A DOMAIN-LEVEL EMPIRICAL CHARACTERIZATION OF AI-AUTHORED PULL REQUESTS IN OPEN-SOURCE SOFTWARE ENGINEERING

12/2/25

Henry Jie, Xiangrui Ke, Zhiheng Lyu

David R. Cheriton School of Computer Science

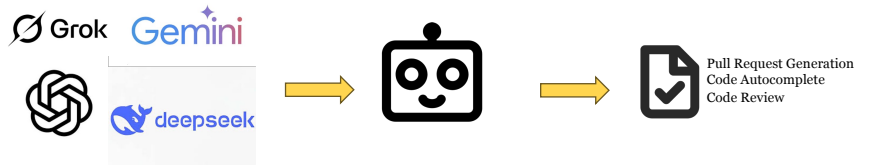
UNIVERSITY OF WATERLOO

BACKGROUND

AI models are shaping today's software industry

AI code agent is hugely involved in software development

Agentic-PRs is an important part of open-source development.



Powerful LLMs AI-agent Human-AI Interation

PROBLEM

Software domains are not homogeneous: they differ in architecture, testing requirements, and review norms.

Existing studies treat all PRs as homogeneous, overlooking domain-specific factors that may influence their reception.

Lack of systematic understanding across domains

Motivation

AI–human Collaboration ??

To Identify where AI coding agents are most effective.

Developer can gain feedback for domain-specific fine-tuning and improvement (e.g accept rate)

An advancing responsible AI adoption in software engineering

Large-scale, domain-aware empirical understanding of Agentic-PR behavior

Domain Definition and Taxonomy

ID	Category	Description
1	system	System related: io, file-system, memory, exception, underflow, overflow, serialization, path, symlink, encoding, datetime, unicode, container (runtime context), os-compat, networking
2	language	Programming language related: type, encoding, unicode, regex, parsing, linting
3	architecture	Design related: schema, design, refactor, model, OOP
4	data-processing	Data related: SQL, NoSQL, parquet, Spark, distributed data, batch, graph, algorithm
5	compute-ml	Machine-learning related: torch, transformer, DNN, RNN
6	compute-gpu	GPU related: OpenGL, CUDA, XLA, Triton, hw-accel, quantization
7	infra	DevOps and configuration: env, container, build, logging, config, packaging, cloud, docker, registry, k8s-controller, custom-resource, CI, CD, release-eng, rollback, pipeline, proxy
8	security	Security and vulnerability related: auth, vulnerability fix
9	observability	Observability related: metrics, tracing, telemetry, log
10	performance	Performance related: bug-fix, caching, compression, session
11	frontend	UI related: screen, DOM, view
12	document	Documentation related: docstring, clarification, explanation, annotation
13	testing-quality	Testing related: test coverage, unit test, integration test
14	api-design	API and backend related: API endpoint, REST, Swagger, OpenAPI
15	others	Not suitable for any of the above categories

Research Questions

RQ.1	RQ.2	RQ.3
How **accurately** can LLMs identify the software domains of Agentic-PRs?	Which domains exhibit higher Agentic-PR **acceptance** rates? What kind of characteristics most strongly influence their likelihood of being merged?	What is the distribution of **review times** of Agentic-PRs in each domain, and which factors would cause the long-tail review delays?

Related Work & Novel Contribution

AI for Software Engineering (AI4SE)

Code Review:

Automating Code Review Activities by Large-Scale Pre-training (Li, et al, 2022)

DevOps:

Explaining GitHub Actions Failures with Large Language Models: Challenges, Insights, and Limitations (Valenzuela-Toledo, et al, 2025)

Some works focus on Agentic-PRs

Quality gatekeepers: investigating the effects of code review bots on pull request activities (Wessel, et al, 2022)

It shows that automation can increase merge rates and alter collaboration patterns in open-source projects

On the Use of Agentic Coding: An Empirical Study of Pull Requests on GitHub (Watanebe, 2025)

It studies the usefulness of Agentic-PRs and the extent to which their contributions are accepted in real-world projects

Related Work & Novel Contribution

The closest related work:

On the Use of Agentic Coding: An Empirical Study of Pull Requests on GitHub (Watanebe, 2025)

How do Agentic-PRs differ from Human PRs in terms of change size and purpose?

What proportion of Agentic-PRs are accepted without revisions? If needed, to what extent?

What changes are required to revise Agentic-PRs?

We study form an orthogonal angle: the acceptance/rejection rate among different theme categories the codes belong to

Table 1. PR purposes based on analysis of PR content

Category	Description	% APRs	% HPRs	% Δ
fix	Code changes that fix bugs and faults within the codebase	31.0%	30.8%	0.2% ↑
feat	Code changes that introduce new features to the codebase, encompassing both internal and user-oriented features	26.8%	27.6%	0.8% ↓
refactor	Code restructuring without changing its behavior, aiming to improve maintainability	24.9%	14.9%	10% ↑
docs	Updates to documentation or comments, such as README edits, typo fixes, or API docs improvements	22.1%	14.0%	8.1% ↑
test	Additions or modifications to test files, including new test cases or updates to existing tests	18.8%	4.5%	14.3% ↑
build	Changes to build configurations (e.g., Maven, Gradle, Cargo). Change examples include updating dependencies, configuring build configurations, and adding scripts	10.8%	3.6%	7.2% ↑
style	Non-functional code changes that improve readability or consistency. This type encompasses aspects like variable naming, indentation, and addressing linting or code analysis warnings	7.5%	1.8%	5.7% ↑
ci	Changes to CI/CD workflows and configurations, e.g., ".travis.yml" and ".github/workflows"	6.1%	7.2%	1.1% ↓

Tools

- **AIDev**: the **first large-scale** (932,791 items), openly available **dataset** capturing Agentic-PRs from real-world GitHub repositories
- Utilize **GPT-5** to labelling **2,000** Agentic PRs
  - LLM is capable of handling complex classification tasks
  - No need to train a new model
  - Fallback plan is to use a traditional ML algorithm for classification task

Tools

Table 3: Prompt of Domain Classification

You are a classification model that assigns technical tags to pull-request descriptions. Given the PR **text** field, output 1–3 tags representing the most relevant technical domains involved.

———— **Domain Taxonomy** ————

1. system - OS, filesystem, IO, memory, exceptions, underflow/overflow, serialization, path, symlink, encoding, datetime, unicode, container runtime, OS compatibility, networking
2. language - programming language correctness: types, encoding, unicode, regex, parsing, linting
3. architecture - design-level concerns: schema, refactor, modeling, OOP, system structure
4. data-processing - SQL/NoSQL, ETL, parquet, Spark, distributed data, batch, algorithmic workflows
5. compute-ml - ML/DL: torch, transformers, DNN/RNN
6. compute-gpu - GPU/CUDA/OpenGL/XLA/Triton, quantization, acceleration
7. infra - DevOps: env, containers, build, logging config, packaging, cloud, docker, registry, k8s, CI/CD, release
8. security - auth, permission, security fixes, vulnerabilities
9. observability - logging, metrics, tracing, telemetry
10. performance - perf fixes, caching, compression, throughput, latency
11. frontend - UI, DOM, rendering
12. document - documentation, explanation, annotation, docstrings
13. testing-quality - tests, coverage, reliability
14. api-design - API endpoints, backend interface design, REST, OpenAPI/Swagger
15. others - only if none above fits

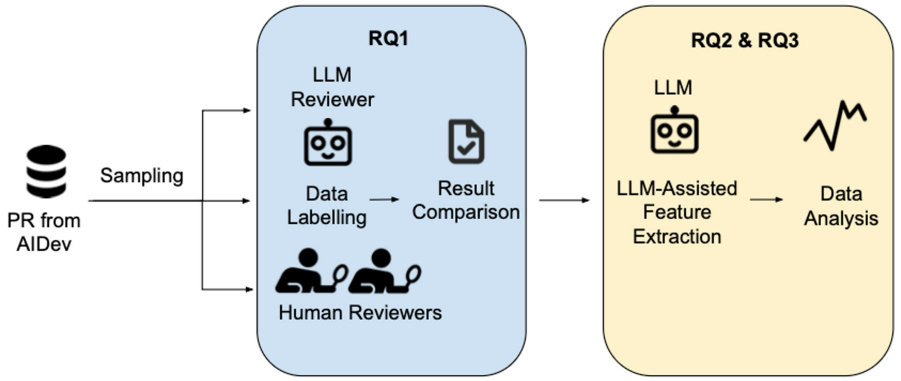
———— **Output Rules** ————

1. Return 1–3 tags, comma-separated, all lowercase, no spaces, no quotes.
2. Choose the most specific tags possible.
3. If multiple domains apply, order by relevance.
4. No explanations or commentary.
5. Output format: `<tag1,tag2,tag3>`

———— **Input Text** ————

`{text}`

Methodology



Study Design

- **Data Preprocessing:**
  - Using ***timestamp*** feature to define ***open, closed,*** and ***merge*** PRs
  - Following prior work, we treat merged PRs as the subset of closed PRs that have been successfully accepted
  - ***Accept Rate*** as the percentage of closed PRs that end in a merg state
  - Closed-but-unmerged ones as ***Rejected***
  - Distinguish ***long-term open PRs*** and ***recently opened*** **PRs**
    - Avoid bias better

Study Design

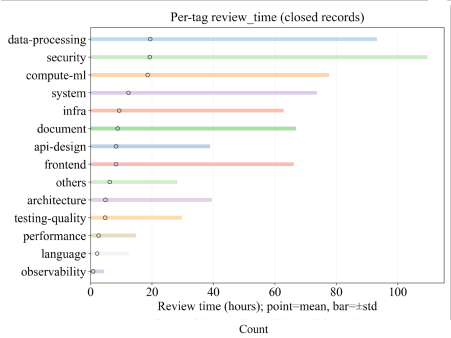
- **For RQ1:**
  - Sample 100 PRs to construct a **gold standard evaluation subset**
  - Two human reviewers label those PRs manually
  - Two human reviewers discuss disagreement then reconcile
  - Warm up by trying different prompts on the LLM
  - Use LLM re-labels the dataset and analyze the results
  - **Metrics**:
    - Exact Match
    - Precision (macro)
    - Recall (macro)
    - F1 (macro)
      - Didn't apply Krippendorff's alpha

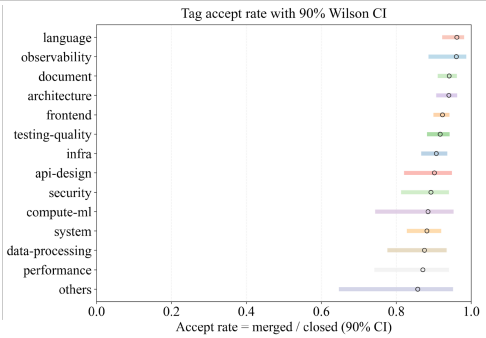
Study Design

- **For RQ2:**
  - We count ***open, closed,*** and ***merged*** PRs in the dataset
  - We calculate **the average closed, accept, and merged rate** for each domain with a **90% Wilson confidence interval** to better understand the dynamics
  - Analyze the result empirically
- **For RQ3:**
  - Also calculate the **average review time** for each domain with a 90% **Wilson confidence interval**
- We also try to find a **pattern of rejection** through keywords in PR messages frequency table as a greater implication for developers

RESULT

EM	Precision (macro)	Recall (macro)	F1 (macro)
0.40	0.710	0.730	0.692





Analysis and Answer to Research Question

The Accuracy of LLM-assistant Annotation

Sufficiently reliable for large-scale inference

Accept Rate across Domains and Influencing Factors

Localized, predictable changes → higher acceptance (e.g documents, language)

High-risk or multi-component domains → lower accept rate (e.g system, performance)

Configuration-heavy domains → more rejections (frequent words: build, npm, hosting, yaml)

Review Time across Domains and Influencing Factors

Privacy concern (e.g security)

Semantic complexity (e.g ML, data-processing)

Dependency/configuration sensitivity (e.g system, infra)

Limitation and Future Work:

- LLM labels may **miss** subtle or secondary domains.
- Repo governance differences may confound acceptance patterns.
- Review-time outliers may reflect noise, not difficulty.
- Open-source data may not generalize to enterprise settings.

- Build **domain-aware agents** with better reasoning modules.
- **Expand taxonomy** with finer-grained semantic layers.
- Study **human–AI collaboration** and trust dynamics.
- Develop models to **predict PR acceptance** and review time.

Summary

- Built a 15-domain taxonomy and validated GPT-5 labeling with **0.69 macro-F1**.
- Annotated **2,000 Agentic-PRs** and revealed **large acceptance gaps (≈30–80%)** across domains.
- Identified heavy-tailed review cycles, with complex domains averaging **70–100+ hours**.
- Demonstrated that domain complexity, dependency depth, and risk strongly shape merge outcomes.

UNIVERSITY OF WATERLOO

Thank you