi et al.	2019	ML Integration Processes	14 interviews, 551 survey	X		Case study research	X		X	X
de Souza et al.	2019	Software Sustainability Factors	19 RSEs	X		Expert interviews				X
Gomez-Diaz et	2019	CDUR Evaluation Protocol	N/A		X	Conceptual framework				
Allamanis et al.	2018	ML Code Naturalness	N/A	X		Literature review	X			

Fig. 2 Thematic distribution of the 39 studies in our semi-structured literature analysis.

new data [25, 49, 50]. These conceptual contributions often highlight fragmentation in tooling, challenges in reproducibility, and the need for engineering best practices in research settings.

Only two studies systematically assess FAIR principles in research software [30, 31], despite growing emphasis on FAIR4RS compliance as a quality metric. Two others

apply large-scale repository mining methods [1, 51], but they focus primarily on metadata completeness and archival sustainability, not engineering maturity or automation pipelines. A single paper directly examines GenAI-assisted development in research contexts [5], although several mention AI assistance in broader software engineering domains without targeting RSE scenarios.

Among the 39 reviewed works, fewer than five engage meaningfully with MLOps topics, and only three discuss both FAIR and automation in the same study. In terms of scale, most empirical studies operate on modest datasets (fewer than 100 repositories or participant responses), limiting generalizability. Additionally, few studies adopt multidimensional coding schemes or compare software artifacts using structured criteria such as modularity, CI/CD, or experiment tracking. Intersections among AI usage, open science principles, and operational tooling remain underexplored.

This thematic distribution reveals a substantial gap. Prior work tends to isolate one or two dimensions, focusing, for instance, on testing and CI/CD [28], FAIR compliance [30], or high-level GenAI trends [5], without attempting to synthesize across dimensions. To date, no research comprehensively integrates AI and GenAI adoption, FAIRness evaluation, MLOps maturity, and software engineering practices within a single empirical framework.

Our study addresses this gap by analyzing 1,512 Zenodo-hosted research software repositories using a unified, scalable model. We examine repositories across all seven dimensions of our metamodel, combining large-scale repository mining with operational tagging and AI usage profiling. In doing so, we establish **AI4RSE** as a novel, data-driven research domain that unites previously siloed threads of RSE, SE, AI, FAIR, and MLOps under a cohesive empirical framework.

3 Methodology

To operationalize the AI4RSE quadrant and explore how AI tools, FAIR principles, and software engineering maturity co-evolve in research settings, we designed a multi-stage, data-driven analysis pipeline. This section outlines the process of curating, processing, and analyzing over 1,500 open research software repositories developed during the GenAI era.

Figure 3 offers a high-level summary. The pipeline progresses through four major phases: (1) repository collection, (2) linkage to development artifacts, (3) feature extraction, and (4) analytical categorization.

3.1 Phase 1: Repository Collection

We began with a query on Zenodo, targeting all records labeled as `Software` and created after November 2022, marking the emergence following ChatGPT. Filtering by metadata quality and eliminating duplicates yielded **1,512** distinct repositories. This served as our seed dataset, grounded in the IEEE Taxonomy 2025 classification of scientific domains [10].

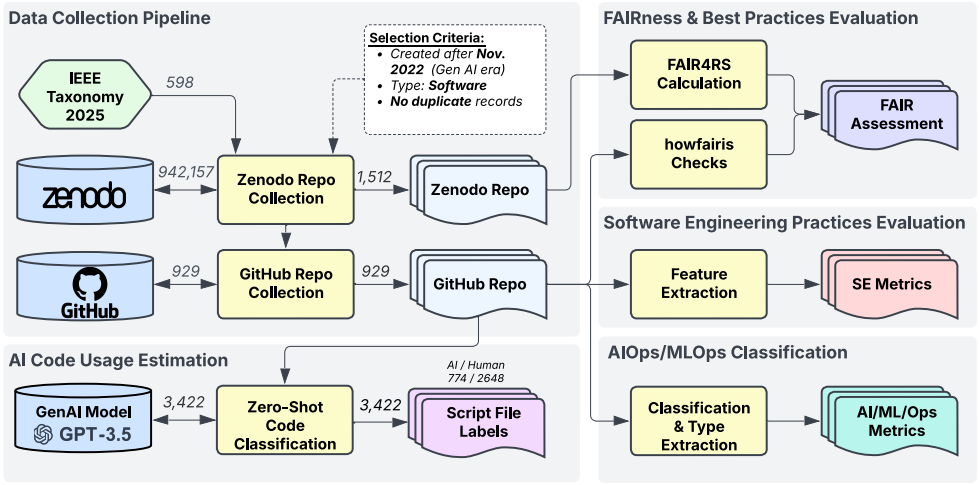


Fig. 3 Pipeline for constructing and analyzing the AI4RSE dataset. For complete implementation and all technical details, visit the RSEMM portal at https://ai-rse.nl/RSEMM/

3.2 Phase 2: GitHub Linking

To analyze development practices, we mapped Zenodo entries to corresponding GitHub repositories using structured metadata fields (e.g., `relatedIdentifier`, `url`). This step enabled access to the codebase, commit history, and CI configurations. We successfully resolved and cloned *929* repositories, forming our core analysis set.

3.3 Phase 3: Feature Extraction and Analysis

We developed automated heuristics[1] to extract metrics from each repository across four analytical dimensions. These components are described below, with each output discretized into ordinal scores (High = 3, Medium = 2, Low = 1, None = 0) for downstream synthesis.

3.3.1 AI Code Usage Estimation

To estimate the extent to which GenAI tools were used in code development, we employed a Zero-Shot Classification approach using a large language model (specifically GPT-3.5 Turbo in this study) following the method proposed by Yang et al. [52] and Ye et al. [53]. For each repository, up to five recently modified script files were sampled and individually submitted to a prompt asking whether the code was likely written by an AI system (such as ChatGPT or Codex) or by a human developer.

The model's responses were aggregated to compute an AI usage ratio, representing the proportion of sampled files labeled as AI-generated. Based on this ratio, we assigned each repository to one of three usage levels: High if more than two-thirds of

[1]For complete implementation and all technical details, visit the RSEMM portal at https://ai-rse.nl/RSEMM/

the sampled files were flagged as AI-generated ($r > 0.66$), Medium if between one-third and two-thirds ($0.33 < r \leq 0.66$), and Low if one-third or fewer ($r \leq 0.33$).

This heuristic offers a lightweight and scalable approach to infer GenAI integration in scientific codebases, facilitating comparative analysis of development patterns across the dataset.

3.3.2 FAIRness and Best Practices

We assessed each repository's alignment with the FAIR4RS principles [38], which emphasize that research software should be Findable, Accessible, Interoperable, and Reusable. This evaluation was based on fifteen specific indicators, such as the presence of persistent identifiers, clarity of license information, metadata richness, and reference formatting. For each indicator, we recorded whether the criterion was fully met, partially satisfied, or not met, assigning numeric scores of 1.0, 0.5, or 0.0, respectively.

These individual scores were then grouped by FAIR dimension and averaged to generate a composite score for each principle. Based on these averages, we labeled the dimension as High if the score was at least 0.75, Medium if it ranged between 0.50 and 0.75, and Low otherwise. This provided a structured way to quantify the FAIRness of each project.

In addition to FAIR metrics, we also evaluated five widely recommended best practices: whether the repository included a license file, a citation file, a versioning scheme, and badges indicating registry or quality compliance. These binary checks helped capture fundamental openness and maintainability signals that complement the FAIR assessment. The resulting FAIRness scores feed directly into the broader RSE evaluation framework.

3.3.3 Software Engineering Maturity

To evaluate how systematically each project applied software engineering practices, we analyzed seven key factors adapted from the framework proposed by Munaiah et al. [54]. These factors were: contributor frequency (Community), presence of configuration files for automated workflows (CI), the ratio of comments to source code (Documentation), commit activity over time (History), the volume of issue-related events (Issues), the presence of a license file (License), and the proportion of test code relative to total codebase size (Testing).

Each of these dimensions was extracted using static analysis and GitHub metadata. For example, we assessed the number of contributors who were active throughout the project's lifetime, whether CI configuration files existed, and the level of documentation and testing in the codebase. We also examined development history, issue engagement, and licensing status.

Repositories were then categorized as High, Medium, or Low in each dimension based on empirically established thresholds. These maturity levels provide a structured and comparable view of engineering quality across projects, serving as a foundational input to our overall RSE scoring model.

3.3.4 AIOps and MLOps Classification

To assess the operational sophistication of research software projects in terms of their AI engineering workflows, we conducted a classification based on the presence of AIOps- and MLOps-related practices in project metadata, documentation, and commit messages. This approach builds on established analyses of AIOps project characteristics [55] and case studies of real-world MLOps pipelines [56].

Our classification procedure identifies the presence of these practices based on curated keyword sets. We analyze each repository's name, description, README, and up to 250 recent commit messages by transforming them into a normalized text corpus. Two binary flags are set to indicate whether any keywords from the AIOps or MLOps vocabularies are detected. A repository is labeled as *AIOps* if it contains only AIOps-related terms, *MLOps* if only MLOps terms are present, *Both* if it reflects features from both domains, and *None* if neither is detected.

For repositories classified as AIOps or MLOps, we also extract more granular workflow patterns using fine-grained keyword sets. These include operational features such as "experiment tracking," "model serving," "log parsing," "monitoring," or "automated recovery." In cases where both sets are present (Ambiguous), we invoke a lightweight large language model prompt to resolve classification and subtype extraction.

This process enables us to quantify the degree to which research software incorporates what we refer to as *modern AI engineering practices*, practices that support the automated, scalable, and maintainable development and operation of AI-enhanced systems. These include continuous integration of models, real-time observability, infrastructure resilience, and the reproducibility of ML pipelines. The resulting classification directly contributes to each repository's AI scoring profile, reflecting the depth and operational maturity of its AI-related workflow infrastructure.

3.4 Phase 4: Composite Scoring and Typology

To synthesize insights across the four analytic dimensions, we computed two composite indices that capture key aspects of research software quality and AI integration. The `RSE_Score` reflects engineering rigor and openness, calculated as the average of the SE maturity score and the FAIRness score, both derived from the assessments described in previous sections. The `AI_Score`, by contrast, captures the presence and depth of GenAI involvement, computed as the average of the AI Code Usage score and the AI/ML operational classification score.

$$\text{RSE_Score} = \frac{\text{SE_Score} + \text{FAIR_Score}}{2}, \quad \text{AI_Score} = \frac{\text{AI_Code_Score} + \text{AI_Ops_Score}}{2}$$

Each score ranges from 0 to 3, corresponding to ordinal labels of Low (1), Medium (2), or High (3), as assigned during the feature extraction process. Using fixed thresholds ($t = 2.0$ for RSE_Score and $u = 0.0$ for AI_Score), we mapped repositories into one of four developmental modes:

- **Exploratory Coding:** (Low RSE, Low AI) typically unsystematic projects with minimal engineering structure or AI use.
- **Vibe Coding:** (Low RSE, High AI) projects with informal workflows but significant GenAI involvement.
- **RSE:** (High RSE, Low AI) traditionally engineered research software without strong GenAI signals.
- **AI4RSE:** (High RSE, High AI) projects that combine systematic engineering practices with AI-augmented development.

This quadrant-based typology enables both quantitative comparisons and qualitative interpretation of how research software evolves in response to emerging tools and community practices.

3.4.1 Sensitivity Analysis of RSE_Score Threshold

To evaluate the robustness of our quadrant classification framework, we conducted a sensitivity analysis by varying the threshold used to define the RSE_Score. This score, representing the average of software engineering maturity and FAIRness (each rated as High = 3, Medium = 2, Low = 1, None = 0), is central to distinguishing between the Exploratory, RSE, AI4RSE, and Vibe Coding quadrants. By incrementally adjusting the RSE_Score threshold from 0.0 to 3.0, we observed how repository assignments shifted across developmental categories.

Table 1 Sensitivity of Quadrant Classification Across RSE_Score Thresholds

RSE Score	Vibe Coding	RSE	AI4RSE	Exploratory Coding
0.00	0	1074	438	0
0.50	0	1074	438	0
0.75	0	729	438	345
1.00	0	491	438	583
1.25	2	469	436	605
1.50	7	451	431	623
1.75	65	357	373	717
2.00	**168**	**234**	**270**	**840**
2.25	289	121	149	953
2.50	350	68	88	1006
2.75	389	18	49	1056
3.00	415	1	23	1073

At lower thresholds (≤ 1.5), the majority of repositories fall into the RSE and AI4RSE quadrants, indicating projects with at least a “Low” or “Medium” score in software engineering maturity and FAIRness. In this range, the Exploratory and Vibe Coding categories remain negligible, indicating few repositories score “None” in both dimensions.

As the threshold increases beyond 1.75, approaching the boundary between "Medium" and "High", a marked redistribution occurs. Repositories previously classified as RSE or AI4RSE are increasingly reclassified as Exploratory or Vibe Coding. This trend highlights the presence of many mid-maturity projects that do not consistently reach "High" scores across both assessment dimensions.
Statistical Summary - A Spearman correlation analysis reveals a strong inverse relationship between Exploratory Coding and RSE classifications ($\rho = -0.98$), and a moderate negative correlation between AI4RSE and Exploratory Coding ($\rho = -0.72$). A Kruskal-Wallis H-test confirms that changes in the RSE_Score threshold significantly impact the distribution of quadrant classifications ($H = 8437.05$, $p < 0.001$).
Threshold Justification - We selected a threshold of 2.0 based on the underlying scoring logic: a score of 2 corresponds to "Medium" ratings in both FAIRness and software engineering maturity. This represents a natural boundary between developmental modes that are still forming (Exploratory, Vibe Coding) and those showing intentional engineering practices (RSE, AI4RSE). The threshold of 2.0 thus captures projects that are maturing but not yet fully optimized.
Sensitivity Considerations - The observed statistical significance and abrupt category shifts emphasize that quadrant boundaries are not absolute. Rather, they function as interpretive heuristics to help characterize project maturity. Researchers are therefore encouraged to test adjacent thresholds (e.g., 1.75 or 2.25) to assess the robustness of findings and avoid overinterpretation of rigid cutoffs.

4 Results and Analysis

In this section, we report the empirical findings from our analysis of 1,512 research software repositories. We structure the results along the four evaluation axes outlined in Section 3: software engineering maturity, FAIRness, AI code usage, and AIOps/MLOps adoption. The repositories were categorized into four development modes based on these scores: *AI4RSE*, *RSE*, *Vibe Coding*, and *Exploratory Coding*, as visualized in Figure 4.

4.1 Fuzzy Aggregation of Metric Distributions

To enable interpretable comparisons across heterogeneous metrics, such as contributor activity, CI/CD usage, documentation quality, or AI tool presence, we adopt a fuzzy aggregation approach [57], inspired by prior work on architectural knowledge scoring [58].

Each analytical dimension is scored based on ordinal classification into three categories: Low (L), Medium (M), and High (H), derived from empirical thresholds over raw feature values. The observed distribution across these categories for a given dimension is expressed as a tuple $\langle L, M, H \rangle$, where each element represents the number of repositories falling into that class.

To generate a normalized fuzzy score, we apply a weighted average over the tuple using fixed category weights:

		AI4RSE	RSE	Vibe Coding	Exploratory Coding
Software Engineering Efforts	*Community*	<7,88,175> 0.81	<0,82,152> 0.82	<56,75,37> 0.44	<78,136,43> 0.43
	CI/CD	<184,0,86> 0.32	<144,0,90> 0.38	<167,0,1> 0.01	<254,0,3> 0.01
	Documentation	<33,0,237> 0.88	<47,0,187> 0.80	<111,2,55> 0.33	<211,0,46> 0.18
	History	<22,31,217> 0.86	<19,21,194> 0.87	<76,14,78> 0.51	<104,22,131> 0.55
	Issues	<142,0,128> 0.47	<108,0,126> 0.54	<158,0,10> 0.06	<243,0,14> 0.05
	License	<19,0,251> 0.93	<1,0,233> 1.00	<109,0,59> 0.35	<163,0,94> 0.37
	Testing	<157,1,112> 0.42	<149,2,83> 0.36	<164,0,4> 0.02	<249,0,8> 0.03
AI	*Code Gen Usage*	<110,123,27> 0.34	<0,0,0> 0.00	<70,73,25> 0.37	<0,0,0> 0.00
	AIOPs/MLOPs	<0,0,21> 1.00	<0,0,0> 0.00	<0,0,0> 0.00	<0,0,0> 0.00
FAIR Assessment	*Findable*	<0,1,269> 1.00	<0,1,233> 1.00	<0,2,166> 0.99	<0,31,809> 0.98
	Accessible	<1,269,0> 0.50	<2,232,0> 0.50	<2,166,0> 0.49	<44,796,0> 0.47
	Interoperable	<14,254,2> 0.48	<15,219,0> 0.47	<15,153,0> 0.46	<674,166,0> 0.10
	Reusable	<63,95,112> 0.59	<58,108,68> 0.52	<103,49,16> 0.24	<504,321,15> 0.21
	How FAIR IS	<32,109,129> 0.68	<14,93,127> 0.74	<71,62,35> 0.39	<714,91,35> 0.10
Zenodo Metadata	*Title Length*	-0.21	-0.27	-0.12	0.16
	Description Length	0.02	-0.02	-0.14	0.03
	Notes Length	-0.09	-0.16	-0.23	0.12
	Affiliation Count	0.09	-0.07	-0.26	0.04
	Keyword Count	0.07	-0.06	-0.4	0.07
	Download Count	-0.08	0.05	-0.13	0.04
	Revision Count	0.07	0.17	-0.02	-0.07
	Grant Presence	0.34	0.39	0.55	0.11
	Community Presence	0.13	0.15	0.2	0.04
GitHub Metadata	*Repo Name Length*	0.51	0.51	0.61	-0.43
	Repo Description Length	0.35	0.48	0.25	-0.3
	Fork Count	0.03	0.27	-0.04	-0.08
	Repo Size	0.23	0.15	0.17	-0.15
	Star Count	0.05	0.24	-0.06	-0.07
	Network Count	0.03	0.28	-0.04	-0.08
	Open Issues	0.1	0.22	-0.07	-0.08
	Branch Count	0.3	0.2	-0.04	-0.14
	Contributor Count	0.13	0.41	-0.02	-0.15
	Update Count	0.14	0.33	-0.07	-0.12
	Commit Message Count	0.14	0.27	-0.05	-0.11
	Author Count	0.13	0.38	0	-0.15
	Committer Count	0.26	0.52	0.06	-0.24
	GitHub Repo Presence	1	1	1	0.31
	Issue Tracker Presence	0.91	0.95	0.91	0.24
	Project Presence	0.91	0.87	0.91	0.25
	Wiki Presence	0.76	0.69	0.77	0.21
	License Presence	0.84	0.92	0.48	0.12
	README Presence	0.96	0.97	0.88	0.24
	#Repos	270	234	168	840

Fig. 4 Landscape analysis across the AI4RSE quadrant model. Cell values represent fuzzy tuple distributions (<Low, Medium, High>) with associated normalized scores.

$$\text{Fuzzy Score} = \frac{w_L \cdot L + w_M \cdot M + w_H \cdot H}{L + M + H}$$

where $w_L = 0.0$, $w_M = 0.5$, and $w_H = 1.0$. These weights reflect an ordinal intensity scale in which only High values contribute fully, Medium values contribute partially, and Low values do not contribute to the final score. The resulting score

ranges from 0 to 1. This approach preserves transparency by showing raw category counts while also enabling ranking and visual comparison using a single composite value per metric.

Color Encoding: For presentation in Figure 4, fuzzy scores are color-coded on a green-to-red gradient: green (score $\geq$ 0.75), yellow (0.5 $\leq$ score $<$ 0.75), orange (0.25 $\leq$ score $<$ 0.5), and red (score $<$ 0.25). This visual encoding highlights maturity differences across development quadrants.

4.2 Quadrant Distribution and Overall Patterns

Applying a threshold of 2.0 on the composite RSE_Score (see Section 3.4.1), the dataset yields the following distribution:

- **AI4RSE:** 270 repositories (17.9%); high AI usage and high software engineering maturity.
- **RSE:** 234 repositories (15.5%); high engineering rigor, but low AI integration.
- **Vibe Coding:** 168 repositories (11.1%); heavy AI use, but lacking systematic practices.
- **Exploratory Coding:** 840 repositories (55.5%); minimal engineering and AI integration.

While most repositories remain in the exploratory category, the AI4RSE group is emerging as a high-performing quadrant, combining systematic engineering with modern AI workflows. RSE projects demonstrate engineering maturity but do not incorporate AI practices. In contrast, Vibe Coding demonstrates AI usage without engineering support, and Exploratory Coding reflects informal or early-stage development.

Software Engineering Practices - AI4RSE and RSE repositories consistently achieve higher scores across core engineering dimensions. Licensing scores are notably high, 0.93 in AI4RSE and 1.00 in RSE, whereas Vibe and Exploratory Coding fall below 0.40. CI/CD usage is virtually absent in unsystematic quadrants (0.01) but moderately present in AI4RSE (0.14) and RSE (0.20). Documentation and contributor history scores follow a similar pattern, underscoring the concentration of rigorous practices in structured quadrants.

FAIRness Evaluation - RSE achieves the highest FAIRness score (0.74), followed closely by AI4RSE (0.68). Vibe Coding (0.39) and Exploratory Coding (0.10) perform poorly, particularly in terms of reusability and metadata quality. Although AI4RSE does not surpass RSE in FAIR scores, it demonstrates a more balanced profile across both FAIRness and AI integration. This indicates a more substantial alignment with open science practices as new tooling paradigms are adopted.

AI Integration and MLOps Adoption - AI code generation features are most prominent in AI4RSE (0.34) and Vibe Coding (0.37), but only AI4RSE demonstrates AIOps/MLOps usage (scoring 1.00). None of the RSE or Exploratory repositories show such integration. This highlights a key distinction: while Vibe Coding relies heavily on GenAI tools for code generation, AI4RSE reflects deeper operational integration of AI-enabled workflows, including automation and deployment pipelines.

4.3 Zenodo and GitHub Metadata Analysis

To complement the quadrant-based analysis, we further examined project-level metadata derived from Zenodo and GitHub. These metadata features serve as indirect yet informative signals of software maturity, visibility, and engagement. All correlation values reported in this section are computed over normalized feature values: continuous numeric attributes (e.g., description length, contributor count) were standardized using z-score normalization, while binary or categorical indicators (e.g., grant presence, license availability) were evaluated using their proportion across repositories within each quadrant.

Zenodo Metadata Insights.

We analyzed fields such as title and description length, number of notes, affiliations, keyword richness, grant acknowledgments, and revision/download activity. Repositories within the *AI4RSE* and *RSE* quadrants consistently displayed stronger metadata profiles.

Grant presence, computed as the ratio of repositories with grant-related metadata, was moderately correlated with RSE (0.39) and AI4RSE (0.34), suggesting that funded projects are more likely to follow systematic development practices. Similarly, *community-linked metadata*, including institutional affiliations and multiple contributors, was more common in structured quadrants. Although *descriptive richness*, captured via fields like title and note length, showed wide variability across disciplines, its correlation with quadrant type was generally weak or inconsistent, likely due to divergent metadata conventions across research domains.

GitHub Metadata Signals.

On GitHub, we assessed development indicators, including repository size, contributor and commit activity, issue tracking behavior, and auxiliary files such as CI configurations and wikis. These indicators differentiated the quadrants.

Repositories classified under AI4RSE and RSE tended to have longer and more descriptive repository names and README files. These attributes, when z-score normalized, yielded correlation values ranging from 0.35 to 0.51. The *number of contributors* was positively correlated with RSE (0.41) and to a lesser extent with AI4RSE (0.13), reinforcing the idea that mature development is often collaborative.

Project health indicators, including update frequency, presence of meaningful commit messages, and use of issue tracking systems, were notably higher in AI4RSE and RSE repositories. Conversely, Vibe Coding and Exploratory repositories generally lacked these signs of sustained maintenance. Binary presence of README and license files followed similar patterns: AI4RSE scored 0.96 and 0.84, respectively, while Exploratory projects fell to 0.48 for documentation and just 0.12 for licenses.

Quadrant Differentiation.

Taken together, these metadata trends reinforce the quadrant-based interpretation. *AI4RSE* repositories exhibit stronger signals across both Zenodo and GitHub, reflecting higher levels of curation, compliance, and community engagement. In contrast,

Exploratory and *Vibe Coding* repositories generally suffer from weaker metadata, smaller teams, and less visible activity. These trends align well with the previously observed technical metrics, suggesting that metadata quality can serve as a useful indicator of research software development maturity.

Such insights can be leveraged in practice to build automated triage or recommendation mechanisms, for instance, in funding platforms, repository indexing systems, or institutional software registries. They offer scalable ways to guide attention toward high-quality, sustainable software projects.

Spearman correlation analysis reveals a strong negative association between Exploratory Coding and RSE classifications ($\rho = -0.98$), as well as a moderately strong negative correlation between AI4RSE and Exploratory Coding ($\rho = -0.72$). These trends support the existence of a maturity gradient across development modes, with AI4RSE and RSE positioned at the structured end and Exploratory Coding at the informal, ad hoc extreme.

While AI4RSE and RSE share common ground in software engineering rigor, they diverge sharply in their use of AI tools and automation, indicating complementary yet distinct development cultures. The separation of Vibe Coding from both mature quadrants further underscores the disconnect between GenAI adoption and systematic engineering practices.

Summary of Observations: AI4RSE repositories combine robust engineering and FAIR practices with AI-based tools and automation, positioning them as an emerging model of modern, high-quality research software. RSE remains the stronghold of traditional software best practices but lacks AI integration. Vibe Coding demonstrates enthusiasm for GenAI but is hindered by a lack of process maturity. Exploratory projects dominate numerically but lag significantly behind in all quality dimensions.
Overall, the results validate the AI4RSE quadrant as a measurable and distinct mode of development. It captures the intersection of sustainable software engineering, reproducibility, and AI-augmented workflows, offering a pathway for maturing research software ecosystems.

5 Discussion

This section reflects on the quadrant distributions presented in Figure 4, using them to interpret current trends in research software engineering and the role of AI in that landscape. We connect these findings to earlier work in RSE and AI4SE [25, 47], highlighting how our results provide an empirical foundation for the emerging concept of *AI4RSE*.

5.1 Key Findings in Context

Our quadrant analysis reveals a clear divide in the current research software ecosystem. More than half of the repositories in our dataset (55.5%, n=840) fall into the *Exploratory Coding* quadrant. These projects often lack documentation, licensing,

and automated testing patterns that have been well documented in prior studies of scientific code and scripting practices [7].

In contrast, only about one-third of the repositories (≈33%, n=504) meet the criteria for the more structured *RSE* or *AI4RSE* quadrants. This demonstrates that disciplined engineering practices and alignment with FAIR principles can coexist with, and potentially improve, AI-supported workflows.

Three notable patterns emerge from the quadrant analysis:

GenAI without engineering support leads to fragile outputs. Repositories in the *Vibe Coding* quadrant (11.1%, n=168) show the highest occurrence of AI-generated code but the lowest scores on CI/CD pipelines and metadata completeness. This aligns with previous findings about code fragility in industrial use of AI assistants [59, 60], suggesting similar risks in scientific software.

FAIR alignment is linked to sustained project activity. Projects that include licenses, CITATION files, and rich metadata also tend to have longer development histories and greater community engagement. This supports the idea of a positive feedback loop between openness and sustainability, as proposed by Barker et al. [30].

AI4RSE practices are emerging but remain limited to specific domains. Only 270 repositories (17.9%) meet our threshold for both high AI usage and high engineering maturity. These projects often include features such as model cards, automated notebooks, and prompt tracking. However, they are concentrated in a few domains, such as bioinformatics and climate science, rather than being broadly distributed across disciplines.

5.2 AI4RSE as a Research Domain: A Research Agenda

This study provides empirical evidence that AI4RSE is emerging as a distinct mode of practice, combining GenAI tooling, engineering maturity, and open science principles. The quadrant analysis (Section 4) reveals that only a small but meaningful subset of repositories (Figure 1) achieves this balance. These projects demonstrate that AI-augmented, reproducible research software is not only feasible but already present in real-world development, although unevenly.

Building on this empirical foundation, we propose a research agenda to support AI4RSE as a formal interdisciplinary domain. The agenda responds to specific gaps identified in our analysis and outlines six interrelated challenges:

- **Sustainable GenAI Integration:** Incorporating generative AI tools into scientific workflows without compromising reproducibility, accountability, or scientific validity.
- **Reproducibility Infrastructure:** Creating tools and standards for tracking, versioning, and validating AI-generated content, particularly in stochastic or interactive environments.
- **Adaptive Automation Frameworks:** Tailoring MLOps and CI/CD practices for research settings characterized by limited engineering support, diverse computing platforms, and short-lived project teams.

- **Human-AI Collaboration Models:** Designing team structures, review workflows, and contribution tracking methods that enable responsible integration of GenAI into collaborative software development.
- **FAIR+AI Compliance:** Extending FAIR4RS principles to account for AI-generated artifacts, including prompts, models, and partially synthesized code.
- **Governance and Ethics:** Developing shared norms, licensing strategies, and risk mitigation practices for the responsible use of GenAI in scientific software.

These priorities reflect both the opportunities and limitations identified in our quadrant analysis (Figure 1). For example, repositories with high levels of AI-generated content often lack essential engineering practices, such as testing and metadata management. In contrast, repositories that demonstrate high maturity tend to avoid GenAI tools entirely. Bridging this gap will require a combination of technical solutions, including reproducibility auditing tools and prompt tracking systems, along with institutional support for shared standards, incentive structures, and community-developed best practices.

We argue that AI4RSE deserves focused research attention as an emerging subfield situated at the intersection of research software engineering and artificial intelligence. This relationship is illustrated in the Venn diagram in Figure 1(b), where AI4RSE appears as a subset of RSE that incorporates elements of both AI and traditional software engineering. By presenting a structured research agenda grounded in large-scale empirical analysis, this study provides a foundation for advancing AI4RSE in a way that promotes transparency, sustainability, and scientific rigor.

5.3 Limitations and Threats to Validity

Before turning to specific categories of validity threats, we briefly discuss broader limitations that shape the context and scope of our study. Our analysis is based on a large-scale dataset of 1,512 research software repositories, drawn from Zenodo and GitHub. While this offers a structured and FAIR-aligned foundation, it introduces certain sampling and methodological boundaries that are important to acknowledge.

First, the dataset reflects a platform-specific bias. Zenodo tends to overrepresent European and life science projects, while repositories hosted on domain-specific forges such as CERN's GitLab or institutional archives may be underrepresented. As a result, our findings may not fully capture disciplinary diversity or the practices of all research communities.

Second, although our GenAI classifier was designed to combine code-level features with repository metadata, it is subject to limitations inherent to zero-shot inference with large language models. A manual spot check revealed that approximately 7% of short utility scripts were misclassified, often due to ambiguous content or limited context. This highlights the challenge of reliably detecting AI-generated code in real-world repositories. Future work could benefit from complementary detection methods, such as compiler watermarking [53, 61].

Third, GenAI tools and practices continue to evolve rapidly. The quadrant distributions we report represent a snapshot in time, and rerunning the pipeline in

future months may yield different patterns due to shifts in tool adoption, repository structures, or community norms.

Despite these limitations, we applied several measures to strengthen the robustness of our findings. Sensitivity analysis (Table 1) was used to test the effects of score thresholds. We triangulated more than 40 distinct indicators, combining static repository attributes with dynamic trace evidence. All intermediate scores and classification outputs are openly released, enabling reproducibility and independent verification[2].

We also assess the validity of our approach across four standard categories:

Construct validity relates to how well our metrics reflect the intended constructs of software engineering maturity, AI integration, and reproducibility. To mitigate this, we adopted validated frameworks and structured pipelines. Engineering maturity was assessed using seven dimensions, including CI/CD presence, licensing, test coverage, documentation, and community activity, following Munaiah et al. [54] and operationalized through our engineering assessment pipeline [63]. Reproducibility and openness were evaluated using a formal FAIRness assessment based on the FAIR4RS principles [38], as detailed in our FAIRness scoring protocol [64]. This included checks for metadata richness, licensing, provenance, standard file formats, and accessibility via persistent identifiers. For AI integration, we applied a multi-step classification approach that combines code-level prompt detection and repository metadata analysis [65]. Our scoring approach intentionally combines static indicators and dynamic traces to reduce bias from any single source. When signals were absent, we used conservative defaults (e.g., "low" scores) to minimize false assumptions about quality or automation.

Internal validity addresses the reliability of our classification procedures, especially the use of large language models to infer AI-generated content. Our zero-shot classification method, adapted from Yang et al. [52] and Ye et al. [53], uses file-level inference on recently changed scripts, as described in [65]. To reduce false positives, ambiguous responses defaulted to "Human." We further validated model outputs manually for a stratified sample of repositories to confirm alignment with declared tool usage and repository contents. These steps helped ensure that AI-ratio estimates are consistent and reliable.

External validity reflects the generalizability of our findings. Our dataset comprises public research software repositories archived on Zenodo and hosted on GitHub, which may overrepresent well-maintained or evident projects. However, this scope is aligned with the FAIR4RS infrastructure and provides structured metadata that supports automated evaluation. Our automation classification (e.g., MLOps, AIOps) was based on established keyword heuristics and ambiguity resolution prompts [66], increasing coverage of diverse workflows.

Conclusion validity concerns whether our scoring thresholds and quadrant boundaries produce sound and reproducible results. All thresholds were derived from empirical score distributions and tested using sensitivity analysis (Section 3.4.1). The quadrant model employs fuzzy scoring to mitigate boundary effects, thereby

[2]The full dataset, including classification outputs, FAIRness ratings, and quadrant labels, is available on Mendeley Data [62], allowing others to replicate or extend our analysis.

reducing artifacts associated with complex classification. The consistency of quadrant distributions across scoring variations supports the robustness of our conclusions.

Although some limitations remain due to dataset scope and tool-based inference, we mitigated their effects through method triangulation, manual validation, and publicly available scoring data. These steps collectively support the validity of our findings and provide a foundation for future research in AI4RSE.

6 Conclusion

This study has introduced and empirically grounded the emerging domain of *AI4RSE*, Artificial Intelligence for Research Software Engineering. Using a dataset of 1,512 Zenodo-hosted research-software repositories, we built a quadrant model that positions projects along two principal axes: software engineering maturity and AI integration.

The analysis shows a maturing but uneven landscape. More than half of the repositories reside in the *Exploratory Coding* quadrant; yet, a growing fraction (approximately 18%) already demonstrates both strong engineering practices and deep AI integration, the hallmarks of *AI4RSE*. These projects confirm that sustainable, reproducible, AI-augmented research software is achievable in the real world.

The data also highlight significant risks. Repositories that rely heavily on GenAI tools while neglecting testing, licensing, or provenance expose the scientific record to hidden defects and legal uncertainty. Addressing these tensions will require coordinated, evidence-based action from tool builders, funders, and research institutions.

AI4RSE is more than a passing trend. It presents an opportunity to rethink how research software is developed, shared, and sustained. By treating AI4RSE as a first-class research domain and pursuing the lines of work outlined above, the community can ensure that future research software is not only faster to develop but also more reproducible, transparent, and impactful.

Future work: Our snapshot captures a single fifteen-month window and is limited to Zenodo and GitHub. Follow-up research should track the same projects over time, compare them with codebases on domain-specific forges, and investigate how quadrant membership shifts as tooling and community norms evolve. Additionally, the field would benefit from open benchmarks that quantify the impact of AI-augmented workflows on code quality, FAIR compliance, and energy use. Equally important are controlled trials that embed prompt logging, license checking, or reproducibility bots into real projects to measure improvements and identify adoption barriers.

References

[1] Trisovic, A., Lau, M.K., Pasquier, T., Crosas, M.: A large-scale study on research code quality and execution. Scientific Data **9**(1), 60 (2022)

[2] Vescovi, R., Chard, R., Saint, N.D., Blaiszik, B., Pruyne, J., Bicer, T., Lavens, A., Liu, Z., Papka, M.E., Narayanan, S., et al.: Linking scientific instruments and computation: Patterns, technologies, and experiences. Patterns **3**(10) (2022)

[3] Ziegler, A., Kalliamvakou, E., Li, X.A., Rice, A., Rifkin, D., Simister, S., Sittampalam, G., Aftandilian, E.: Measuring github copilot's impact on productivity. Communications of the ACM **67**(3), 54–63 (2024)

[4] Wang, Y., Pan, Y., Yan, M., Su, Z., Luan, T.H.: A survey on chatgpt: Ai–generated contents, challenges, and solutions. IEEE Open Journal of the Computer Society **4**, 280–302 (2023)

[5] Cao, S., Sun, X., Widyasari, R., Lo, D., Wu, X., Bo, L., Zhang, J., Li, B., Liu, W., Wu, D., et al.: A systematic literature review on explainability for machine/deep learning-based software engineering research. arXiv preprint arXiv:2401.14617 (2024)

[6] Russo, D.: Navigating the complexity of generative ai adoption in software engineering. ACM Transactions on Software Engineering and Methodology **33**(5), 1–50 (2024)

[7] Carver, J.C., Weber, N., Ram, K., Gesing, S., Katz, D.S.: A survey of the state of the practice for research software in the united states. Peerj computer science **8**, 963 (2022)

[8] Lamprecht, A.-L., Martinez-Ortiz, C., Barker, M., Bartholomew, S.L., Barton, J., Chue Hong, N., Cohen, J., Druskat, S., Forest, J., Grad, J.-N., et al.: What do we (not) know about research software engineering? Journal of Open Research Software **10**(1) (2022)

[9] Alenezi, M., Akour, M.: Ai-driven innovations in software engineering: A review of current practices and future directions. Applied Sciences **15**(3), 1344 (2025)

[10] The Institute of Electrical and Electronics Engineers (IEEE): IEEE Taxonomy for 2025. Version 2025 – IEEE Taxonomy; available from IEEE Standards Association (2025)

[11] Jensen, E.A., Katz, D.S.: Awareness of fair and fair4rs among international research software funders. Scientific Data **12**(1), 627 (2025)

[12] Zhao, Z., Koulouzis, S., Bianchi, R., Farshidi, S., Shi, Z., Xin, R., Wang, Y., Li, N., Shi, Y., Timmermans, J., *et al.*: Notebook-as-a-vre (naavre): From private notebooks to a collaborative cloud virtual research environment. Software: Practice and Experience **52**(9), 1947–1966 (2022)

[13] Gill, M., Singh, D.: A comprehensive study of simulation frameworks and research directions in fog computing. Computer Science Review **40**, 100391 (2021)

[14] Wong, M.-F., Guo, S., Hang, C.-N., Ho, S.-W., Tan, C.-W.: Natural language generation and understanding of big code for ai-assisted programming: A review. Entropy **25**(6), 888 (2023)

[15] Jiang, J., Wang, F., Shen, J., Kim, S., Kim, S.: A survey on large language models for code generation. arXiv preprint arXiv:2406.00515 (2024)

[16] Khojah, R., Mohamad, M., Leitner, P., Oliveira Neto, F.G.: Beyond code generation: An observational study of chatgpt usage in software engineering practice. Proceedings of the ACM on Software Engineering **1**(FSE), 1819–1840 (2024)

[17] Xu, F.F., Alon, U., Neubig, G., Hellendoorn, V.J.: A systematic evaluation of large language models of code. In: Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming, pp. 1–10 (2022)

[18] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.D.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al.: Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374 (2021)

[19] Kery, M.B., Myers, B.A.: Exploring exploratory programming. In: 2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 25–29 (2017). IEEE

[20] Kery, M.B., Horvath, A., Myers, B.A.: Variolite: Supporting exploratory programming by data scientists. In: CHI, vol. 10, pp. 3025453–3025626 (2017)

[21] Betz, S.: What is patch management? how software patching can protect you, your customers, and your organization. Built In (2022)

[22] Ray, P.P.: A review on vibe coding: Fundamentals, state-of-the-art, challenges and future directions. Authorea Preprints (2025)

[23] Taulli, T.: AI-Assisted Programming: Better Planning, Coding, Testing, and Deployment. " O'Reilly Media, Inc.", ??? (2024)

[24] Cohen, J., Katz, D.S., Barker, M., Hong, N.C., Haines, R., Jay, C.: The four pillars of research software engineering. IEEE Software **38**(1), 97–105 (2020)

[25] Felderer, M., Goedicke, M., Grunske, L., Hasselbring, W., Lamprecht, A.-L., Rumpe, B.: Investigating research software engineering: Toward rse research. Communications of the ACM (2025)

[26] Milewicz, R., Bisila, J., Mundt, M., Teves, J.: Seeking enlightenment: Incorporating evidence-based practice techniques in a research software engineering team. arXiv preprint arXiv:2403.16827 (2024)

[27] Felderer, M., Goedicke, M., Grunske, L., Hasselbring, W., Lamprecht, A.-L., Rumpe, B.: Toward research software engineering research. Zenodo (2023)

[28] Eisty, N.U., Kanewala, U., Carver, J.C.: Testing research software: an in-depth survey of practices, methods, and tools. Empirical Software Engineering **30**(3), 81 (2025)

[29] Malviya-Thakur, A., Bernholdt, D.E., Godoy, W.F., Watson, G.R., Doucet, M., Coletti, M.A., Rogers, D.M., McDonnell, M., Billings, J.J., Maccabe, B.: Research software engineering at oak ridge national laboratory. Computing in Science & Engineering **24**(5), 14–23 (2023)

[30] Barker, M., Chue Hong, N.P., Katz, D.S., Lamprecht, A.-L., Martinez-Ortiz, C., Psomopoulos, F., Harrow, J., Castro, L.J., Gruenpeter, M., Martinez, P.A., *et al.*: Introducing the fair principles for research software. Scientific Data **9**(1), 622 (2022)

[31] Farshidi, S., Maassen, J., Bakhshi, R., Van Nieuwpoort, R., Jansen, S., *et al.*: Fairseco: An extensible framework for impact measurement of research software. In: 2023 IEEE 19th International Conference on e-Science (e-Science), pp. 1–10 (2023). IEEE

[32] Hyrynsalmi, S.M., Liebel, G., Souza Santos, R., Baltes, S.: Not real or too soft? on the challenges of publishing interdisciplinary software engineering research. In: 2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS), pp. 106–117 (2025). IEEE

[33] Xu, Y., Liu, X., Cao, X., Huang, C., Liu, E., Qian, S., Liu, X., Wu, Y., Dong, F., Qiu, C.-W., et al.: Artificial intelligence: A powerful paradigm for scientific research. The Innovation **2**(4) (2021)

[34] Heroux, M.A.: Research software science: Expanding the impact of research software engineering. Computing in Science & Engineering **24**(6), 22–27 (2023)

[35] Notaro, P., Cardoso, J., Gerndt, M.: A survey of aiops methods for failure management. ACM Transactions on Intelligent Systems and Technology (TIST) **12**(6), 1–45 (2021)

[36] Kreuzberger, D., Kühl, N., Hirschl, S.: Machine learning operations (mlops): Overview, definition, and architecture. IEEE access **11**, 31866–31879 (2023)

[37] Barenkamp, M., Rebstadt, J., Thomas, O.: Applications of ai in classical software engineering. AI Perspectives **2**(1), 1 (2020)

[38] Chue Hong, N.P., Katz, D.S., Barker, M., Lamprecht, A.-L., Martinez, C., Psomopoulos, F.E., Harrow, J., Castro, L.J., Gruenpeter, M., Martinez, P.A., et al.: Fair principles for research software (fair4rs principles). Zenodo (2022)

[39] Chrissis, M.B., Konrad, M., Shrum, S.: CMMI for Development: Guidelines for Process Integration and Product Improvement. Pearson Education, ??? (2011)

[40] Bakhshi, R., Maassen, J., Ortiz, C.M., Nieuwpoort, R., Jansen, S., et al.: Rsmm: A framework to assess maturity of research software project. arXiv preprint arXiv:2406.01788 (2024)

[41] Batchu, A., Walsh, P., Brasier, M., Khandabattu, H.: Magic quadrant for ai code assistants. Magic quadrant, Gartner, Inc. (August 2024). Gartner does not endorse any vendor, product or service.

[42] Zimmermann, A., al.: Emerging Tech Impact Radar: Generative AI. Research report, Gartner, Inc. (December 2023). Available via Gartner reprint ID 1-2KKSPDO5

[43] Zarour, M., Alzabut, H., Alsarayrah, K.: Mlops best practices, challenges and maturity models: A systematic literature review. Information and Software Technology, 107733 (2025)

[44] Woźniak, A.P., Milczarek, M., Woźniak, J.: Mlops components, tools, process and metrics-a systematic literature review. IEEE Access (2025)

[45] John, M.M., Olsson, H.H., Bosch, J.: An empirical guide to mlops adoption: Framework, maturity model and taxonomy. Information and Software Technology **183**, 107725 (2025)

[46] Lu, Q., Zhu, L., Xu, X., Whittle, J., Douglas, D., Sanderson, C.: Software engineering for responsible ai: An empirical study and operationalised patterns. In: Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice, pp. 241–242 (2022)

[47] McDermott, T., DeLaurentis, D., Beling, P., Blackburn, M., Bone, M.: Ai4se and se4ai: A research roadmap. Insight **23**(1), 8–14 (2020)

[48] Tenquist, M., Azman, A., Meaden, R., Onikan, A., Jay, C., Banerji, A.: Recommendations for developing effective inclusivity initiatives in research software engineering. Computing in Science & Engineering (2025)

[49] Ariamajd, A., Castro, R.L.-R., Volkamer, A.: Pypackit: Automated research software engineering for scientific python applications on github. arXiv preprint arXiv:2503.04921 (2025)

[50] Hasselbring, W., Druskat, S., Bernoth, J., Betker, P., Felderer, M., Ferenz, S., Hermann, B., Lamprecht, A.-L., Linxweiler, J., Prat, A., et al.: Multi-dimensional categorization of research software with examples (2024)

[51] Hasselbring, W., Carr, L., Hettrick, S., Packer, H., Tiropanis, T.: From fair research data toward fair and open research software. it-Information Technology **62**(1), 39–47 (2020)

[52] Yang, X., Zhang, K., Chen, H., Petzold, L., Wang, W.Y., Cheng, W.: Zero-shot detection of machine-generated codes. arXiv preprint arXiv:2310.05103 (2023)

[53] Ye, T., Du, Y., Ma, T., Wu, L., Zhang, X., Ji, S., Wang, W.: Uncovering llm-generated code: A zero-shot synthetic code detector via code rewriting. In: AAAI Conference on Artificial Intelligence, vol. 39, pp. 968–976 (2025)

[54] Munaiah, N., Kroh, S., Cabrey, C., Nagappan, M.: Curating github for engineered software projects. Empirical Software Engineering **22**, 3219–3253 (2017)

[55] Aghili, R., Li, H., Khomh, F.: Studying the characteristics of aiops projects on github. Empirical Software Engineering **28**(6) (2023)

[56] Zhou, Y., Yu, Y., Ding, B.: Towards mlops: A case study of ml pipeline platform. In: International Conference on Artificial Intelligence and Computer Engineering (ICAICE), pp. 494–500 (2020). IEEE

[57] Chen, S.-M.: Aggregating fuzzy opinions in the group decision-making environment. Cybernetics & Systems **29**(4), 363–376 (1998)

[58] Farshidi, S., Jansen, S., Werf, J.M.: Capturing software architecture knowledge for pattern-driven design. Journal of Systems and Software **169**, 110714 (2020)

[59] Sergeyuk, A., Golubev, Y., Bryksin, T., Ahmed, I.: Using ai-based coding assistants in practice: State of affairs, perceptions, and ways forward. Information and Software Technology **178**, 107610 (2025)

[60] Perry, N., Srivastava, M., Kumar, D., Boneh, D.: Do users write more insecure code with ai assistants? In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, pp. 2785–2799 (2023)

[61] Lee, T., Hong, S., Ahn, J., Hong, I., Lee, H., Yun, S., Shin, J., Kim, G.: Who wrote this code? watermarking for code generation. arXiv preprint arXiv:2305.15060 (2023)

[62] Farshidi, S.: AI4RSE. Mendeley Data (2025). https://doi.org/10.17632/t2dygzcsyt.1 . https://doi.org/10.17632/t2dygzcsyt.1

[63] AI4RSE Lab, INF Chair Group: Software Engineering Practices Evaluation (Internal Documentation). https://ai-rse.nl/RSEMM/. Unpublished internal technical documentation (2025)

[64] AI4RSE Lab, INF Chair Group: FAIRness Assessment Process (Internal Documentation). https://ai-rse.nl/RSEMM/. Unpublished internal technical documentation (2025)

[65] AI4RSE Lab, INF Chair Group: AI vs. Human Code Classification (Internal Documentation). https://ai-rse.nl/RSEMM/. Unpublished internal technical documentation (2025)

[66] AI4RSE Lab, INF Chair Group: AI/ML/Ops Classification Process (Internal Documentation). https://ai-rse.nl/RSEMM/. Unpublished internal technical documentation (2025)

[67] Goth, F., Thiele, J.P.: Foundational competencies and specializations of a research software engineer. Computing in Science & Engineering (2025)

[68] Tistelgrén, S.: Artificial intelligence in software development: Exploring utilisation, tools, and value creation (2024)

[69] Odeh, A.: Exploring ai innovations in automated software source code generation: Progress, hurdles, and future paths. Informatica **48**(8) (2024)

[70] Ali, S., Abuhmed, T., El-Sappagh, S., Muhammad, K., Alonso-Moral, J.M., Confalonieri, R., Guidotti, R., Del Ser, J., Díaz-Rodríguez, N., Herrera, F.: Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. Information fusion **99**, 101805 (2023)

[71] Huerta, E., Blaiszik, B., Brinson, L.C., Bouchard, K.E., Diaz, D., Doglioni, C., Duarte, J.M., Emani, M., Foster, I., Fox, G., *et al.*: Fair for ai: An interdisciplinary and international community building perspective. Scientific data **10**(1), 487 (2023)

[72] Heroux, M.A., Bernholdt, D.E., McInnes, L.C., Cary, J.R., Katz, D.S., Raybourn, E.M., Rouson, D.: Basic research needs in the science of scientific software development and use: Investment in software is investment in science. Technical report, US Department of Energy (USDOE), Washington, DC (United States). Office of ... (2023)

[73] Leng, J.: The development of research software engineering as a profession. Open Access Government, 278–279 (2023)

[74] Deschamps, J., Dalle Nogare, D., Jug, F.: Better research software tools to elevate the rate of scientific discovery or why we need to invest in research software engineering. Frontiers in Bioinformatics **3**, 1255159 (2023)

[75] Serebrenik, A., Staron, M., Cabot, J., Penzenstadler, B., Hochstein, L., Carver, J.C.: Ai engineering research in software engineering venues. IEEE Software **39**(6), 105–108 (2022)

[76] Sochat, V., May, N., Cosden, I., Martinez-Ortiz, C., Bartholomew, S.: The research software encyclopedia: a community framework to define research software. Journal of Open Research Software (2022)

[77] Roscher, R., Bohn, B., Duarte, M.F., Garcke, J.: Explainable machine learning for scientific insights and discoveries. Ieee Access **8**, 42200–42216 (2020)

[78] Lamprecht, A.-L., Garcia, L., Kuzak, M., Martinez, C., Arcila, R., Martin Del Pico, E., Dominguez Del Angel, V., Van De Sandt, S., Ison, J., Martinez, P.A., *et al.*: Towards fair principles for research software. Data Science **3**(1), 37–59 (2020)

[79] Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., Zimmermann, T.: Software engineering for machine learning: A case study. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pp. 291–300 (2019). IEEE

[80] Souza, M.R., Haines, R., Vigo, M., Jay, C.: What makes research software sustainable? an interview study with research software engineers. In: 2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), pp. 135–138 (2019). IEEE

[81] Gomez-Diaz, T., Recio, T.: On the evaluation of research software: the cdur procedure. F1000Research **8**, 1353 (2019)

[82] Allamanis, M., Barr, E.T., Devanbu, P., Sutton, C.: A survey of machine learning for big code and naturalness. ACM Computing Surveys (CSUR) **51**(4), 1–37 (2018)