TU Delft

Delft University of Technology

Document Version
Final published version

Licence
CC BY

Citation (APA)
Asgari, A., Panichella, A., Derakhshanfar, P., & Olsthoorn, M. (2027). What challenges do developers face in AI agent systems? An empirical study on Stack Overflow & GitHub issues. *Journal of Systems and Software*, *243*, Article 113108. https://doi.org/10.1016/j.jss.2026.113108

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

*This work is downloaded from Delft University of Technology.*

The Journal of Systems and Software 243 (2027) 113108

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

# What challenges do developers face in AI agent systems? An empirical study on Stack Overflow & GitHub issues☆

Ali Asgari [a],*, Annibale Panichella [a], Pouria Derakhshanfar [b], Mitchell Olsthoorn [a]

[a] Delft University of Technology, Delft, The Netherlands
[b] JetBrains Research, Amsterdam, The Netherlands

ARTICLE INFO

*Keywords:*
AI agents
Empirical software engineering
Topic modeling
Software maintenance

ABSTRACT

AI Agents have rapidly gained prominence in both research and industry as systems that extend large language models with planning, tool use, memory, and goal-directed action. Despite this progress, the development and maintenance of Agent systems present recurring engineering difficulties that are not yet well characterized in developer-facing evidence. To address this gap, this study analyzes developer discussions on Stack Overflow and failure reports from GitHub issue trackers associated with widely used Agent frameworks. For Stack Overflow, an Agent-focused corpus is constructed through tag expansion and filtering, latent themes are derived using LDA-MALLET, and topics are manually validated and labeled. For GitHub, a taxonomy of issue themes is developed to capture deployment-time failures and maintenance burdens. Analysis across both platforms identifies seven Stack Overflow topics (comprising 28 subtopics) and thirteen GitHub issue topics, which are synthesized into five overarching families of major challenges related to developing and maintaining AI agents: (1) environment, platforms, and dependency management; (2) retrieval, embeddings, and Agent memory; (3) orchestration and execution control; (4) interaction contracts between models and tools; and (5) runtime reliability and operational robustness. Topic popularity and difficulty are quantified, revealing that widely discussed issues, such as installation and prompting, are often resolved more quickly, whereas retrieval- and orchestration-related challenges are less visible, more complex, and tend to persist as ongoing maintenance burdens on GitHub.

## 1. Introduction

AI Agents have recently emerged as a paradigm for constructing AI-enabled software systems that extend large language models (LLMs) with capabilities (Acharya et al., 2025) such as tool invocation, persistent state, planning, and multi-step execution. Unlike conventional AI components, Agent-based systems integrate autonomous decision-making with external tools (Rizk et al., 2018), dynamic environments, and long-running workflows, introducing new sources of complexity (Acharya et al., 2025) in configuration, orchestration, and runtime behavior. As a result, these systems challenge established software engineering practices for integration, testing, debugging, and maintenance. Driven by these capabilities, AI Agents are increasingly adopted in research prototypes (Yang et al., 2024) as well as production systems,[1] supported by a rapidly growing ecosystem of frameworks and platforms.

Although prior work has examined AI Agents through surveys, conceptual frameworks, and system descriptions, empirical evidence on the engineering challenges developers encounter in practice remains limited. Existing studies largely rely on author-driven experiences, controlled evaluations, or small-scale case analyses, which provide valuable insights but do not capture the diversity and frequency of challenges faced by developers in real-world settings (Cemri et al., 2025; Deng et al., 2025; Zou et al., 2025; de Witt, 2025; Gabriel et al., 2025; Kolt, 2025; Tian et al., 2025; Du and Ding, 2021; Du et al., 2025; Sapkota et al., 2025). As a result, it is still unclear which challenges are most prevalent, which are most difficult to resolve, and how these challenges vary across different Agent-based systems and platforms.

To address this gap, this study analyzes discussions from Stack Overflow and GitHub. Stack Overflow is a large developer-oriented Q&A platform with approximately 60 million questions and answers and around 30 million registered users. It has been widely utilized

☆ Editor: Gabriele Bavota.

* Corresponding author.
*E-mail addresses:* a.asgari-2@tudelft.nl (A. Asgari), a.panichella@tudelft.nl (A. Panichella), pouria.derakhshanfar@jetbrains.com (P. Derakhshanfar), m.j.g.olsthoorn@tudelft.nl (M. Olsthoorn).

[1] E.g., https://www.jetbrains.com/junie/.

https://doi.org/10.1016/j.jss.2026.113108
Received 4 February 2026; Received in revised form 20 July 2026; Accepted 7 September 2026
Available online 12 September 2026
0164-1212/© 2026 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

in empirical software engineering research across domains such as Explainable AI (Sayyadnejad et al., 2025), concurrency (Ahmed and Bagherzadeh, 2018), GPU programming (Yang et al., 2023), Ruby programming (Akbarpour et al., 2025), and deep learning (Han et al., 2020).

GitHub is widely-used platform for collaborative software development hosting over 400 million repositories and serving more than 100 million developers worldwide. It has been used to gather software analytics to study wide-spread development practices, such as vulnerability fixes (Bhandari et al., 2021), self-admitted technical debt detection (OBrien et al., 2022), and automate code review (Lin et al., 2024).

In this paper, we leverage Stack Overflow and GitHub as complementary sources to identify and systematically characterize the technical and practical challenges developers encounter when building and maintaining AI Agent systems. By analyzing both developer questions and issue reports, we aim to capture challenges arising during agent development, maintenance, and operation. The research is structured around the following research questions:

**RQ1**: *What topics do AI Agent developers ask about?*
**RQ2**: *Which topics are the most popular and the most difficult?*
**RQ3**: *How do challenge signatures differ across AI Agent ecosystem domains?*

To address **RQ1**, we mined Stack Overflow at scale to construct an Agent-focused corpus through tag expansion and filtering, resulting in 3191 unique questions with accepted answers. Topic modeling was performed using LDA-MALLET (McCallum, 2002), producing a taxonomy of seven high-level topics and 28 subtopics. A statistically representative sample of 343 questions was selected using Cochran's sample size formula with finite-population correction (Woolson et al., 1986) and Neyman allocation (Neyman, 1938) for manual validation. In addition, we extracted 64,098 GitHub issues from 18 widely used AI Agent repositories. The same LDA-MALLET pipeline was applied to derive a complementary taxonomy of 13 issue topics, capturing implementation, integration, and maintenance failures in deployed Agent systems. These topics were further validated through manual analysis of a stratified sample of 382 GitHub issues, selected using Neyman allocation to preserve proportional topic representation. Finally, insights from Stack Overflow and GitHub were synthesized into five higher-level families of major Agent-specific challenges: (1) environment, platforms, and dependency management; (2) retrieval, embeddings, and agent memory; (3) orchestration and execution control; (4) interaction contracts between models and tools; and (5) runtime reliability and operational robustness. These families generalize across platform-specific manifestations and capture the core technical difficulties that recur throughout the AI Agent development lifecycle.

To answer **RQ2**, we measure how popular and difficult different topics are, using established metrics (Uddin et al., 2021; Alamin et al., 2023) on Stack Overflow and GitHub issues. On Stack Overflow, developers mostly focus on *Installation & Dependency Conflicts* and *Prompt & Output Engineering*. The topics developers find most challenging are *RAG Engineering*, *Document Embeddings & Vector Stores*, and *Orchestration*, which take longer to resolve and often go unanswered, even though they are less frequently discussed. GitHub issues reveal complementary patterns in real-world system use While a large volume of issues concerns workflows, platforms, and integrations, the most difficult issues are related to *Agent Orchestration & Invocation Semantics*, *Policy & Template Enforcement Friction*, and *Platforms & Integrations*. Across both platforms, configuration- and prompt-related issues tend to be resolved relatively quickly, whereas challenges related to information retrieval and agent coordination persist longer and remain more difficult to address.

To answer **RQ3**, we examined GitHub issues across 18 AI Agent repositories spanning distinct ecosystem domains, including core orchestration frameworks, multi-Agent coordination systems, retrieval components, workflow platforms, user interface layers, and end-user Agent applications. Our analysis reveals that each group faces different types of challenges. Infrastructure- and orchestration-focused systems exhibit fewer reported issues overall, but these issues tend to be more complex and persist longer, particularly those related to integration and execution control. In contrast, workflow and user interface platforms experience a higher volume of issues that are typically resolved more quickly. End-user Agent applications surface compound challenges that originate from upstream architectural and tooling decisions, showing how design choices in Agent frameworks propagate to operational reliability and user-facing behavior.

We discuss the implications of these findings for researchers, practitioners, and educators. The results provide empirical guidance for prioritizing reliability, observability, and contract enforcement in the design of Agent tooling, documentation, and evaluation methods. To support transparency and replication, all datasets, and topic models are released in a public replication package.[2]

## 2. Methodology

This section describes the data sources and analytical procedures used in this study (Fig. 1). We analyze Stack Overflow posts and GitHub issues using parallel mining and topic modeling pipelines, followed by a synthesis of the results across both sources. The following sections detail the data collection, filtering, and analysis steps for each platform.

### 2.1. Stack-overflow mining

Stack Overflow contains questions, answers, and metadata, but it does not provide explicit topic annotations for AI Agent development. We therefore apply a multi-step filtering pipeline followed by LDA-based topic modeling, a well-established technique for analyzing software engineering artifacts and corpora.

**Step 1**: *StackExchange Data Explorer for Stack Overflow.* We collect data using the Stack Exchange Data Explorer,[3] which provides access to official Stack Exchange data dumps (Sayyadnejad et al., 2025). We focus on Stack Overflow, as other Stack Exchange sites (e.g., Artificial Intelligence, Data Science, Machine Learning) currently contain limited AI Agent–specific discussions, whereas Stack Overflow offers a large and diverse corpus well-suited for mining practitioner challenges, consistent with prior software engineering studies (Elshan et al., 2024; Ahmed et al., 2025; Bi et al., 2021).

**Step 2**: *Identify AI Agent Tags.* To identify the most relevant AI Agent-related tags, we followed established tag-expansion strategies from prior studies (Abdellatif et al., 2020; Bagherzadeh and Khatchadourian, 2019; Sayyadnejad et al., 2025; Rosen and Shihab, 2016). We started querying with the *Agent* tag alone, which returns 1379 questions. To broaden coverage, we expanded our tag set by using the *Agent* co-occurring tags, focusing on avoiding adding noise to our data, since these new tags should help us find AI Agent posts only. We extracted all the *Agent* co-occurring tags from Agent-tagged posts, resulting in 1047 unique co-tags. Next, we applied two heuristic measures to filter and retain only meaningful tags: Tag Relevance Threshold (TRT) and Tag Significance Threshold (TST) (Sayyadnejad et al., 2025; Rosen and Shihab, 2016). TRT measures how related a specific tag is to the Agent-tagged posts; TST measures how prominent a specific tag is in the Agent-tagged posts. We calculate the TRT and TST values as follows (Sayyadnejad et al., 2025; Rosen and Shihab, 2016):

$$\mathrm{TRT}_{\mathrm{tag}} = \frac{\#\ \text{AI Agent posts}}{\#\ \text{posts}}, \mathrm{TST}_{\mathrm{tag}} = \frac{\#\ \text{AI Agent posts}}{\#\ \text{posts in the initial tag set}} \quad (1)$$

[2] zenodo.org/records/17406358
[3] Stack Exchange Data Explorer : https://data.stackexchange.com.

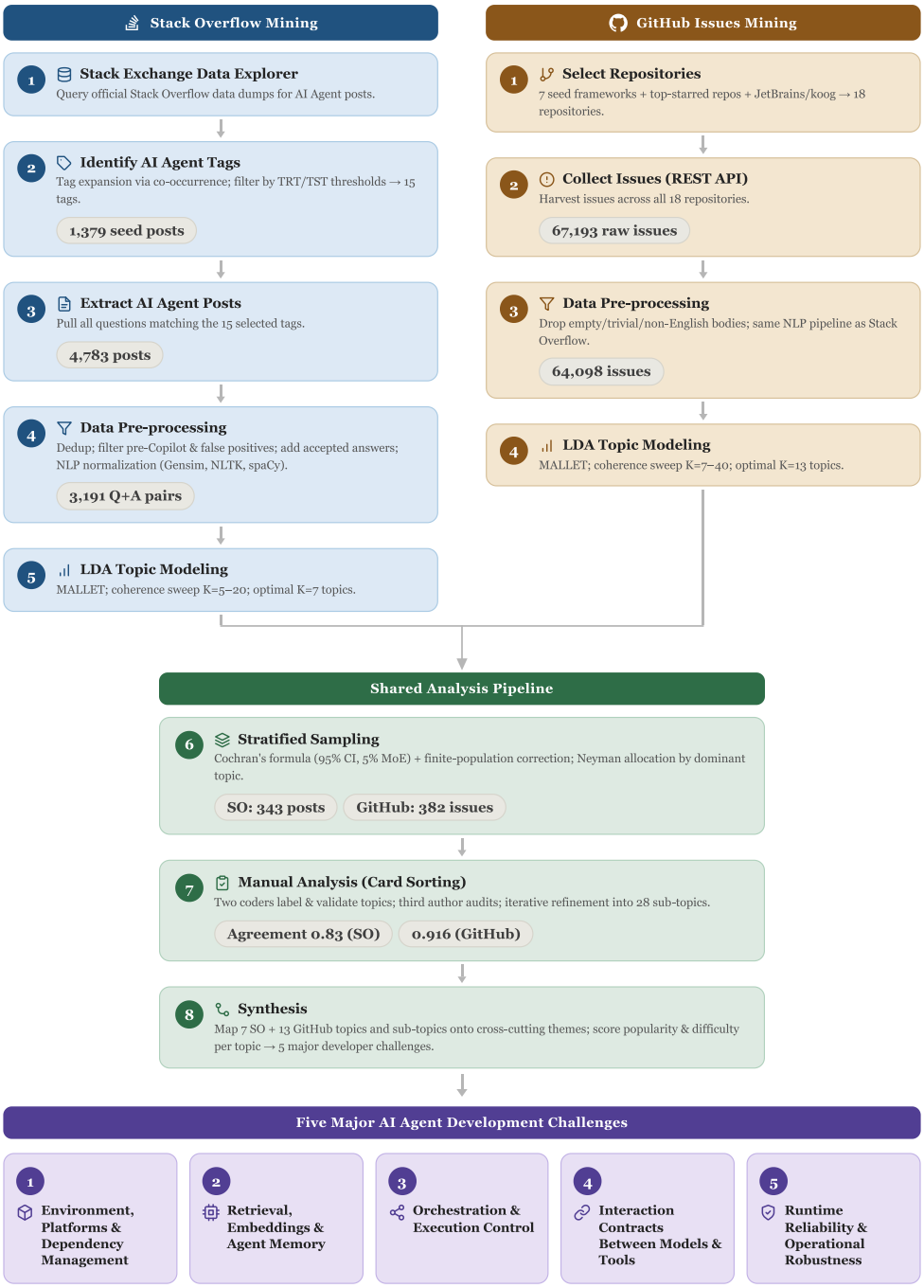



**Fig. 1.** Overview of the study methodology.

A tag was considered both relevant and significant if its corresponding TRT and TST exceeded predetermined thresholds. The first two authors, with expertise in LLM and AI Agent development (particularly for Software Engineering tasks), independently examined the tags under different TRT and TST numerical thresholds. For each tag, we inspected a randomly selected sample of posts to determine when the tags became less relevant to AI Agents; in addition, paying attention to the tag descriptions on Stack Overflow proved necessary and helpful. Ultimately, after testing TRT thresholds $(0.05, 0.10, 0.15, 0.20, 0.25, 0.30)$ and TST thresholds $(0.001, 0.002, 0.005, 0.010, 0.015, 0.020, 0.30)$, which align closely with previous studies (Sayyadnejad et al., 2025; Rosen and Shihab, 2016; Abdellatif et al., 2020; Uddin et al., 2021; Yang et al., 2023).

We also retained tags based on manual validation (Sayyadnejad et al., 2025; Abdellatif et al., 2020). *langchain-js* and retrieval-augmented-generation tag were included because manual verification showed that they consistently refer to AI Agent development, introduce no observable noise, and complement closely related tags that already met the thresholds (e.g., *langchain*, langchain-agents, and *rag*). This ensured coverage of JavaScript-based agent frameworks and equivalent terminology used for retrieval-augmented agent pipelines. This process resulted in a final set of 15 AI Agent related tags: {*agent*, *agents*, *multi-Agent*, *langgraph*, *langchain-agents*, *crewai*, *ms-autogen*, *phi-data*, *openai-agents*, *langchain*, *py-langchain*, *rag*, *retrievalqa*, *langchain-js*, retrieval-augmented-generation}.

**Step 3**: *Extract AI Agent Posts.* Using the selected tags, we extracted all Stack Overflow questions (*PostTypeId*=1) related to AI Agents, yielding 4783 posts with metadata, consistent with prior work (Tahir et al., 2018; Sayyadnejad et al., 2025).

**Step 4**: *Data Pre-processing.* Before conducting topic modeling, we performed several text pre-processing steps to remove noise from the dataset. First, we removed duplicate posts, since searching each tag separately may return the same question multiple times (a post can have multiple tags). This eliminated 740 duplicates. Next, we restricted the dataset to questions posted after the first major AI impact on software development, the release of GitHub Copilot (2021-06-21), which removed 1172 older posts. We then expanded the dataset by adding accepted answers from Stack Overflow using `PostTypeId=2`, yielding 530 accepted answers linked to 2871 questions, for a total of 3401 pairs of questions and answers. A final validation was conducted to eliminate false positives where the term Agent referred to unrelated concepts, such as Agent-based simulation (e.g., AnyLogic), DevOps and CI/CD execution agents (e.g., Jenkins build agents, GitHub Actions runners, Azure DevOps pipelines), reinforcement-learning agents (e.g., PPO, DQN, Gym-based environments), and robotics or control systems (e.g., ROS-based agents). To derive reliable exclusion criteria, the authors independently inspected random samples of the dataset and then jointly agreed on a consolidated filtering rule through discussion. After applying this rule, the remaining posts were manually re-checked to confirm relevance, resulting in a final dataset of 3191 AI Agent–related questions with accepted answers. Following established Stack Overflow mining practices (Ahmed and Bagherzadeh, 2018; Barua et al., 2014; Sayyadnejad et al., 2025), code blocks, HTML tags, URLs, and images were removed. A standard natural language processing pipeline was then applied, utilizing Gensim for tokenization and bigrams (Řehůřek and Sojka, 2010), NLTK for stopword removal (Loper and Bird, 2002), and spaCy for lemmatization of informative parts of speech (Honnibal et al., 2020). This process yielded a clean, normalized corpus suitable for topic modeling.

**Step 5**: *LDA Topic Modeling.* Latent Dirichlet Allocation (LDA) is widely used for topic modeling in software repositories (Hindle et al., 2016), including technical Q&A posts (Sayyadnejad et al., 2025; Barua et al., 2014) and issue reports. In this study, we employed the MALLET implementation of LDA (McCallum, 2002), a method widely applied in software engineering research due to its higher coherence scores compared to the Gensim library (Abdellatif et al., 2020; Li et al., 2021; Sayyadnejad et al., 2025). LDA leverages two crucial hyperparameters, $\alpha$ (alpha) and $\beta$ (beta), which govern the distribution of words across topics and the assignment of posts to topics (Sayyadnejad et al., 2025; Allamanis and Sutton, 2013; Uddin et al., 2021; Bagherzadeh and Khatchadourian, 2019; Ahmed and Bagherzadeh, 2018; Rosen and Shihab, 2016).

We adhered to conventional settings: $\alpha = 50/K$, where $K$ is the number of topics, and $\beta = 0.01$ (Bagherzadeh and Khatchadourian, 2019). These values are widely accepted in prior research and provide a robust benchmark for our experimental framework in topic modeling (Sayyadnejad et al., 2025). Selecting the optimal number of topics (K) is a key challenge in LDA (Panichella et al., 2013), as large K values yield overly specific topics while small K values produce generic ones. We evaluated K values from 5 to 20 using topic coherence, a metric shown to correlate with human judgment (Röder et al., 2015), and repeated LDA–MALLET training three times to reduce stochastic effects. Across all runs, K=7 consistently achieved the highest coherence, with a maximum score of 0.462. Although K=5,6, and 8 showed comparable performance, topics at K=7 were more distinct, a finding independently confirmed by the first two authors through inspection of the top 20 posts per topic. We therefore selected K=7 as the optimal number of topics.

**Step 6**: *Sampling.* To validate the LDA results while keeping manual analysis feasible, we applied a stratified sampling strategy that accounts for both topic volume and the model's assignment confidence, following prior empirical studies (Sayyadnejad et al., 2025; Asgari et al., 2025; Uddin et al., 2021; Ahmed and Bagherzadeh, 2018; Elshan et al., 2024; Abdellatif et al., 2020). We estimated the required sample size using Cochran's method with a 95% confidence level and a 5% margin of error, applying finite population correction due to the bounded corpus size (Woolson et al., 1986; Renthlei and Lallawmkima, 2025; Cochran, 1977). The resulting sample was proportionally allocated across topics using Neyman's allocation (Neyman, 1938; Kubiak and Kawalec, 2022), where within-topic variability was measured using the standard deviation of `Topic_Perc_Contrib`. Within each topic, records were sampled with probability proportional to this confidence score to prioritize high-confidence assignments while preserving topical diversity; the resulting allocation is summarized in Table 1.

**Table 1**
Neyman allocation by topic for Stack Overflow and GitHub Issues.

**Stack Overflow**
(Population $N = 3191$; $n_0 = 384.2$; Final Corrected Population $n = 343$)

| Topic | Stratum $N_h$ | SD $S_h$ | Alloc $n_h$ |
|---|---|---|---|
| 1 | 390 | 0.109269 | 41 |
| 2 | 554 | 0.116195 | 62 |
| 3 | 309 | 0.110661 | 33 |
| 4 | 415 | 0.107397 | 43 |
| 5 | 667 | 0.115997 | 75 |
| 6 | 312 | 0.119681 | 36 |
| 7 | 544 | 0.101866 | 53 |
| **Total** | **3,191** | | **343** |

**GitHub Issues**
(Population $N = 64,098$; $n_0 = 384.2$; Final Corrected Population $n = 382$)

| Topic | Stratum size | Within-SD | Alloc $n_h$ |
|---|---|---|---|
| 1 | 8177 | 0.141207 | 45 |
| 2 | 5612 | 0.161634 | 35 |
| 3 | 3846 | 0.146409 | 22 |
| 4 | 2479 | 0.172917 | 17 |
| 5 | 3769 | 0.162406 | 24 |
| 6 | 9125 | 0.150321 | 53 |
| 7 | 4327 | 0.167707 | 28 |
| 8 | 2625 | 0.150413 | 15 |
| 9 | 3756 | 0.155045 | 22 |
| 10 | 5713 | 0.159515 | 35 |
| 11 | 4048 | 0.173298 | 27 |
| 12 | 4874 | 0.143373 | 27 |
| 13 | 5747 | 0.145429 | 32 |
| **Total** | **64,098** | | **382** |

**Step 7**: *Manual Analysis.* Following prior qualitative topic-interpretation studies (Li et al., 2021; Ahmed and Bagherzadeh, 2018; Bagherzadeh and Khatchadourian, 2019; Alamin et al., 2023; Sayyadnejad et al., 2025), we used a card-sorting approach to label and validate the LDA topics (Fincher and Tenenberg, 2005; Uddin et al., 2021). The first two authors independently reviewed the sampled posts and topic keywords, while the third author audited the process for consistency. After discussion, the authors agreed on topic labels with a Cohen's Kappa of $\kappa \approx 0.83$ (McHugh, 2012). Using the Neyman-allocated samples (Table 1), the first two authors then iteratively identified and refined sub-topics, converging after multiple rounds of discussion on a final set of 28 sub-topics (Uddin et al., 2021; Sayyadnejad et al., 2025).

*Popularity & Difficulty of Topics.* We assessed the popularity of AI Agent topics using three standard Stack Overflow metrics widely adopted in prior studies: average view count, average post score, and average number of comments per question (Asgari et al., 2024; Sayyadnejad et al., 2025; Yang et al., 2016; Rosen and Shihab, 2016; Uddin et al., 2021; Alamin et al., 2023; Abdellatif et al., 2020). These

**Table 2**
Technology coverage of the GitHub and Stack Overflow datasets.

| GitHub Repository | Issues | Stack Overflow technology | Share (%) |
|---|---|---|---|
| langgenius/dify | 15,550 | LangChain | 70.6 |
| langchain-ai/langchain | 8927 | OpenAI | 24.2 |
| run-llama/llama_index | 6833 | ChromaDB | 9.6 |
| infiniflow/ragflow | 6186 | Hugging Face | 5.8 |
| lobehub/lobe-chat | 4984 | Llama models/tooling | 5.0 |
| microsoft/semantic-kernel | 4806 | LangGraph | 4.7 |
| OpenHands/OpenHands | 3611 | LlamaIndex | 4.1 |
| langflow-ai/langflow | 3051 | FAISS | 3.8 |
| microsoft/autogen | 2910 | Ollama | 3.8 |
| FlowiseAI/Flowise | 2442 | Azure OpenAI | 3.8 |
| agno-agi/agno | 1768 | Streamlit | 2.9 |
| crewAIInc/crewAI | 1651 | Pinecone | 2.6 |
| mem0ai/mem0 | 1295 | AutoGen | 2.3 |
| langchain-ai/langgraph | 1107 | Google GenAI | 2.0 |
| FoundationAgents/MetaGPT | 888 | Azure Cognitive Search | 2.0 |
| CopilotKit/CopilotKit | 775 | CrewAI | 1.5 |
| reworkd/AgentGPT | 469 | Semantic-Kernel | 1.5 |
| JetBrains/koog | 279 | Sentence-Transformers | 1.5 |

Notes: GitHub coverage is measured by the number of collected issues per repository. Stack Overflow coverage is measured as the percentage of AI Agent-related posts mentioning each technology. The GitHub repositories and Stack Overflow technologies are ranked independently; entries appearing on the same row do not imply a direct correspondence. Only the most frequently mentioned Stack Overflow technologies are shown to keep the overview compact.

metrics respectively capture topic visibility, perceived usefulness, and community engagement (Sengupta and Haythornthwaite, 2020; Yang et al., 2016). Topic difficulty was measured using two established proxies: (i) the proportion of questions without accepted answers, and (ii) the median time required to receive an accepted answer (Asgari et al., 2024; Sayyadnejad et al., 2025; Rosen and Shihab, 2016; Barua et al., 2014; Bagherzadeh and Khatchadourian, 2019; Abdellatif et al., 2020). Higher values on these metrics indicate greater difficulty. To enable comparison across topics, we followed prior work and computed fused popularity and difficulty scores by normalizing each metric relative to the global topic average and then averaging them (Uddin et al., 2021; Alamin et al., 2023). This aggregation provides a concise representation of overall visibility and resolution effort per topic.

### 2.2. GitHub issues mining

We complement the Stack Overflow analysis with GitHub issue reports from widely used AI Agent repositories. We seeded repository selection with seven core Agent frameworks frequently referenced in our Stack Overflow corpus (Table 10): *langchain*, *langgraph*, *crewAI*, *Flowise*, *llama_index*, *semantic-kernel*, and *autogen*. Following prior repository-mining studies (Braiek et al., 2018), we used GitHub stars as a proxy for repository visibility and adoption within the ecosystem. Using the GitHub REST API,[4] we searched GitHub Topics for repositories associated with the keywords *Agent* and *Agents* and selected the top five repositories per topic ranked by stars. To broaden coverage beyond Python-centric ecosystems, we additionally included *JetBrains/koog* to represent Agent development in Java and Kotlin. This process yielded 18 repositories (Table 11).

From these repositories, we collected 67,193 issues. We removed issues with empty bodies, trivial content (e.g., one-word reports), and non-English text, resulting in a final dataset of 64,098 issues. We processed the issue text using the same pre-processing pipeline as in the Stack Overflow analysis and applied the same LDA–MALLET configuration. To select the number of topics, we evaluated $K$ from 7 to 40 across three independent runs. The configuration with $K = 13$ produced the most stable and interpretable topics, ranking highest in two runs and second in the remaining run. As an additional qualitative check, the first two authors inspected the top 20 issues per topic under competing configurations ($K = 10$ and $K = 15$) and agreed that $K = 13$ yielded the clearest topic separation and labeling.

For manual validation, we applied the same stratified sampling strategy described in Step 6, using Neyman allocation with finite-population correction. This resulted in a representative sample of 382 GitHub issues (Table 1). The strata were defined using the dominant LDA topic of each issue, and the allocation considered both the number of issues in each topic and the within-topic variability of `Topic_Perc_Contrib`. We then analyzed these sampled issues using the same manual coding process as in Step 7. Agreement between the first two authors was strong (Cohen's $\kappa = 0.916$), and any disagreements were resolved through discussion with a third author.

For the quantitative topic counts reported (Table 1, and Fig. 3), each GitHub issue was assigned to a single *dominant topic*, defined as the topic with the highest contribution score. Therefore, the topic counts are mutually exclusive: each issue contributes to exactly one GitHub topic. Although LDA provides a probability distribution over all topics, we use the dominant-topic assignment to obtain interpretable and non-overlapping counts.

Fig. 3 reports the total number of GitHub issues assigned to each dominant topic across the full corpus of 64,098 issues, not only the manually validated sample. To distinguish open issues from the total number of issues, we used the `state` field returned by the GitHub REST API.[5] Issues with `state = open` were counted as open issues, while the total issue count includes both open and closed issues.

*Dataset coverage of AI Agent technologies.* To clarify the scope of the study, Table 2 summarizes the principal AI Agent repositories included in the GitHub dataset and the most frequently mentioned technologies in the Stack Overflow dataset. For GitHub, coverage is represented by the number of collected issues per repository. For Stack Overflow, it is represented by the percentage of AI Agent-related posts that mention each technology. Together, the two sources provide complementary coverage of orchestration frameworks, retrieval and vector-store technologies, model APIs and runtimes, workflow platforms, and end-user Agent applications. Presenting this overview before the topic analysis also makes the technological composition and boundaries of the datasets explicit.

## 3. Results

In this section, we present the results that address the three research questions introduced earlier.

### 3.1. RQ1: Taxonomy of challenges

To address **RQ1**, we constructed complementary topic taxonomies from Stack Overflow questions and GitHub issues related to AI Agent development. Stack Overflow questions reflect developer-reported difficulties encountered during usage and understanding, while GitHub issues document failures, limitations, and unexpected behavior observed during implementation and maintenance. We therefore interpret the extracted topics as recurring technical challenges faced by Agent developers.

At the first level, fine-grained, topic taxonomies were derived. Stack Overflow yielded seven topics comprising 28 subtopics (ST; Table 3), reflecting common difficulties related to usage and understanding. GitHub yielded thirteen issue topics (GT; Table 4) that capture recurring implementation, integration, and maintenance problems reported across Agent frameworks.

For clearer interpretation and systematic comparison across platforms, we further organized these fine-grained topics into *five higher-level families of major Agent challenges*: (i) dependency and environment

[4] https://api.github.com. We executed all queries on December 28, 2025.

[5] https://api.github.com

**Table 3**
Stack Overflow Data—Topics and subtopics with within-topic percentage usage.

| ST1 – Operational Issues (Runtime & Integration) *Topic share: 11.95% of posts* | | ST2 – Document Embeddings & Vector Stores *Topic share: 18.08% of posts* | | ST3 – Robustness, Reliability & Evaluation *Topic share: 9.62% of posts* | | ST4 – Orchestration & Workflow Control *Topic share: 12.54% of posts* | |
|---|---|---|---|---|---|---|---|
| Subtopic | % | Subtopic | % | Subtopic | % | Subtopic | % |
| Packaging & Environments (Build/CI/CD) | 17.07 | Ingestion & Context Modeling | 30.65 | Evaluation & Reproducibility | 9.09 | Graph & State Architecture | 26.83 |
| Model Runtimes & Performance | 12.20 | Vector Storage & Index Management | 25.81 | Learning & Planning Correctness | 15.15 | Execution Policies & Human Control | 31.71 |
| Connectivity & Cloud Operations | 24.39 | Retrieval Quality & Similarity Signals | 14.52 | Contracts, Tools & Grounding | 42.42 | Persistence, Concurrency & Real-Time I/O | 7.32 |
| Application/SDK Integration & Data Pipelines | 46.34 | Store Integrations & Runtime Ops | 29.03 | Operational Reliability & Instrumentation | 33.33 | Integration, Services & Observability | 34.15 |

| ST5 – Installation & Dependency Conflicts *Topic share: 21.90% of posts* | | ST6 – RAG Engineering *Topic share: 10.50% of posts* | | ST7 – Prompt & Output Engineering *Topic share: 15.45% of posts* | |
|---|---|---|---|---|---|
| Subtopic | % | Subtopic | % | Subtopic | % |
| Python Ecosystem Dependency Health | 20.00 | RAG Ingestion & Retrieval Modeling | 22.22 | Prompt Design & Chat Orchestration | 56.60 |
| Framework & SDK API Churn | 49.33 | RAG Indexes, Stores & Caching | 13.89 | JSON & Tool-Calling Output Contracts | 18.87 |
| Platform Compatibility Issues | 18.67 | RAG Scale, Cost & Runtime Operations | 30.56 | Memory & Response Shaping | 22.64 |
| Provider & Vector Service Compatibility Issues | 12.00 | RAG Design, Evaluation & Query Strategy | 33.33 | Multimodal Prompting & Output Control | 1.89 |

**Table 4**
GitHub Issues Data—Topic Taxonomy.

| Topic | Scope |
|---|---|
| GT1: Platforms & Integrations *(Stratum: 12.76%)* | Cross-platform support, SDKs, connectors, cloud services, and external system integrations. |
| GT2: Embeddings & Retrieval *(Stratum: 8.76%)* | Vector databases, embedding generation, indexing, similarity search, and RAG retrieval pipelines. |
| GT3: Model–Agent–Tool Interaction Contracts (Prompting & Parsing) *(Stratum: 6.00%)* | Failures to follow structured interaction contracts, including prompt formats, tool calling schemas, action inputs, multimodal inputs, and output parsing. |
| GT4: Version-Update Regressions *(Stratum: 3.87%)* | Bugs and behavior changes introduced by library, framework, or model version upgrades. |
| GT5: Dependencies, Installs, Imports, Environments *(Stratum: 5.88%)* | Package installation issues, dependency conflicts, import errors, environment setup, and version pinning. |
| GT6: Workflow, UI & API Behavior *(Stratum: 14.24%)* | Workflow logic, user interface behavior, REST/SDK API usage, request/response handling, and UX issues. |
| GT7: Feature Requests & Configurability *(Stratum: 6.75%)* | Requests for new features, enhancements, configuration options, and extensibility. |
| GT8: Tracebacks & Unhandled Exceptions *(Stratum: 4.10%)* | Crashes, stack traces, uncaught exceptions, and errors directly surfaced to users or logs. |
| GT9: LangChain Runtime & Integration Breakages *(Stratum: 5.86%)* | Runtime failures, breaking changes, and integration issues specific to the LangChain ecosystem. |
| GT10: Policy & Template Enforcement Friction *(Stratum: 8.91%)* | Prompt policies, guardrails, templates, schema enforcement, validation constraints, and compliance friction. |
| GT11: Runtime Execution & Operational Failures (Docker, CLI, Headless) *(Stratum: 6.32%)* | Execution failures in deployment and operations, including Docker, CLI tools, servers, and headless runs. |
| GT12: Agent Orchestration & Invocation Semantics *(Stratum: 7.60%)* | multi-Agent coordination, sequencing, routing, invocation order, concurrency, and orchestration logic. |
| GT13: Provider/Model Inconsistencies & Tooling Limits *(Stratum: 8.97%)* | Provider- or model-specific behavior differences, capability gaps, API mismatches, and tooling limitations. |

management, (ii) retrieval and memory pipelines, (iii) orchestration and execution control, (iv) interaction contracts, and (v) runtime reliability and operational robustness. These families abstract over platform-specific formulations and group topics by their underlying technical concerns rather than terminology.

We derived the five challenge families through open coding and consensus-based synthesis over the extracted topics. Two authors independently examined topic titles, top keywords, and a stratified sample of 15 representative Stack Overflow posts or GitHub issues per topic to identify the core difficulty each topic represents. The first two authors independently grouped related topics into higher-level challenge families by focusing on the underlying technical concerns. Inter-rater agreement was evaluated using Cohen's $\kappa$ (Cohen, 1960), yielding $\kappa = 0.825$ and a percent agreement of $P_o = 85.7\%$, indicating substantial agreement according to the guidelines of Landis and Koch (1977). The remaining disagreements were reviewed and resolved through discussion with a third author to ensure conceptual consistency.

Across both data sources, the most prominent challenges involve environment and dependency management (ST5; GT1, GT4, GT5), retrieval and knowledge integration within RAG pipelines (ST2, ST6; GT2, GT13), and Agent orchestration and execution control (ST4; GT12). Interaction contracts, encompassing prompting, tool invocation, and structured outputs, constitute another major family (ST7; GT3, GT10). Additionally, runtime reliability and operational robustness issues frequently arise during execution and deployment (ST1, ST3; GT8, GT9, GT11). Collectively, these families offer a unified, hierarchical perspective on the technical challenges that define contemporary AI Agent development.

Table 5 further clarifies the level of abstraction associated with the identified challenges. This distinction is significant because not all topics represent the same type of difficulty. Some challenges primarily relate to the maturity and evolution of current frameworks, whereas others reflect more persistent Agent-engineering concerns. Furthermore, two GitHub topics, GT6 and GT7, are classified as cross-cutting

**Table 5**
Interpretation of major and cross-cutting AI Agent challenge topics.

| Challenge/Topic | Associated topics | Interpretation | Rationale |
|---|---|---|---|
| Challenge 1: Environment, Platforms, and Dependency Management | ST5; GT1, GT4, GT5 | Framework/tooling-oriented | Mainly concerns installation, dependency conflicts, API changes, platform integration, and version regressions. |
| Challenge 2: Retrieval, Embeddings, and Agent Memory | ST2, ST6; GT2, GT13 | Core Agent challenge | Concerns retrieval quality, memory, grounding, embeddings, and knowledge integration in Agent workflows. |
| Challenge 3: Orchestration and Execution Control | ST4; GT12 | Core Agent challenge | Concerns multi-step execution, tool sequencing, workflow control, invocation semantics, and multi-Agent coordination. |
| Challenge 4: Interaction Contracts Between Models and Tools | ST7; GT3, GT10 | Core Agent challenge | Concerns prompt formats, structured outputs, tool schemas, model–tool interfaces, and contract violations. |
| Challenge 5: Runtime Reliability and Operational Robustness | ST1, ST3; GT8, GT9, GT11 | Mixed/operational | Concerns runtime failures, crashes, exceptions, and long-running execution; these are core operational concerns but are also affected by framework maturity. |
| GT6: Workflow, UI & API Behavior | GT6 | Cross-cutting topic | Captures issues related to workflow behavior, user interfaces, API behavior, request/response handling, and user-facing Agent execution. |
| GT7: Feature Requests & Configurability | GT7 | Cross-cutting topic | Captures requests for new features, configuration options, extensibility, customization, and better control over Agent systems. |

challenge topics because they capture recurring concerns regarding workflow behavior, interface design, configurability, and extensibility.

**Major Challenge 1:** *Environment, Platforms, and Dependency Management.* This challenge concerns configuring execution environments, resolving dependency conflicts, and integrating Agent frameworks with external platforms and services. It is Agent-specific because Agents combine multiple tightly coupled components — models, tools, memory stores, runtimes, and UI or service layers — where version drift or platform mismatches can break end-to-end execution. On Stack Overflow, this challenge appears primarily as *Installation & Dependency Conflicts* (**ST5**), where developers report failures caused by framework API churn or incompatible library upgrades (e.g., Q77338572, Q78613825). In contrast, GitHub issues emphasize platform-level and integration failures encountered during real deployments, including *Platforms & Integrations* (**GT1**), *Dependencies, Installs, Imports, Environments* (**GT5**), and *Version-Update Regressions* (**GT4**). Representative examples include cross-platform runtime and SDK integration failures in OpenHands (e.g., I11521), environment and import errors preventing Agent initialization in AutoGen (e.g., I2756), and breaking changes introduced by LangChain upgrades that invalidate existing Agent pipelines (e.g., I23517). This contrast highlights how Stack Overflow reflects setup-time confusion, while GitHub exposes deployment-time fragility and integration risk in production Agent systems.

**Major Challenge 2:** *Retrieval, Embeddings, and Agent Memory.* AI Agents rely on retrieval pipelines and persistent memory to ground decisions across multiple steps and interactions, making retrieval quality and state management central concerns. This challenge goes beyond general LLM applications, which typically retrieve context once per request, whereas Agents repeatedly query, update, and reason over stored knowledge. On Stack Overflow, these difficulties surface primarily as *Document Embeddings & Vector Stores* and *RAG Engineering* (**ST2**, **ST6**), where developers struggle with document chunking strategies, indexing semantics, and performance bottlenecks in vector stores (e.g., Q78015622, Q78505822). These discussions reflect design-time uncertainty about how to structure and persist Agent memory. In contrast, GitHub issues emphasize deployment-time and framework-level failures captured by *Embeddings & Retrieval* (**GT2**) and *Provider/Model Inconsistencies & Tooling Limits* (**GT13**). Representative issues include embedding and retrieval mismatches in LlamaIndex that break Agent memory pipelines (e.g., I13488) and provider-specific behavior that causes inconsistent retrieval results across Agent runs (e.g., I5980). Together, these findings show that retrieval in Agent systems is not merely a modeling concern but a core operational dependency that directly affects planning correctness, tool invocation, and long-horizon Agent behavior.

**Major Challenge 3:** *Orchestration and Execution Control.* This challenge captures how Agents coordinate tools, regulate execution flow, and maintain control across multi-step or multi-Agent workflows. Orchestration is inherently Agent-specific because it governs delegation, sequencing, and execution boundaries beyond a single LLM invocation. On Stack Overflow, this challenge appears as *Orchestration & Workflow Control* (**ST4**), where developers raise questions about controlling tool invocation policies and diagnosing missing or ambiguous observability in complex multi-Agent workflows (e.g., Q79332599, Q79771469). In contrast, GitHub issues reveal orchestration breakdowns that emerge during real-world execution, captured by *Agent Orchestration & Invocation Semantics* (**GT12**). For example, Semantic Kernel reports failures where Agents surface function-call intents during streaming but do not execute them, leading to stalled execution despite correct tool configuration (e.g., I9245). This contrast highlights how Stack Overflow surfaces design-time uncertainty and observability gaps, while GitHub exposes execution-time orchestration failures in deployed Agent systems.

**Major Challenge 4:** *Interaction Contracts Between Models and Tools.* AI agents rely on strict interaction contracts that govern prompts, tool calls, parameter schemas, templates, validation rules, and structured outputs. These contracts are essential because even minor violations can halt planning loops, disrupt execution, or cause silent failures that are difficult to diagnose. On Stack Overflow, such issues are discussed under *Prompt & Output Engineering* (**ST7**), where developers report challenges in enforcing JSON schemas, constraining model outputs, and reliably triggering tool calls (e.g., Q77227902, Q77362103). These discussions highlight prompt design and output validation as mechanisms for maintaining contract compliance during development. GitHub issues reveal the downstream consequences of contract violations in deployed systems, as captured by both *Model–Agent–Tool Interaction Contracts* (**GT3**) and *Policy & Template Enforcement Friction* (**GT10**). For example, in RagFlow, a framework upgrade removed an expected tool-side parameter class, causing agents to fail at runtime due to broken assumptions about tool interfaces (e.g., I9141). Similarly, LlamaIndex users report being unable to constrain the generated entity and relation types when constructing knowledge graphs,

highlighting the absence of enforceable schemas that align model outputs with downstream tool expectations (e.g., I10120). GT10 further captures cases where template, schema, or validation constraints are not reliably enforced across the model–tool boundary. For instance, a RagFlow issue reports that the `chunk.update` API does not preserve the expected `positions` field, demonstrating how schema-level expectations and validation behavior can disrupt downstream retrieval or document-processing workflows (e.g., I10944). Collectively, these issues illustrate how interaction contracts evolve from prompt-level usability concerns on Stack Overflow into challenges of correctness, robustness, validation, and versioning in real-world agent systems.

**Major Challenge 5:** *Runtime Reliability and Operational Robustness.* The final challenge concerns keeping AI Agents reliable during execution, deployment, and long-running operation. Agents are particularly vulnerable because they execute over extended periods, invoke external services, and maintain evolving state. On Stack Overflow, this challenge appears in *Operational Issues (Runtime & Integration)* and *Robustness, Reliability & Evaluation* (**ST1**, **ST3**), for example when permission or configuration errors invalidate evaluations (e.g., Q76815315) or malformed tool outputs repeatedly break execution (e.g., Q79134666). GitHub issues emphasize the operational consequences of these problems, captured by *Tracebacks & Unhandled Exceptions* (**GT8**), *Runtime Execution & Operational Failures* (**GT11**), and ecosystem-specific breakages such as LangChain runtime failures (**GT9**). Representative examples include repeated crashes in MetaGPT deployments (e.g., I1862), execution failures in OpenHands during long-running tasks (e.g., I907), and unhandled runtime errors in LangChain Agent pipelines (e.g., I7900).

*Workflow, UI & API Behavior* (**GT6**) constitutes the largest stratum in the GitHub Issues dataset (14.24%), underscoring its importance in practice. These issues primarily fall under Challenge 3 and Challenge 1, as they concern how Agent workflows are composed, triggered, and executed through user interfaces and APIs. Most cases arise from UI-centric Agent platforms — particularly langgenius/dify — where Agents are built and deployed via visual workflows rather than code alone (e.g., I15785). This reflects a shift toward production-grade, user-facing Agent systems, where orchestration complexity and integration failures increasingly surface at the workflow and interface layer.

*Feature Requests & Configurability* (**GT7**) covers a unique but related set of Agent challenges, making up 6.75% of GitHub issues. Rather than focusing on bugs or failures, these issues show that developers often need more ways to extend, customize, or control how Agents work and fit into their infrastructure. Many of these requests come up after Agents are used in real-world settings, where the built-in features no longer meet all needs. For example, developers might want more configuration options, better API access, or easier ways to connect Agents with existing company systems. One case is developers using langgenius/dify who asked for extra OpenAPI endpoints so they could manage Agent annotations through code, pointing to a gap between what the UI can do and what automation requires (see I6878). Most of these issues relate to Challenge 1 (integration with environments and platforms) and Challenge 3 (control over orchestration and execution), since they are about making Agent systems more adaptable and manageable beyond default setups. The rise of GT7 shows that as Agent platforms get more advanced, developers spend less time fixing basic errors and more time seeking flexibility, compatibility, and maintainability for the long term.

GT6 and GT7 are treated as cross-cutting topics because they do not correspond to a single underlying technical challenge. GT6 groups workflow, user-interface, and API issues that arise from several challenge families, including orchestration, integrations, and runtime behavior. Similarly, GT7 aggregates feature requests spanning multiple technical areas (e.g., orchestration, retrieval, memory, deployment, and tooling), reflecting desired capabilities rather than a single engineering challenge. By contrast, GT10 consistently concerns one technical phenomenon—interaction contracts between models, prompts, tools, templates, schemas, and validation rules—and therefore naturally belongs to the Interaction Contracts challenge family together with ST7 and GT3. Accordingly, GT6 and GT7 are presented as cross-cutting topics, while GT10 is included as a major challenge.

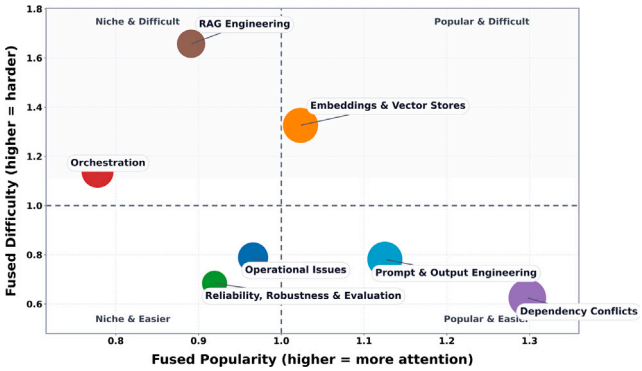



**Fig. 2.** Stack Overflow Fused Popularity & Fused Difficulty.

**Findings (AI Agent Topics).** AI Agent development involves a distinct and recurring set of technical challenges that cannot be fully explained by general LLM-based application development. These challenges span environment setup and dependency management, retrieval and memory pipelines, multi-Agent orchestration, model–tool interaction contracts, runtime reliability, and workflow- and configuration-level concerns. Their recurrence indicates that Agent-specific difficulty arises not only from autonomy, tool-mediated execution, and long-running behavior, but also from the need to compose, customize, and govern Agents through user interfaces, APIs, and extensible configurations rather than isolated framework flaws or misuse.

### 3.2. RQ2: Popularity & difficulty

To address **RQ2**, we quantify the popularity and difficulty of Agent-related topics on Stack Overflow and GitHub using the metrics defined in Section 2.

*Popularity of Agent Challenges (Stack Overflow).* Developer attention on Stack Overflow is concentrated around a small set of recurring topics (Table 7). The most popular challenges are *Installation & Dependency Conflicts* (**ST5**), followed by *Prompt & Output Engineering* (**ST7**) and *Document Embeddings & Vector Stores* (**ST2**). These topics reflect common entry points into Agent development, where practitioners seek help configuring environments, controlling Agent behavior, and enabling retrieval-based memory. In contrast, *Orchestration* (**ST4**) and *RAG Engineering* (**ST6**) receive comparatively less attention, suggesting that advanced Agent workflows affect a smaller, more specialized subset of developers.

*Difficulty of Agent Challenges (Stack Overflow).* Difficulty on Stack Overflow follows a different pattern as shown in Fig. 2. The hardest topics are *RAG Engineering* (**ST6**), *Document Embeddings & Vector Stores* (**ST2**), and *Orchestration* (**ST4**), which exhibit long resolution times and a high share of unanswered questions. These challenges involve multi-step reasoning, persistent state, and coordination across tools, capabilities that are intrinsic to Agent systems. By contrast, although *Installation & Dependency Conflicts* (**ST5**) is the most frequently discussed topic, it is among the easiest to resolve, indicating that such issues are widespread but relatively well understood once the root cause is identified (see Fig. 2).

*Issue Volume and Resolution Difficulty (GitHub).* GitHub issues capture a different perspective, reflecting engineering workload and maintenance burden in production Agent frameworks. The largest issue

**Table 6**
Repository-normalized sensitivity analysis of GitHub topic distributions.

| Topic | Raw Share | Avg. Repo Share | Repos | Avg. Repo Open% | Top Repo | Top Repo Share | Top 3 Share | Interpretation |
|---|---|---|---|---|---|---|---|---|
| GT1 | 12.8% | 17.3% | 18/18 | 24.6% | semantic-kernel | 18.6% | 42.0% | Broad ecosystem-level topic. |
| GT2 | 8.8% | 6.7% | 18/18 | 16.9% | llama_index | 61.2% | 85.0% | Broadly present, but concentrated in RAG repositories. |
| GT3 | 6.0% | 8.0% | 18/18 | 15.1% | langchain | 31.5% | 54.1% | Broad ecosystem-level topic. |
| GT4 | 3.9% | 4.8% | 13/18 | 17.0% | langchain | 70.8% | 99.5% | Strongly concentrated in core orchestration. |
| GT5 | 5.9% | 8.6% | 18/18 | 10.2% | autogen | 22.3% | 50.5% | Broad dependency/setup concern. |
| GT6 | 14.2% | 3.4% | 8/18 | 22.1% | dify | 99.8% | 99.9% | High-volume but repository-specific. |
| GT7 | 6.8% | 2.4% | 17/18 | 14.8% | dify | 94.5% | 97.3% | Mainly concentrated in visual platforms. |
| GT8 | 4.1% | 5.3% | 18/18 | 17.3% | lobe-chat | 31.8% | 55.9% | Broad runtime exception concern. |
| GT9 | 5.9% | 3.2% | 16/18 | 13.2% | langchain | 76.5% | 92.4% | LangChain-centered ecosystem issue. |
| GT10 | 8.9% | 6.1% | 15/18 | 10.1% | ragflow | 81.5% | 98.3% | Important but RAG-concentrated. |
| GT11 | 6.3% | 9.2% | 18/18 | 16.3% | OpenHands | 50.5% | 65.4% | Broad operational reliability issue. |
| GT12 | 7.6% | 12.2% | 18/18 | 20.2% | semantic-kernel | 32.6% | 50.2% | Broad orchestration concern. |
| GT13 | 9.0% | 12.8% | 18/18 | 16.8% | langflow | 25.6% | 64.0% | Broad provider/model concern. |

Notes: Raw Share is the topic's share in the full GitHub issue corpus. Avg. Repo Share first computes each topic's share within each repository and then averages across repositories, giving each repository equal weight. Avg. Repo Open% is the average open-issue rate computed within repository-topic pairs. Top Repo Share and Top 3 Share indicate how concentrated a topic is in its largest contributing repositories. Repository names are shortened for readability.

volumes are observed for *Workflow, UI & API Behavior* (**GT6**), *Platforms & Integrations* (**GT1**), and *Policy & Template Enforcement Friction* (**GT10**), highlighting the operational challenges of deploying user-facing, platform-integrated Agents. Resolution difficulty is most pronounced for *Policy & Template Enforcement Friction* (**GT10**), *Agent Orchestration & Invocation Semantics* (**GT12**), and *Platforms & Integrations* (**GT1**), which exhibit high open-issue rates and prolonged activity. In contrast, topics such as *Workflow, UI & API Behavior* (**GT6**) and *Feature Requests & Configurability* (**GT7**) tend to close quickly, indicating that while frequent, these issues are often localized and easier for maintainers to address.

*Repository-normalized sensitivity analysis of GitHub topic distributions.* To assess how strongly the GitHub topic distribution is influenced by repository size, we conducted a repository-normalized sensitivity analysis (Table 6). We first computed the proportion of issues assigned to each topic within every repository and then averaged these proportions across repositories, giving each repository equal weight regardless of its issue volume. We also report the number of repositories in which each topic appears and the shares contributed by the largest repository and the three largest repositories. Finally, we compare open-issue rates at the repository–topic level to avoid allowing repositories with large issue volumes to dominate the aggregate rate.

The comparison shows that raw issue volume does not always reflect how broadly a topic occurs across the analyzed repositories. For example, *Workflow, UI & API Behavior* (GT6) has the largest raw issue count, but 99.8% of these issues originate from a single repository, *langgenius/dify*. *Feature Requests & Configurability* (GT7), *LangChain Runtime & Integration Breakages* (GT9), and *Policy & Template Enforcement Friction* (GT10) are also concentrated in a small number of repositories. Their high volume therefore reflects substantial maintenance activity in particular parts of the ecosystem rather than a challenge that is equally prevalent across repositories. In contrast, *Platforms & Integrations* (GT1), *Agent Orchestration & Invocation Semantics* (GT12), and *Provider/Model Inconsistencies & Tooling Limits* (GT13) appear in all 18 repositories and retain relatively high shares after repository normalization. GT1 and GT12 also have comparatively high average repository-level open-issue rates. Although open-issue rates may be influenced by maintainer practices and project maturity, these results suggest that the prominence of integration and orchestration issues is not solely an artifact of a few large repositories. Overall, the sensitivity analysis distinguishes broadly recurring concerns from topics whose raw volume is driven mainly by particular repositories or domains.

*Stack Overflow vs. GitHub Comparison.* Taken together, the two platforms reveal complementary views of Agent difficulty. Stack Overflow highlights where developers struggle conceptually and during early development, with retrieval, orchestration, and RAG emerging as the hardest topics despite limited visibility. GitHub, in contrast, exposes where Agent systems fail at scale: orchestration semantics, policy enforcement, and platform integration translate into sustained maintenance effort and unresolved issues. Notably, *Orchestration* shifts from being niche-but-hard on Stack Overflow (***ST4***) to a mid-to-high volume and difficult topic on GitHub (**GT12**), reflecting its growing importance as Agents move from prototypes to production systems (see Table 8).

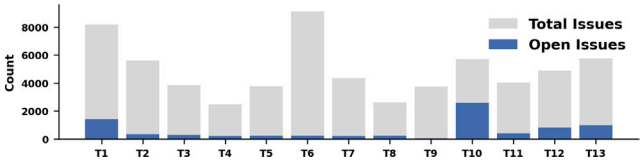



**Fig. 3.** GitHub Issues Topics.

**Findings (Popularity & Difficulty).** Easy-to-fix configuration and prompting issues dominate attention, while retrieval- and orchestration-centric challenges are less visible but substantially harder. As Agent systems transition into production, these harder challenges reappear on GitHub as prolonged operational and maintenance burdens, underscoring that Agent difficulty stems from autonomy, statefulness, and tool-mediated execution rather than isolated implementation errors.

*Importance and difficulty of major challenge families.* Table 9 brings together the existing Stack Overflow and GitHub metrics for the five major challenge families and the two cross-cutting GitHub topics. The purpose is to distinguish topics that are merely frequent from those that also appear difficult to resolve. We do not combine the two platforms into a single score because they reflect different forms of difficulty. Stack Overflow captures help-seeking difficulty through unresolved questions and time to an accepted answer, whereas GitHub captures maintenance difficulty through open-issue rates, closing time, open-issue age, and activity span.

The comparison shows that popularity and difficulty do not necessarily coincide. Environment and setup problems, along with workflow and user-interface issues, occur frequently, but many are resolved relatively quickly. Retrieval, orchestration, model–tool contracts, policy enforcement, provider inconsistencies, and runtime breakages are less uniform in volume, yet they tend to remain unresolved for longer or require sustained maintenance activity. The most consequential challenges are therefore not simply the largest topics, but those that combine substantial developer attention with unresolved questions, long resolution times, or prolonged activity across the two platforms.

**Findings (Importance vs. Difficulty).** Importance and difficulty do not always coincide. Some of the most prevalent topics (e.g., environment and setup) are resolved relatively quickly, whereas retrieval, orchestration, interaction contracts, provider inconsistencies, and runtime failures consistently score higher on one or more difficulty indicators, including unresolved questions, time to accepted answer, open-issue rates, and issue age.

**Table 7**
Popularity and difficulty of AI Agent topics.

| Topic | Views (avg) | Comments (avg) | Answers (avg) | Score (avg) | No accepted ans. (share) | Median hrs. to accepted | Fused Popularity | Fused Difficulty |
|---|---|---|---|---|---|---|---|---|
| 1. Operations (Runtime & Integration) | 1759.70 | 1.082 | 0.915 | 0.848 | 0.827 | 22.52 | **0.97** | **0.79** |
| 2. Document Embeddings & Vector Stores | 2426.09 | 0.626 | 0.965 | 1.265 | 0.805 | 65.38 | **1.02** | **1.33** |
| 3. Robustness, Reliability & Evaluation | 1441.01 | 1.031 | 0.855 | 0.953 | 0.772 | 16.92 | **0.92** | **0.68** |
| 4. Orchestration | 1259.11 | 0.768 | 0.734 | 0.905 | 0.884 | 46.93 | **0.78** | **1.14** |
| 5. Installation & Dependency Conflicts | 3231.54 | 1.166 | 1.096 | 1.200 | 0.785 | 11.71 | **1.30** | **0.62** |
| 6. RAG Engineering | 1677.71 | 0.678 | 0.746 | 1.286 | 0.884 | 87.44 | **0.89** | **1.66** |
| 7. Prompt & Output Engineering | 2621.59 | 0.778 | 0.975 | 1.409 | 0.836 | 21.51 | **1.13** | **0.78** |
| *All topics (avg)* | **2059.54** | **0.876** | **0.898** | **1.124** | **0.828** | **38.92** | **1.00** | **1.00** |

**Table 8**
Issue metrics by topic (medians). Close d = median time-to-close (days, closed issues only). Open d = median age (days, open issues only). Act h = median activity span (hours).

| Topic | Open% | Close d | Open d | Act h | Com |
|---|---|---|---|---|---|
| T1 | 17.4 | 39.9 | 214.2 | 1064.6 | 2 |
| T2 | 6.2 | 35.4 | 217.0 | 1304.7 | 3 |
| T3 | 7.7 | 50.4 | 310.6 | 1647.4 | 3 |
| T4 | 8.7 | 32.4 | 133.3 | 1331.1 | 2 |
| T5 | 6.5 | 6.0 | 211.4 | 300.2 | 3 |
| T6 | 2.7 | 0.4 | 48.9 | 24.9 | 2 |
| T7 | 4.9 | 1.8 | 125.1 | 115.6 | 1 |
| T8 | 9.2 | 4.1 | 247.7 | 1352.5 | 4 |
| T9 | 0.9 | 98.4 | 185.3 | 2382.4 | 3 |
| T10 | 44.9 | 2.2 | 253.8 | 89.1 | 3 |
| T11 | 10.3 | 6.7 | 270.1 | 342.6 | 3 |
| T12 | 16.3 | 15.0 | 161.5 | 550.6 | 2 |
| T13 | 17.2 | 11.6 | 244.8 | 893.4 | 3 |

### 3.3. RQ3: Domains of the AI agent ecosystem

To address RQ3, the GitHub repositories listed in Table 11 were organized into seven ecosystem domains according to their functional roles and intended uses within the AI Agent landscape. These domains clarify the positioning of Agent technologies in the ecosystem, encompassing core orchestration frameworks, memory components, visual workflow platforms, and end-user Agent applications. Domain assignments were made through a collaborative, consensus-driven qualitative analysis of the repositories in Table 11. Three authors jointly examined each repository's README file, description, and, when available, official project documentation or website to determine its primary purpose, target users, and architectural role.

This step was designed to categorize repositories into ecosystem domains rather than assign new labels to individual GitHub issues. The issue-level topics used in RQ3 were previously obtained from the LDA analysis described in RQ1. Thus, repository documentation and project intent served as the primary sources for domain assignment, while issue inspection functioned as a contextual validation step.

To complement the repository documentation, we reviewed a random sample of 15 GitHub issues per repository to understand typical usage scenarios and failure modes. Then, the authors iteratively discussed and refined domain boundaries until consensus was reached for all repositories. This consensus-based approach was selected because ecosystem domains are interpretive constructs that depend on repository intent and usage context rather than isolated technical features. The resulting seven domains represent a structured synthesis grounded in repository documentation, observed issue patterns, and shared expert judgment.

To further assess the stability of this categorization, the three authors conducted an additional validation check by increasing the number of inspected issues from 15 to 50, randomly selected per repository. Increasing the sample from 15 to 50 randomly selected issues per repository did not result in any changes to the domain assignments, suggesting that the categorization is stable with respect to the amount of issues considered. The most informative evidence for assigning domains was derived from each repository's stated purpose, README, description, and official documentation, while the sampled issues primarily served to confirm how the repositories were used and discussed in practice.

Fig. 4 summarizes the distribution of GitHub issue topics across the seven ecosystem domains. Each domain exhibits a distinct challenge profile, yet several topics recur across multiple domains. Accordingly, the following analysis addresses both the dominant topic in each domain and the broader topic distribution that characterizes its role within the AI Agent ecosystem.

**UI/Copilot Components** are concerned with embedding Agents into interactive user experiences. Although primarily represented by a single repository (*CopilotKit*, Table 11), this domain is characterized by *Agent Orchestration & Invocation Semantics* (GT12), *Platforms & Integrations* (GT1), and *Runtime Execution & Operational Failures* (GT11). It also encompasses *Dependencies, Installs, Imports, Environments* (GT5), *Provider/Model Inconsistencies & Tooling Limits* (GT13), and *Model–Agent–Tool Interaction Contracts* (GT3). These findings indicate that UI/Copilot components introduce not only interface-level concerns but also orchestration, integration, runtime, and provider-related challenges when Agents are embedded into user-facing software.

**Memory & State Components**, represented primarily by *mem0*, are dominated by *Embeddings & Retrieval* (GT2), *Platforms & Integrations* (GT1), and *Provider/Model Inconsistencies & Tooling Limits* (GT13). The domain also includes issues related to *Dependencies, Installs, Imports, Environments* (GT5), *Model–Agent–Tool Interaction Contracts* (GT3), and *Runtime Execution & Operational Failures* (GT11). This pattern demonstrates that persistent memory is closely linked to the broader Agent stack, relying on retrieval infrastructure, provider behavior, integration APIs, and reliable runtime execution. These challenges arise from maintaining persistent Agent memory across providers and execution contexts, and they tend to persist due to the architectural complexity of state management.

**Multi-Agent Coordination Frameworks**, including *AutoGen*, *CrewAI*, *MetaGPT*, and *Agno*, exhibit a high concentration of issues in *Dependencies, Installs, Imports, Environments* (GT5), *Platforms & Integrations* (GT1), *Agent Orchestration & Invocation Semantics* (GT12), and *Model–Agent–Tool Interaction Contracts* (GT3). These frameworks also display significant activity in *Tracebacks & Unhandled Exceptions* (GT8), *Provider/Model Inconsistencies & Tooling Limits* (GT13), and *Runtime Execution & Operational Failures* (GT11). This broader distribution suggests that multi-agent frameworks encompass both coordination-specific challenges and supporting engineering issues related to dependencies, integration, tool interfaces, provider behavior, and runtime failures. Delegation, shared memory, and inter-Agent communication further amplify failure modes that are difficult to localize and resolve.

**Table 9**
Importance and difficulty of major and cross-cutting AI Agent challenge groups.

| Challenge group | Stack overflow evidence | GitHub evidence | Interpretation |
|---|---|---|---|
| Environment, Platforms, and Dependency Management | ST5 is the most popular Stack Overflow topic (Fused Popularity = 1.30), but has low difficulty (Fused Difficulty = 0.62; median 11.71 h to accepted answer). | GT1 has high volume (8177 issues), a 17.4% open rate, and long activity span (1064.6 h). GT4 has a median close time of 32.4 days, while GT5 closes more quickly (6.0 days). | Highly important, but mixed difficulty. Setup problems are frequent and often resolvable, while platform integration and version-change issues create longer maintenance burdens on GitHub. |
| Retrieval, Embeddings, and Agent Memory | ST2 has above-average popularity (1.02) and high difficulty (1.33), while ST6 has the highest difficulty among Stack Overflow topics (1.66; median 87.44 h). | GT2 has substantial volume (5612 issues) and long activity span (1304.7 h). GT13 is also visible (5747 issues), with a 17.2% open rate and activity span of 893.4 h. | Both important and difficult. Retrieval and memory challenges involve embeddings, vector stores, grounding, provider behavior, and knowledge integration, making them harder to resolve. |
| Orchestration and Execution Control | ST4 has below-average popularity (0.78) but above-average difficulty (1.14), with 88.4% of questions lacking accepted answers. | GT12 has substantial volume (4874 issues), a 16.3% open rate, and a median activity span of 550.6 h. | Less visible during early help-seeking, but difficult and persistent in implementation. Orchestration becomes more important as Agents move toward multi-step, tool-using, and multi-agent workflows. |
| Interaction Contracts Between Models and Tools | ST7 is popular (1.13) but has below-average difficulty (0.78), suggesting that many prompt/output questions are answerable in Q&A settings. | GT3 has a long median close time (50.4 days) and long activity span (1647.4 h). GT10 has high volume (5713 issues) and the highest open rate (44.9%). | Important and difficult at the system level. Simple prompting issues may be resolved on Stack Overflow, but schema enforcement, tool calls, parsing, and policy/template constraints are harder in real repositories. |
| Runtime Reliability and Operational Robustness | ST1 and ST3 have below-average difficulty (0.79 and 0.68), indicating that many individual runtime or evaluation questions are answerable. | GT8 has long activity span (1352.5 h), GT9 has the longest close time (98.4 days) and activity span (2382.4 h), and GT11 has long open issue age (270.1 days). | Operationally important and difficult when failures are tied to frameworks, deployment, exceptions, or long-running execution. GitHub reveals persistence that is less visible in Stack Overflow metrics. |
| Workflow/UI Behavior and Configurability | Not directly represented as a separate Stack Overflow high-level family. | GT6 has the largest volume (9125 issues), but a low open rate (2.7%) and very short median close time (0.4 days). GT7 also closes quickly (1.8 days) and has a low open rate (4.9%). | Highly visible but comparatively easier to resolve. These topics represent frequent, localized maintenance work rather than the deepest technical difficulty. |

**Core Orchestration Frameworks** (e.g., *langchain*, *langgraph*, *semantic-kernel*, and *koog*) constitute the backbone of the ecosystem and account for some of the largest issue volumes. Their primary challenges include *LangChain Runtime & Integration Breakages* (GT9), *Platforms & Integrations* (GT1), *Version-Update Regressions* (GT4), *Agent Orchestration & Invocation Semantics* (GT12), and *Model–Agent–Tool Interaction Contracts* (GT3). These frameworks also report substantial counts for *Embeddings & Retrieval* (GT2), *Dependencies, Installs, Imports, Environments* (GT5), and *Provider/Model Inconsistencies & Tooling Limits* (GT13). This pattern confirms that core orchestration frameworks function as integration hubs, with failures arising not only from orchestration logic but also from their connections to models, tools, retrievers, providers, and downstream applications. These issues often persist, highlighting the fragility introduced by API changes in foundational Agent infrastructure.

**RAG & Knowledge Systems**, led by *ragflow* and *llama-index*, are predominantly characterized by *Policy & Template Enforcement Friction* (GT10) and *Embeddings & Retrieval* (GT2). These systems also exhibit notable concentrations of *Platforms & Integrations* (GT1), *Dependencies, Installs, Imports, Environments* (GT5), *Model–Agent–Tool Interaction Contracts* (GT3), *LangChain Runtime & Integration Breakages* (GT9), *Provider/Model Inconsistencies & Tooling Limits* (GT13), *Tracebacks & Unhandled Exceptions* (GT8), and *Agent Orchestration & Invocation Semantics* (GT12). This distribution demonstrates that RAG and knowledge systems address not only retrieval quality but also integration, schema and template enforcement, provider compatibility, runtime robustness, and orchestration with downstream Agent workflows. This domain highlights that retrieval quality, grounding guarantees, and policy enforcement remain persistent technical barriers for knowledge-intensive Agents.

**Visual Workflow Platforms** represent the largest domain by issue volume, primarily driven by *langgenius/dify*, *langflow*, and *Flowise*. Challenges are predominantly concentrated in *Workflow, UI & API Behavior* (GT6), *Feature Requests & Configurability* (GT7), and *Provider/Model Inconsistencies & Tooling Limits* (GT13). The domain also includes significant volumes of *Platforms & Integrations* (GT1), *Runtime Execution*

*& Operational Failures* (GT11), *Agent Orchestration & Invocation Semantics* (GT12), *Tracebacks & Unhandled Exceptions* (GT8), and *Model–Agent–Tool Interaction Contracts* (GT3). These findings indicate that visual workflow platforms reveal Agent complexity through user-facing abstractions, with users encountering Agent problems as workflow misbehavior, missing configuration options, API issues, provider-specific inconsistencies, and runtime failures. These issues are typically resolved quickly, suggesting that low-code Agent builders generate frequent but localized problems as users construct production workflows using graphical abstractions.

**Agent Apps (End-User)** aggregate failures across the entire stack. Repositories such as *lobe-chat*, *OpenHands*, and *AgentGPT* report issues spanning *Runtime Execution & Operational Failures* (GT11), *Platforms & Integrations* (GT1), *Provider/Model Inconsistencies & Tooling Limits* (GT13), *Tracebacks & Unhandled Exceptions* (GT8), and *Policy & Template Enforcement Friction* (GT10). These repositories also exhibit notable activity in *Model–Agent–Tool Interaction Contracts* (GT3), *Dependencies, Installs, Imports, Environments* (GT5), *Agent Orchestration & Invocation Semantics* (GT12), and *Feature Requests & Configurability* (GT7). This pattern indicates that end-user Agent applications serve as convergence points where upstream problems in providers, tools, orchestration, policy enforcement, dependencies, and runtime execution become apparent to users. This domain illustrates how early design decisions manifest as user-facing problems when Agents are deployed as complete applications.

### 3.3.1. Domain-level difficulty and resolution dynamics

In addition to identifying the topics present within the ecosystem, we further examine how the difficulty of developer challenges varies across different AI Agent repository domains. The analysis compares domains using resolution-oriented indicators, such as open-issue rate, median time-to-close, activity span, and comment volume. These metrics distinguish domains where issues are frequently reported but resolved quickly from those where problems remain open for extended periods, require prolonged discussion, or persist throughout lengthy maintenance cycles.

Table 13 demonstrates systematic differences in the evolution of challenges across Agent ecosystem domains. *UI/Copilot Components* display the highest open-issue rate (52.0%, compared to an overall 12.5%), which reflects the challenge of stabilizing interactive Agent behavior during rapid UI iteration. Their short median time-to-close (14.1 days) suggests that individual issues are not particularly complex, but they accumulate more rapidly than they are resolved. *Core Orchestration Frameworks* exhibit the longest activity spans and time-to-close (2350.5 h and 97.7 days, the highest in both categories), which aligns with the architectural complexity of orchestration logic and the necessity for rigorous regression control. In contrast, *Visual Workflow Platforms* and *RAG & Knowledge Systems* typically resolve issues quickly (median time-to-close of 1.7 and 4.8 days), indicating that many reports are incremental or configuration-related. *Agent Apps (End-User)* combine moderate open-issue rates (14.7%) with extended activity spans (1233.4 h) and the highest engagement (4 comments per issue), suggesting that application-level Agents reveal compounded failures originating from multiple upstream domains.

This comparison indicates that developer difficulty is not distributed uniformly across the AI Agent ecosystem. Domains providing core infrastructure, such as orchestration frameworks and memory or state components, typically involve more complex architectural or state-management challenges that require extended investigation, as evidenced by their longer activity spans. End-user Agent applications also exhibit persistent difficulty due to the integration of multiple upstream components, including models, tools, retrieval systems, UI layers, and execution environments. In contrast, visual workflow platforms and RAG-oriented systems generate a high volume of issues, but their shorter time-to-close suggests that a greater proportion of these reports pertain to configuration, usability, or incremental feature-level fixes.

Overall, Agent difficulty varies systematically with domain role, rather than being uniformly distributed across the ecosystem.

**Table 10**
AI Agent technologies used on Stack Overflow.

| Category | Technologies (share %) |
|---|---|
| Orchestration frameworks | LangChain (70.6), LangGraph (4.7), CrewAI (1.5), AutoGen[a] (2.3), Flowise (0.6), Semantic-Kernel (1.5) |
| Model APIs & runtimes | OpenAI (24.2), Llama models/tooling (5.0), Ollama (3.8), Azure OpenAI (3.8), Google GenAI (2.0), AutoGen[a] (2.3), Groq (0.6), vLLM (0.6) |
| Retrieval & Indexing | ChromaDB (9.6), LlamaIndex (4.1), FAISS (3.8), Pinecone (2.6), Weaviate (0.6) |
| Embedding Models | Hugging Face (5.8), Sentence-Transformers (1.5) |
| Evaluation | RAGAS (0.6) |
| Search services | Azure Cognitive Search (2.0), Elasticsearch/OpenSearch (0.9) |
| Other | Streamlit (2.9), Neo4j (1.2), Anthropic (0.9), Azure AI Inference (0.3) |

[a] Cross-listed: AutoGen appears under both Orchestration and Model APIs (used as an agentic framework and as a model/runtime interface).

**Findings (Domains).** The AI Agent ecosystem is structurally heterogeneous: infrastructure-oriented domains concentrate fewer but harder orchestration and integration challenges, while UI- and workflow-oriented domains generate large volumes of faster-resolving issues. End-user Agent applications amplify challenges across all domains, confirming that Agent-specific difficulty arises from autonomy, persistent state, and cross-system coordination rather than isolated framework defects.

### 3.3.2. Cross-domain recurrence of GitHub topics

In addition to identifying the main topics within each ecosystem domain, Fig. 4 and Table 14 illustrate the recurrence of individual GitHub topics across domains. This analysis distinguishes three patterns. Cross-domain topics are distributed across several repository categories and indicate broader ecosystem-level challenges. Domain-sensitive topics appear in multiple domains but are more prominent in one or two categories. Domain-specific or ecosystem-specific topics are strongly concentrated in a particular domain or framework ecosystem. This distinction is significant because a topic may appear across several domains, yet its practical impact may be driven primarily by a single dominant category.

Several topics demonstrate clear cross-domain recurrence. Platforms & Integrations (GT1) is the most prominent example, accounting for 8177 issues and distributed across multiple domains, with the largest share in Core Orchestration Frameworks (32.1%), followed by Multi-Agent Coordination Frameworks (18.5%) and Agent Apps (16.3%). This distribution indicates that integration is not limited to a single type of Agent technology but is a general challenge across the ecosystem. Similarly, Model–Agent–Tool Interaction Contracts (GT3) spans Core Orchestration Frameworks, Multi-Agent Coordination Frameworks, and RAG & Knowledge Systems, indicating that model–tool interfaces and structured interaction contracts influence several types of Agent systems. Dependencies, Installs, Imports, Environments (GT5), Tracebacks & Unhandled Exceptions (GT8), and Agent Orchestration & Invocation Semantics (GT12) also recur across multiple domains, suggesting that dependency management, exception handling, and orchestration are ecosystem-level concerns rather than isolated framework issues.

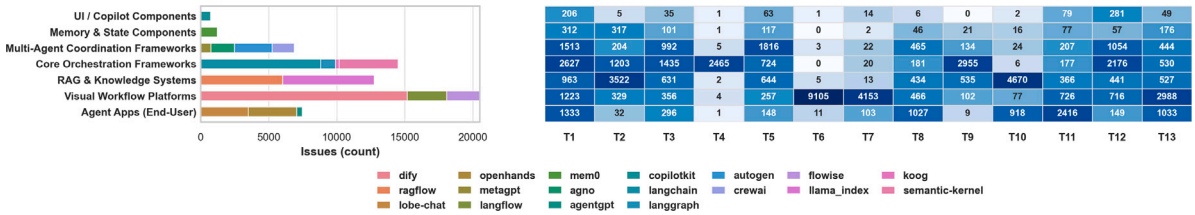



**Fig. 4.** GitHub repo category topic counts.

**Table 11**
GitHub AI agent repositories.

| Repository | Category | Stars | Forks | Open issues | All issues | Prime lang |
|---|---|---|---|---|---|---|
| langgenius/dify | Visual Workflow Platforms | 123,817 | 19,238 | 668 | 15,550 | TS |
| infiniflow/ragflow | RAG & Knowledge Systems | 70,522 | 7671 | 3004 | 6186 | Py |
| lobehub/lobe-chat | Agent Apps (End-User) | 69,535 | 14,288 | 1158 | 4984 | TS |
| OpenHands/OpenHands | Agent Apps (End-User) | 65,958 | 8125 | 209 | 3611 | Py |
| FoundationAgents/MetaGPT | multi-Agent Coordination | 61,866 | 7737 | 61 | 888 | Py |
| langflow-ai/langflow | Visual Workflow Platforms | 142,175 | 8216 | 936 | 3051 | Py |
| mem0ai/mem0 | Memory & State Components | 44,730 | 4864 | 573 | 1295 | Py |
| agno-agi/agno | multi-Agent Coordination | 36,398 | 4818 | 384 | 1768 | Py |
| reworkd/AgentGPT | Agent Apps (End-User) | 35,405 | 9465 | 217 | 469 | TS |
| CopilotKit/CopilotKit | UI/Copilot Components | 27,509 | 3580 | 466 | 775 | TS |
| langchain-ai/langchain | Core Orchestration | 122,819 | 20,254 | 299 | 8927 | Py |
| langchain-ai/langgraph | Core Orchestration | 22,626 | 3977 | 276 | 1107 | Py |
| microsoft/autogen | multi-Agent Coordination | 52,913 | 8040 | 532 | 2910 | Py |
| crewAIInc/crewAI | multi-Agent Coordination | 41,836 | 5594 | 187 | 1651 | Py |
| FlowiseAI/Flowise | Visual Workflow Platforms | 47,600 | 23,439 | 744 | 2442 | TS |
| run-llama/llama_index | RAG & Knowledge Systems | 46,053 | 6664 | 270 | 6833 | Py |
| JetBrains/koog | Core Orchestration | 3570 | 283 | 186 | 279 | Kotlin |
| microsoft/semantic-kernel | Core Orchestration | 26,914 | 4400 | 569 | 4806 | C# |

**Table 12**
Language footprint of the Stack Overflow and GitHub corpora.

| Language | SO share (%) | Repo bytes | Repo share (%) |
|---|---|---|---|
| Python | 75.8 | 109,655,152 | 50.87 |
| TypeScript | 5.5 | 57,855,653 | 26.84 |
| JavaScript | 5.0 | 9,078,924 | 4.21 |
| C# | 1.5 | 17,487,007 | 8.11 |
| Java | 3.2 | – | – |
| SQL | 2.6 | – | – |
| Jupyter Notebook | – | 10,380,564 | 4.82 |
| Kotlin | – | 6,428,889 | 2.98 |
| Others/NA[a] | 12.2 | 4,684,636 | 2.17 |

[a] "Others/NA" aggregates Stack Overflow posts with unspecified language tags and GitHub languages outside the top six. Repo totals sum to 215,570,825 bytes.

**Table 13**
Domain metrics by category in GitHub Issues (medians; d=days, h=hours).

| Cat. | Open% | Close d | Open d | Act h | Com |
|---|---|---|---|---|---|
| UI/Copilot | 52.0 | 14.1 | 166.5 | 477.3 | 3 |
| Memory/State | 28.7 | 13.7 | 177.2 | 279.3 | 2 |
| multi-Agent | 9.4 | 21.0 | 245.5 | 735.2 | 3 |
| Orch. Core | 6.5 | 97.7 | 167.4 | 2350.5 | 2 |
| RAG/Knowl. | 24.8 | 4.8 | 265.6 | 117.5 | 3 |
| Workflow Vis. | 6.9 | 1.7 | 172.3 | 116.6 | 2 |
| Agent Apps | 14.7 | 4.8 | 258.3 | 1233.4 | 4 |
| **Overall** | **12.5** | **9.2** | **218.8** | **484.1** | **2** |

Other topics are more accurately characterized as domain-sensitive. Embeddings & Retrieval (GT2) appears beyond RAG-specific repositories, but 62.8% of its issues are concentrated in RAG & Knowledge Systems, with a secondary presence in Core Orchestration Frameworks (21.4%). This distribution indicates that retrieval is a general capability in Agent systems, although its most significant practical impact remains in knowledge-intensive systems. Runtime Execution & Operational Failures (GT11) appears across several domains, but is most prominent in Agent Apps (59.7%), where failures from providers, tools, orchestration, and deployment become user-facing. Provider/Model Inconsistencies & Tooling Limits (GT13) is also broad, but more visible in Visual Workflow Platforms (52.0%) and Agent Apps (18.0%), where users directly encounter provider differences and model capability gaps.

A third group is strongly domain-specific or ecosystem-specific. Workflow, UI & API Behavior (GT6) is almost entirely concentrated in Visual Workflow Platforms (99.8%), reflecting the central role of graphical workflows, APIs, and user-facing execution paths in tools such as dify, langflow, and Flowise. Feature Requests & Configurability (GT7) follows a similar pattern, with 96.0% of its issues in Visual Workflow Platforms, indicating that low-code Agent builders generate substantial demand for customization and extensibility. Version-Update Regressions (GT4) is concentrated in Core Orchestration Frameworks (99.4%), while LangChain Runtime & Integration Breakages (GT9) is mainly concentrated in the orchestration ecosystem (78.7%). Policy & Template Enforcement Friction (GT10) is concentrated in RAG & Knowledge Systems (81.7%), indicating that template, policy, and grounding constraints are particularly significant in knowledge-intensive Agent workflows.

The broadest ecosystem-level findings are not necessarily the largest topics, but rather those that recur across multiple domains without being dominated by a single category. Integration issues, model–tool contracts, dependency management, exception handling, orchestration, runtime reliability, and provider inconsistency are present across different repository categories and therefore represent broad challenges in AI Agent development. In contrast, workflow and UI behavior,

**Table 14**
Cross-domain recurrence patterns of GitHub topics across AI Agent ecosystem domains.

| Topic | Total | Pattern | Dominant/Top Domains | Interpretation |
|---|---|---|---|---|
| GT1 | 8177 | Cross-domain | Core Orch. (2627; 32.1%), Multi-Agent (1513; 18.5%), Agent Apps (1333; 16.3%) | Platform and integration issues are broadly distributed across the ecosystem. |
| GT2 | 5612 | Domain-sensitive | RAG/Knowl. (3522; 62.8%), Core Orch. (1203; 21.4%), Visual Workflow (329; 5.9%) | Retrieval issues recur across domains but are mainly concentrated in RAG and knowledge systems. |
| GT3 | 3846 | Cross-domain | Core Orch. (1435; 37.3%), Multi-Agent (992; 25.8%), RAG/Knowl. (631; 16.4%) | Model–tool interaction contracts affect several framework and knowledge-system domains. |
| GT4 | 2479 | Domain-specific | Core Orch. (2465; 99.4%) | Version-update regressions are almost entirely concentrated in core orchestration frameworks. |
| GT5 | 3769 | Cross-domain | Multi-Agent (1816; 48.2%), Core Orch. (724; 19.2%), RAG/Knowl. (644; 17.1%) | Dependency and environment issues recur across several framework-oriented domains. |
| GT6 | 9125 | Domain-specific | Visual Workflow (9105; 99.8%) | Workflow, UI, and API behavior is almost entirely concentrated in visual workflow platforms. |
| GT7 | 4327 | Domain-specific | Visual Workflow (4153; 96.0%) | Feature requests and configurability are mainly concentrated in visual workflow platforms. |
| GT8 | 2625 | Cross-domain | Agent Apps (1027; 39.1%), Visual Workflow (466; 17.8%), Multi-Agent (465; 17.7%) | Tracebacks and unhandled exceptions appear across several user-facing and framework domains. |
| GT9 | 3756 | Ecosystem-specific | Core Orch. (2955; 78.7%) | LangChain-related runtime and integration breakages are concentrated in the orchestration ecosystem. |
| GT10 | 5713 | Domain-specific | RAG/Knowl. (4670; 81.7%) | Policy and template enforcement is mainly concentrated in knowledge-intensive systems. |
| GT11 | 4048 | Domain-sensitive | Agent Apps (2416; 59.7%), Visual Workflow (726; 17.9%), RAG/Knowl. (366; 9.0%) | Runtime execution failures appear across domains but are most visible in end-user applications. |
| GT12 | 4874 | Cross-domain | Core Orch. (2176; 44.6%), Multi-Agent (1054; 21.6%), Visual Workflow (716; 14.7%) | Orchestration and invocation issues recur across core, multi-agent, and workflow-oriented domains. |
| GT13 | 5747 | Domain-sensitive | Visual Workflow (2988; 52.0%), Agent Apps (1033; 18.0%), Core Orch. (530; 9.2%) | Provider/model inconsistencies are broad but especially visible in visual and user-facing systems. |

configurability, version regressions, policy and template enforcement, and LangChain-specific breakages are significant developer pain points, but their prominence is largely determined by the design, popularity, and maturity of specific domains within the current public AI Agent ecosystem.

**Findings (Cross-Domain Recurrence).** Cross-domain recurrence helps distinguish ecosystem-level challenges from topics whose volume is concentrated in a particular domain. Integration, interaction contracts, dependency management, orchestration, runtime reliability, and provider inconsistencies recur across multiple repository categories.

## 4. Discussion

This section discusses the broader implications of the findings reported for RQ1–RQ3. In addition to relating the results across the two data sources, we present targeted exploratory analyses that help explain selected patterns observed in the main results. These follow-up analyses do not introduce new topic classifications or address separate research questions; instead, they provide contextual data regarding the evolution, practical implications, and ecosystem dependence of the identified challenges.

### 4.1. Evolution of AI agent challenges

Having established the per-topic prevalence and difficulty of challenges in RQ1 and RQ2, we now interpret how the relative prominence of these topics shifts over time. We view this temporal pattern as a progression from initial adoption barriers to system-level complexity, rather than as a distinct set of new findings. Fig. 6 presents the monthly evolution of Stack Overflow topics, while Fig. 7 illustrates the corresponding trends on GitHub. Considered together, these trajectories indicate that the dominant challenge transitions as the ecosystem evolves from early experimentation with Agent frameworks to more advanced deployment, orchestration, and workflow integration.

In early 2023, Stack Overflow activity increased notably around *Installation & Dependency Conflicts* (ST5), *Document Embeddings & Vector Stores* (ST2), and *RAG Engineering* (ST6). This pattern indicates that developers initially encountered difficulties in assembling and operationalizing emerging Agent stacks. Concurrently, GitHub exhibited heightened activity related to *Runtime Execution & Operational Failures* (GT11) and *Provider/Model Inconsistencies & Tooling Limits* (GT13), which suggests that early Agent frameworks were fragile and highly sensitive to environment configuration, provider behavior, and deployment context.

From late 2023 into early 2024, several foundational challenges began to stabilize. The volume of dependency-related Stack Overflow questions plateaued, and GitHub issues concerning runtime failures

**Table 15**
Repository-level resolution dynamics of feature-request and enhancement issues.

| Repository | Issues | Open | Closed | Open (%) | Days to close | |
|---|---|---|---|---|---|---|
| | | | | | Median | Mean |
| langgenius/dify | 3792 | 146 | 3646 | 3.9 | 8.7 | 27.1 |
| OpenHands/OpenHands | 1053 | 80 | 973 | 7.6 | 38.0 | 51.4 |
| run-llama/llama_index | 938 | 68 | 870 | 7.2 | 97.8 | 74.1 |
| infiniflow/ragflow | 753 | 431 | 322 | 57.2 | 23.5 | 88.6 |
| langflow-ai/langflow | 472 | 107 | 365 | 22.7 | 98.0 | 92.9 |
| FlowiseAI/Flowise | 471 | 280 | 191 | 59.4 | 47.1 | 107.5 |
| lobehub/lobe-chat | 469 | 283 | 186 | 60.3 | 3.1 | 48.7 |
| agno-agi/agno | 377 | 100 | 277 | 26.5 | 16.9 | 30.8 |
| crewAIInc/crewAI | 255 | 9 | 246 | 3.5 | 35.9 | 52.4 |
| CopilotKit/CopilotKit | 187 | 120 | 67 | 64.2 | 18.4 | 64.4 |
| mem0ai/mem0 | 137 | 15 | 122 | 10.9 | 344.4 | 338.6 |
| microsoft/semantic-kernel | 122 | 10 | 112 | 8.2 | 107.4 | 122.1 |
| reworkd/AgentGPT | 116 | 50 | 66 | 43.1 | 13.6 | 28.9 |
| langchain-ai/langchain | 104 | 63 | 41 | 60.6 | 8.7 | 13.6 |
| JetBrains/koog | 61 | 27 | 34 | 44.3 | 20.2 | 46.3 |
| langchain-ai/langgraph | 29 | 18 | 11 | 62.1 | 68.4 | 85.1 |
| FoundationAgents/MetaGPT | 15 | 0 | 15 | 0.0 | 233.3 | 280.4 |

(GT11) peaked before declining. These trends reflect the partial maturation of core frameworks and the adoption of more reliable deployment practices. However, *Platforms & Integrations* (GT1) remained consistently high throughout the observed period, indicating that integrating Agents with external services, SDKs, APIs, and cloud environments continues to present significant challenges, even as internal framework stability improves.

Beginning in mid-2024, GitHub data reveals a pronounced shift toward challenges related to workflow, policy, and user-facing system behavior. As indicated in Table 16, *Workflow, UI & API Behavior* (GT6) emerges as the dominant monthly topic from July 2024 onward. Its prominence increases substantially through 2025, when it becomes the leading monthly topic by a wide margin (Table 16). Simultaneously, *Policy & Template Enforcement Friction* (GT10) becomes a recurring leading topic, particularly from February 2025 onward. These trends suggest that as Agent systems are more widely deployed through workflow platforms and applications, developer challenges increasingly shift from basic setup and execution to workflow composition, API behavior, policy enforcement, and template constraints.

This trend is further illustrated in Fig. 7, where GT6 increases sharply in 2025 and remains the most prominent GitHub topic for much of the year. This pattern aligns with the expansion of Visual Workflow Platforms such as `Dify`, `LangFlow`, and `Flowise`. While these platforms enhance the accessibility of Agent construction, they also introduce new complexities. Users are required to configure workflows, connect tools and providers, manage API behavior, and debug user-facing execution paths. Consequently, the rise of GT6 reflects not only an increase in UI-related issues but also a broader shift in Agent development toward low-code and workflow-based environments.

Stack Overflow primarily captures early-stage learning and configuration challenges, whereas GitHub reflects later-stage operational and ecosystem-level friction. The two sources thus provide complementary perspectives on the evolution of AI Agent challenges. Stack Overflow highlights the difficulties developers encounter when learning and assembling Agent components, while GitHub reveals issues that arise during implementation, integration, maintenance, and use in real repositories. Rather than diminishing over time, the nature of Agent-related challenges shifts upward in the technology stack. As execution stabilizes, challenges increasingly concentrate around orchestration, policy enforcement, provider behavior, and user-facing workflows. This progression demonstrates that AI Agent complexity arises not only from underlying models but also from autonomy, persistent state, tool integration, and large-scale cross-system coordination.

### 4.2. Feature requests & enhancements

While most challenge topics focus on failures, *Feature Requests & Configurability* (GT7), identified in RQ1 as a cross-cutting GitHub topic, highlights a complementary developer challenge: requests for capabilities that current Agent frameworks do not yet provide, expose, or stabilize. Table 15 and Fig. 5 present a repository-level analysis of this demand using feature-request-related GitHub issues. These issues comprise 9351 records, representing 14.6% of our filtered GitHub dataset of 64,098 issues, and are reported with their resolution status and time to close.

GitHub issues include a `labels` field; however, projects may use different label vocabularies. Some use GitHub's default `enhancement` label, some define explicit `feature request`-style labels, and others use custom scope labels such as `feature:/feat:`. We used these three feature-related label families to extract feature requests and enhancements. Issues explicitly marked as bugs — either through a bug-type label or through a `[Bug]`-prefixed title generated by the repository's issue template — were excluded. *microsoft/autogen* labels issues only by triage state and project area and is therefore absent.

This indicates that a substantial part of repository activity involves not only defect repair but also discussion of what Agent frameworks should support as they mature.

The distribution in Table 15 indicates that demand for new features is particularly prominent in repositories that expose Agent functionality through workflows, APIs, retrieval pipelines, model-provider integrations, user interfaces, and deployment settings. This pattern is expected, as frameworks that are more accessible to end users and application developers tend to receive more requests for concrete and configurable behaviors. For instance, requests in *dify* include downloadable file outputs from custom tools or nodes (dify I19826) and support for downstream nodes consuming structured LLM array outputs (dify I22965). *Flowise* users request workflow export to Python (Flowise I275), and *CopilotKit* users seek finer control over chat-window behavior (CopilotKit I2892). Comparable expectations are also observed in smaller or more specialized repositories, such as exporting AgentGPT tasks in a tree view (AgentGPT I82) or extending MetaGPT usage to additional tools and model interfaces (MetaGPT I1240).

These requests demonstrate that feature demand frequently overlaps with the same technical areas identified as challenging in failure-oriented topics. Developers request additional model providers (dify I1508; OpenHandsI6468), support for open or private LLMs (agno I121; AgentGPT I322), emerging agent protocols such as Model Context Protocol support (dify I11415; crewAI I1813), and additional vector-store

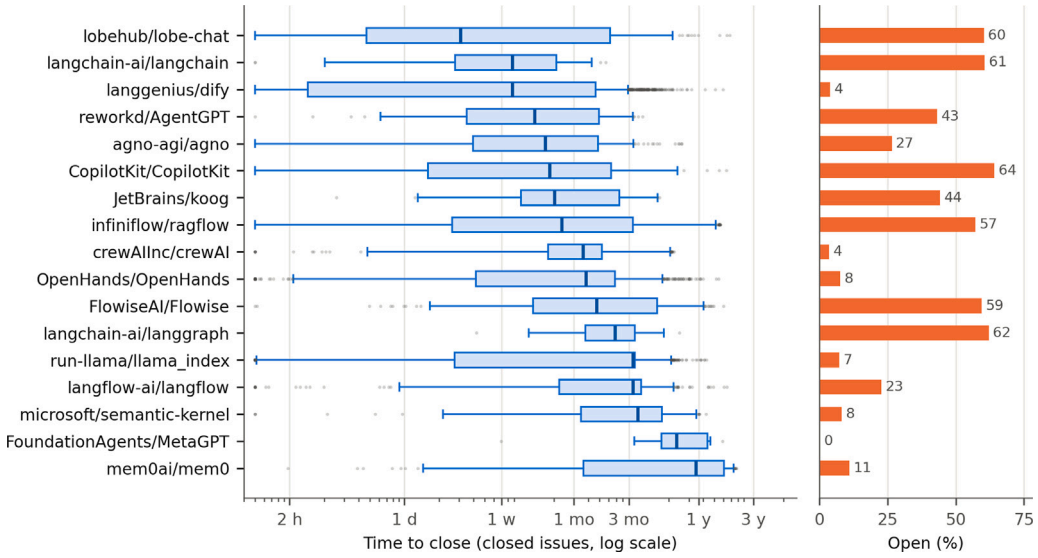



**Fig. 5. Resolution dynamics of feature-request and enhancement issues from** Table 15. The middle panel reports time to close for closed issues on a logarithmic scale, with boxes showing the interquartile range, whiskers the 5th–95th percentiles, and vertical lines the medians. The right panel reports the share of issues that remain open per repository. Repositories are ordered by median time to close.

integrations (langchain I32984; mem0 I1166). Thus, the integration challenges observed as failures in GT1 and GT2 are also reflected in requests for expanded ecosystem coverage.

A comparable pattern is evident in the areas of retrieval, memory, and context management. Requests for GraphRAG support (dify I6019; Flowise I2837), multiple embeddings per node (llama_index I10486), token-aware context compaction (agno I4952), chat-history size control (semantic-kernel I6155), and Ollama context-window configuration (koog I295) indicate that developers seek not only to overcome failures in retrieval and memory, but also to improve the configurability, observability, and adaptability of these mechanisms for diverse application contexts.

A similar interpretation extends to execution, deployment, and operational readiness. Requests for standardized parameter handling (CopilotKit I2896), graceful handling of recursion-limit termination (langchain I33653), support for `uv` in `langgraph dev` (langgraph I5463), streaming tool-call support (koog I513), and per-span observability attributes in Agent DSLs (koog I819) demonstrate that developers expect Agent frameworks to provide more predictable execution and monitoring. Enterprise-oriented requests, including single sign-on (langflow I8374), Active Directory authentication (ragflow I6111), API-key expiration settings (dify I20271), deployment configuration persistence across upgrades (dify I2804), and organization-scale knowledge-base management (lobe-chat I4005), further suggest that Agent frameworks are increasingly expected to support production and organizational environments.

The resolution patterns in Table 15 indicate that this demand is handled unevenly across repositories. Some projects address feature-related requests promptly and maintain a small open backlog. For instance, *dify* has the largest number of feature-related issues, yet only 3.9% remain open, with a median close time of 8.7 days. In contrast, other repositories, such as *CopilotKit*, *langgraph*, *langchain*, *lobe-chat*, *Flowise*, and *ragflow*, have more than half of these issues still open. This should not be interpreted solely as a lack of responsiveness, since feature requests often depend on roadmap priorities, architectural redesign, compatibility constraints, or maintainer triage policies. Rather, it suggests that many requested capabilities represent longer-term design and product decisions rather than minor fixes.

GT7 complements the failure-focused topics by identifying areas where current AI Agent frameworks do not yet meet developer expectations. Bug reports and runtime failures show where systems fail or behave unexpectedly, whereas feature-related and configurability-related issues highlight areas where developers seek ecosystem expansion. Accordingly, GT7 serves as a cross-cutting indicator of ecosystem maturation: as Agent frameworks transition from experimental stages to practical adoption, developer focus shifts from basic functionality to extensibility, configurability, integration, observability, deployment, and operational readiness in real-world contexts.

### 4.3. Technologies in practice

Tables 10 and 11 summarize the AI Agent technologies observed in our Stack Overflow and GitHub datasets. On GitHub, a small set of repositories accounts for most stars and issue activity in our sample. *Langflow*, *Dify*, and *LangChain* each exceed 120 K stars and exhibit substantial issue volumes, with more than 3000 total issues per repository. These projects primarily target agent orchestration, workflow construction, and system integration, indicating that coordination and tooling remain central concerns in agent development.

On Stack Overflow, two frameworks (`LangChain` and `LangGraph`) account for more than 75% of Agent-related posts. As shown in Table 10, `LangChain` alone accounts for 70.6% of Agent-related posts, while `LangGraph` contributes an additional 4.7%. Other orchestration frameworks, such as `CrewAI` and `AutoGen`, appear much less frequently, each representing fewer than 3% of posts. A similar distribution is observed for supporting technologies: OpenAI-related APIs appear in 24.2% of posts, while retrieval and indexing tools such as `ChromaDB` (9.6%) and `LlamaIndex` (4.1%) are discussed at lower but non-trivial rates.

Although we identified a broader set of Agent technologies on GitHub, many of these projects lack stable Stack Overflow tags or sufficient post volume to support reliable analysis. This limits cross-platform comparison and likely contributes to the dominance of a small number of frameworks in Stack Overflow discussions. The difference also reflects a language bias: Stack Overflow Agent discussions are predominantly Python-centric, whereas GitHub repositories span a wider range of languages, including TypeScript, Kotlin, and C# (see Table 12).f

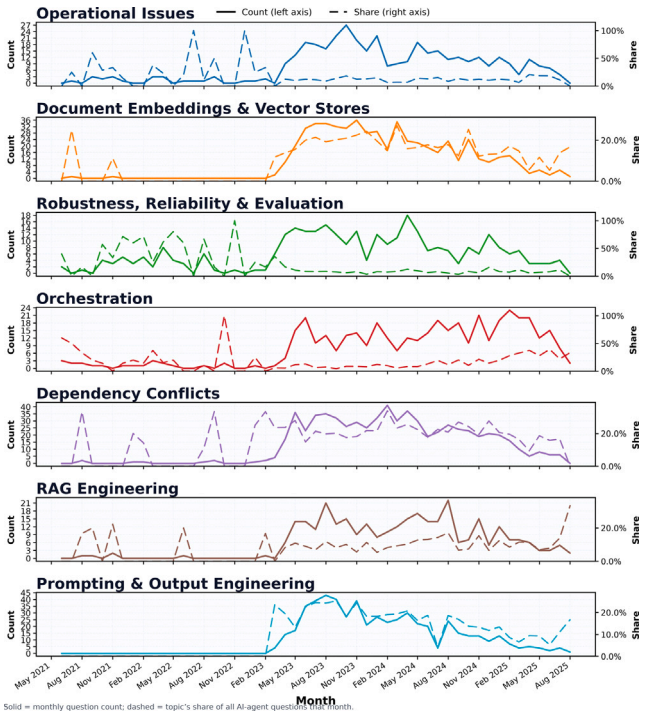



**Fig. 6.** Stack Overflow AI Agent Topics Evolution.

**Table 16**
Top GitHub topics by monthly issue count from July 2024 to December 2025.

| Month | Top 1 | Top 2 | Top 3 |
|---|---|---|---|
| 2024–07 | GT6 (269) | GT1 (248) | GT13 (245) |
| 2024–08 | GT6 (253) | GT13 (242) | GT1 (205) |
| 2024–09 | GT6 (329) | GT13 (216) | GT2 (187) |
| 2024–10 | GT6 (344) | GT1 (295) | GT7 (215) |
| 2024–11 | GT6 (303) | GT1 (282) | GT13 (213) |
| 2024–12 | GT6 (272) | GT1 (241) | GT10 (204) |
| 2025–01 | GT1 (270) | GT6 (262) | GT13 (256) |
| 2025–02 | GT6 (528) | GT10 (389) | GT13 (318) |
| 2025–03 | GT6 (1214) | GT10 (647) | GT7 (363) |
| 2025–04 | GT6 (816) | GT10 (468) | GT1 (280) |
| 2025–05 | GT6 (491) | GT1 (289) | GT10 (267) |
| 2025–06 | GT6 (407) | GT1 (300) | GT10 (276) |
| 2025–07 | GT6 (497) | GT1 (273) | GT10 (263) |
| 2025–08 | GT6 (413) | GT10 (326) | GT1 (318) |
| 2025–09 | GT6 (524) | GT10 (249) | GT1 (247) |
| 2025–10 | GT6 (444) | GT10 (264) | GT1 (229) |
| 2025–11 | GT6 (400) | GT10 (362) | GT1 (265) |
| 2025–12 | GT6 (384) | GT10 (321) | GT1 (244) |

### 4.4. Comparing to other fields

Table 17 compares AI Agent questions with five previously studied Stack Overflow domains using common popularity and difficulty metrics. AI Agent questions receive an average of 2240 views, which is comparable to Mobile (2300) and close to Security (2461). In contrast, their average score (1.16) is lower than Mobile (2.1) and Security (2.7), and their average answer count (0.93) is the lowest among domains where this metric is reported.

Differences are more pronounced for difficulty-related measures. A large majority of AI Agent questions (82.6%) do not have an accepted answer, compared to 67.7% for Chatbot questions and 52.0% and 48.2% for Mobile and Security, respectively. Even domains previously characterized as challenging, such as Big Data (60.3%) and IoT (64.0%), exhibit lower rates of unresolved questions.

Time to resolution further distinguishes AI Agent questions. The median time to an accepted answer is 23.4 h, compared to 14.8 h for Chatbot questions and less than one hour for Mobile (0.7 h) and Security (0.9 h). Big Data and IoT questions are typically resolved within a few hours (3.3 and 2.9 h, respectively), still substantially faster than AI Agent questions.

Overall, these results show that AI Agent questions attract levels of attention similar to established domains but are resolved less frequently and more slowly. This combination suggests that, relative to prior fields studied on Stack Overflow, AI Agent-related problems are harder for the community to answer in a timely and definitive manner.

### 4.5. Influence of ecosystem bias on observed challenges

A persistent challenge in mining-based software engineering research is that the observed issues are shaped by the communities, tools, and technologies represented in the data. This concern is relevant in our study because the Stack Overflow corpus is strongly skewed toward Python and LangChain-related discussions. As shown in Table 12, Python accounts for 75.8% of the Stack Overflow language footprint, while the GitHub repositories show a broader distribution, with Python representing 50.87% of repository bytes and TypeScript representing 26.84%. This indicates that Stack Overflow provides a valuable but ecosystem-biased view of AI Agent development.

Such imbalances are common in empirical studies of software ecosystems. For example, Asgari et al. (2026) found that research on testing deep code models is mainly concentrated on Java, followed by Python and C/C++, while B. Zhang et al. (2023) observed that practitioners using GitHub Copilot are more active in JavaScript/TypeScript and Python ecosystems. These examples show that research and practice may emphasize different languages and environments, which supports our decision to use GitHub Issues as a complementary source rather than relying only on Stack Overflow.

The GitHub Issues corpus helps reduce this bias by covering multiple repositories, framework categories, and programming languages. It also exposes challenges that are less visible in Stack Overflow discussions, such as workflow behavior, API usage, feature requests, configurability, and repository-specific runtime failures. For example, Stack Overflow mainly highlights installation and dependency conflicts (ST5), document embeddings and vector stores (ST2), prompt/output engineering (ST7), and orchestration questions (ST4). In contrast, GitHub Issues reveals implementation and maintenance concerns such as platforms and integrations (GT1), workflow, UI, and API behavior (GT6), feature requests and configurability (GT7), runtime execution failures (GT11), and provider/model inconsistencies (GT13).

This bias can affect the interpretation of our findings. Challenge families that appear in both Stack Overflow and GitHub Issues — such as retrieval and memory pipelines, orchestration and execution control, model–tool interaction contracts, and runtime reliability — provide stronger evidence for our broader conclusion that AI Agent development is increasingly a systems engineering problem. In contrast, the exact ranking and relative weight of some topics should be interpreted with more caution. For instance, Table 11 shows that langgenius/dify alone contributes 15,550 issues and belongs to the Visual Workflow Platforms category. This helps explain why workflow, UI, and API behavior (GT6) is highly visible in Fig. 4. Similarly, RAG-related issues are shaped by repositories such as infiniflow/ragflow and run-llama/llama_index, while orchestration and framework-specific runtime issues are influenced by large framework repositories such as langchain-ai/langchain, microsoft/semantic-kernel, microsoft/autogen, and crewAIInc/crewAI.

Therefore, our high-level conclusions are strongest when the same challenge family appears across both data sources, whereas the precise ordering of topics should be read as a snapshot of the current public AI Agent ecosystem. In particular, setup, dependency, workflow UI, and framework-specific runtime issues reflect real developer pain points, but their prominence is also shaped by repository popularity, framework maturity, and the visibility of issues in public communities.

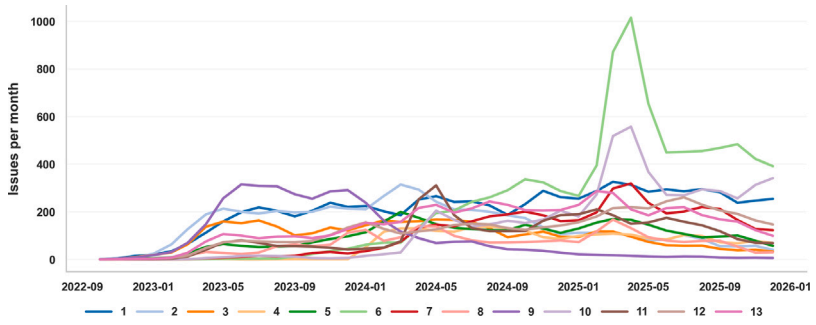



**Fig. 7.** GitHub Issues topics overtime.

**Table 17**
AI Agents vs. five prior Stack Overflow fields.

| Metric | AI Agents | Chatbot | Mobile | Security | Big Data | IoT |
|---|---|---|---|---|---|---|
| **(Popularity)** | | | | | | |
| Avg. View | 2240.1 | 512.4 | 2300.0 | 2461.1 | 1560.4 | 1320.3 |
| Avg. Score | 1.16 | 0.7 | 2.1 | 2.7 | 1.4 | 0.8 |
| Avg. AnswerCount | 0.93 | 1.0 | 1.5 | 1.6 | 1.1 | – |
| **(Difficulty)** | | | | | | |
| % w/o Accepted Ans. | 82.6% | 67.7% | 52.0% | 48.2% | 60.3% | 64.0% |
| Median Hrs. to Accepted | 23.4 | 14.8 | 0.7 | 0.9 | 3.3 | 2.9 |

Notes: Prior-domain values reported from earlier SO studies (Ahmed and Bagherzadeh, 2018; Rosen and Shihab, 2016; Yang et al., 2016; Bagherzadeh and Khatchadourian, 2019; Abdellatif et al., 2020).

## 5. Related work

In this study, we use the term *AI Agent Systems* to refer to software systems that integrate one or more AI Agents capable of reasoning over tasks, interacting with external tools or environments, maintaining context or memory, and executing actions through workflows, APIs, or other software components. Recent work describes LLM-based Agents as systems that extend standalone LLMs with capabilities such as tool use, planning, perception of external resources, and autonomous or semi-autonomous action (Wang et al., 2024; Liu et al., 2024; Guo et al., 2025; Plaat et al., 2025). Accordingly, our scope includes end-user agent applications, agent development platforms, orchestration frameworks, multi-Agent coordination frameworks, RAG and memory systems, and supporting UI/API components for building, deploying, or maintaining AI Agent Systems.

**AI Agents**. Cemri et al. (2025) derive a data-driven taxonomy of multi-Agent LLM failures — *specification*, *inter-Agent misalignment*, *task verification* — aligned to pre /execution/post stages via manual coding of 200+ end-to-end traces. Different studies focus on distinct AI Agent challenges, including Security (Deng et al., 2025; Zou et al., 2025; de Witt, 2025), Ethics & governance (Gabriel et al., 2025; Kolt, 2025), Foundations & effectiveness of multi-Agent AI systems (MAS) (Tian et al., 2025; Du and Ding, 2021; Du et al., 2025; Sapkota et al., 2025), and Design & applications (Mo et al., 2025; Shetty et al., 2024; Cemri et al., 2025). And also Schneider (2025) in a conceptual survey clarify how Agentic AI extends GenAI (reasoning, tools, memory, interaction), formalize agent specification, and catalog open challenges.

*Security*. Deng et al. (2025) survey the agent threat surface (intra-execution, agent–env, agent–agent, memory) and defenses (debate, RAG, constraints, post-correction, tool-use auditing). Zou et al. (2025) show near-universal, transferable prompt-injection/policy-violation attacks in large-scale red teaming, exposing defense gaps. de Witt (2025) argue for *multi-Agent security* as a distinct field, outlining open problems (collusion, attribution, cascade dynamics) and a research agenda. *Ethics & governance*. Gabriel et al. (2025) call for new ethics for autonomous agents—safety, alignment, and social-coordination risks as autonomy grows. Kolt (2025) frame governance as a principal–agent problem and advocate transparency, incentive design, monitoring, and accountability across developers/deployers/users. *Foundations & effectiveness of MAS*. Tian et al. (2025) formalize multi-Agent AI, showing gains from task reallocation and diverse agents, but error amplification under overlap/misalignment; emphasize feedback that reshapes topology. Du and Ding (2021) survey multi-Agent deep reinforcement learning, stressing non-stationarity, partial observability, scalability, and credit assignment. Du et al. (2025) review agent–agent communication, distilling five pillars (scalability, security, real-time, performance, manageability) and standardization paths. *Design & applications*. Mo et al. (2025) study an interactive refactoring agent, reporting decision strategies and collaboration principles. Shetty et al. (2024) sketch requirements and a prototype (AIOpsLab) for a standardized framework to build/test/compare cloud-ops agents.

**Mining Stack Overflow & GitHub Issues with Topic Modeling**. Mining developer discussions has long been used to understand software engineering challenges. Prior studies have applied topic modeling to Stack Overflow and other developer artifacts to characterize developer concerns across domains such as mobile development (Rosen and Shihab, 2016), IoT (Uddin et al., 2021), chatbot development (Abdellatif et al., 2020), release engineering (Openja et al., 2020), Docker (Haque et al., 2020), AutoML (Wang et al., 2023), explainable AI (Sayyadnejad et al., 2025), and low-code development (Elshan et al., 2024). Similar approaches have also been applied to GitHub issues, for example to study quantum software engineering (Li et al., 2021) and ChatGPT-related repositories (Lin et al., 2025). These studies demonstrate that combining topic modeling with manual qualitative analysis is an effective way to construct taxonomies of developer challenges (Barua et al., 2014; Chen et al., 2020). Accordingly, our study follows this established mixed-method approach by combining LDA-MALLET with targeted manual analysis to refine, validate, and organize the discovered topics into higher-level challenge families (Abdellatif et al., 2020; Li et al., 2021; Sayyadnejad et al., 2025; Uddin et al., 2021; Mahmood et al., 2023).

**Contributions Beyond Existing AI Agent Studies.** Recent research has begun to characterize the challenges involved in building AI Agent systems. Cemri et al. (2025) introduce MAST, a taxonomy of multi-agent failure modes derived from execution traces and organized around system-design failures, inter-agent misalignment, and

verification-related failures. Zhu et al. (2026) manually analyze bug reports from LangChain and CrewAI, identifying root causes such as API misuse, API incompatibility, and documentation desynchronization. Other studies survey agent architectures, methodologies, and applications (Guo et al., 2024; Luo et al., 2025), agent operations and reliability concerns (Wang et al., 2025; Tian et al., 2025), and broader conceptual, trustworthiness, and societal aspects of agentic systems (Sapkota et al., 2026; Raheem and Hossain, 2025). These studies provide valuable complementary perspectives, but their findings are shaped by their sources of evidence: execution-trace studies emphasize benchmarked runtime behavior, bug-report studies focus on confirmed framework defects, and surveys synthesize the research literature rather than the practical concerns raised by developers.

Compared with the existing literature, this study makes four complementary contributions. First, unlike previous studies that analyze either Stack Overflow or GitHub issues in isolation, we jointly analyze both sources, treating them as complementary indicators of developer difficulty. Stack Overflow reflects help-seeking challenges, such as installation problems, dependency conflicts, document embeddings, vector stores, and prompt or output engineering. In contrast, GitHub issues capture maintenance challenges, including workflow or API behavior, runtime execution failures, version regressions, and operational disruptions. This distinction highlights an asymmetry between build-time and deployment-time challenges that is not observable from either source alone. Second, the study adopts an ecosystem-level scope by analyzing eighteen repositories across seven functional domains, in addition to Stack Overflow discussions, rather than focusing on one or two specific frameworks. Third, we characterize developer challenges more broadly than confirmed bugs by also considering usage, configuration, conceptual, and feature-level difficulties, including RAG engineering, memory management, and configurability requests. Fourth, we combine topic modeling with manual validation to identify recurring challenge structures at scale, rather than relying exclusively on predefined manually constructed taxonomies.

Despite differences in data sources and methodologies, the findings of this study converge with prior work on several key points. Dependency, version, and API-evolution issues are prominent in the Stack Overflow data, aligning with Zhu et al.'s observation that API misuse, API incompatibility, and documentation desynchronization are major sources of agent-framework bugs. Similarly, the identified topics of interaction contracts and policy enforcement correspond to failure modes reported in execution-trace studies, where agent behavior may fail due to weak specifications, tool-use mismatches, or verification failures. The dual-source developer-discourse approach also reveals challenges that are less likely to be captured by trace- and bug-based studies, particularly in areas such as retrieval and memory engineering, prompt or output usage, configuration requirements, and workflow-level usability. It is important to note that the taxonomy developed here focuses on engineering challenges discussed in public developer forums and does not address adversarial security in depth, which is examined by a separate body of research on red-teaming and attack benchmarks (Zou et al., 2025). Collectively, the convergence across execution traces, framework bug reports, surveys, and large-scale developer discourse suggests that AI Agent development is transitioning from a model-centric phase to a systems-engineering phase, where the primary challenges stem from integration, orchestration, interaction contracts, retrieval, memory, and state management rather than solely from model capability.

## 6. Implications and recommendations

We organize the implications into two complementary categories. First, we present five engineering implications that emerge from the identified challenge patterns. These implications concern the design, implementation, deployment, and evolution of AI Agent Systems and are broadly relevant to stakeholders across the AI Agent ecosystem. Second, we discuss stakeholder-specific implications, explaining how practitioners, educators, and researchers can apply these engineering insights within their respective contexts.

**Engineering Implications**. Based on our empirical findings, we derive the following engineering implications for the design and evolution of AI Agent Systems:

*(1) Treat orchestration as a first-class architectural concern.* Across both Stack Overflow (ST4) and GitHub (GT12), orchestration-related challenges are consistently among the most difficult, despite relatively lower visibility on Q&A platforms. This aligns strongly with *Planning* and *multi-Agent* design patterns, where Agents explicitly decompose goals into sub-tasks, route control between steps or sub-Agents, and coordinate execution over time rather than issuing a single model call. Failures in these patterns typically arise from task decomposition, routing, and execution order rather than individual model outputs. Frameworks should therefore expose orchestration primitives explicitly (i.e., state transitions, retries, and termination conditions) rather than embedding them implicitly inside chains or graphs. Clear execution semantics and debuggable control flow are essential for scaling beyond single-step Agents.

*(2) Stabilize interaction contracts between models, tools, and memory*. A large fraction of unresolved issues concentrate around structured inputs and outputs (ST7, GT3, GT10), reflecting fragility in prompt schemas, tool-calling formats, and validation rules. These challenges directly affect *ReAct* and *Tool Use* patterns, in which an Agent alternates between reasoning steps and external actions (e.g., API calls, database queries), and where even minor contract violations can halt entire execution loops. The rise of Model Context Protocol (MCP)–style abstractions is a promising direction: explicit, versioned contracts for context, tools, and memory can reduce silent failures and make Agent behavior more predictable across providers and environments.

*(3) Design memory and retrieval as evolving systems, not static components.* Retrieval- and memory-related topics (ST2, ST6, GT2, GT13) are consistently among the hardest to resolve and show long-lived issue activity. This reflects the mismatch between static RAG pipelines and the needs of long-running, stateful Agents. Patterns such as *Reflection* and *Planning* rely on the ability of Agents to store intermediate results, revisit prior decisions, and update internal state over time. Current tooling does not consistently support these capabilities. We therefore recommend treating Agent memory as a lifecycle-managed subsystem, with explicit policies for updating, pruning, and validating state, rather than as a passive vector store queried once per task.

*(4) Expect UI and workflow-centric Agents to dominate future development effort.* Over time, GitHub activity increasingly concentrates in *Workflow, UI & API Behavior* (GT6), particularly within visual workflow platforms such as Dify, Langflow, and Flowise. These systems operationalize Agentic design patterns by allowing users to compose planning, tool use, and multi-step execution through graphical or declarative workflows, often to build other Agents or application-level pipelines. While many UI-related issues resolve quickly, their sheer volume indicates that the next phase of Agent adoption is driven by end-user composition rather than low-level model access. This suggests that Agent frameworks must prioritize explainability, safe defaults, and recoverability at the workflow level.

*(5) Application-level Agents amplify upstream design decisions.* Agent applications aggregate failures from across the stack, including runtime execution (GT11), platform integrations (GT1), provider inconsistencies (GT13), and unhandled exceptions (GT8). In design-pattern terms, application-level Agents frequently combine *Tool Use*, *Planning*, and *multi-Agent* behaviors into a single system, making them especially sensitive to upstream instability. Our results indicate that robustness at the application layer depends less on adding local safeguards and more on improving guarantees in orchestration, interaction contracts, and memory management below.

The engineering implications discussed above provide a common set of technical recommendations that are broadly relevant to stakeholders

across the AI Agent ecosystem. The following paragraphs highlight additional implications that are specific to practitioners, educators, and researchers.

**Implications for Practitioners.** For practitioners building and deploying AI Agents, our findings emphasize that most failures originate at system boundaries rather than within individual model calls. Issues related to orchestration, runtime execution, and workflow behavior dominate GitHub activity, especially in application-level Agents and visual workflow platforms. Practitioners should therefore prioritize explicit control flow, strong observability, and defensive execution strategies over increasing Agent autonomy. In practice, this means treating planning loops, retries, and tool invocation as auditable system components, and investing early in workflow-level safeguards when deploying user-facing or application-based Agents.

**Implications for Educators.** For educators, the results reveal a gap between what learners most frequently encounter and what they find hardest to master. While installation, prompting, and basic RAG pipelines attract the most attention, the most difficult challenges involve orchestration, persistent memory, and long-running execution. Educational material should therefore move beyond linear prompt–response examples and instead teach Agent behavior over time, including failure modes, execution traces, and debugging practices across Agentic design patterns such as ReAct, Planning, and multi-Agent coordination.

**Implications for Researchers.** For researchers, the findings suggest that progress in AI Agents depends less on improving standalone model capabilities and more on formalizing system-level abstractions. Persistent difficulty in orchestration, retrieval, and interaction contracts indicates a need for research on execution semantics, memory lifecycle management, and standardized interfaces between models and tools. Emerging efforts such as Model Context Protocols point toward promising directions, but our data shows that these abstractions must be evaluated under long-running, user-facing conditions. Future research should therefore prioritize reproducible Agent failures, coordination mechanisms, and cross-component guarantees rather than isolated reasoning benchmarks.

**Overall Implication.** Our findings indicate that AI Agent development is transitioning from a model-centric phase to a systems-engineering phase. The dominant challenges now arise from autonomy, persistent state, and cross-component coordination rather than prompt quality alone. Agentic design patterns remain useful abstractions, but their practical success increasingly depends on explicit orchestration semantics, stable interaction protocols, and tooling that supports long-running, user-facing workflows.

## 7. Threats to validity

**Selection of data source & Bias**. Following previous work (Haque et al., 2020; Alshangiti et al., 2019; Ahmed and Bagherzadeh, 2018; Yang et al., 2023; Elshan et al., 2024; Ahmed et al., 2025; Bi et al., 2021; Rosen and Shihab, 2016), we selected Stack Overflow as one of our primary data sources. This selection introduces a potential threat to validity because Stack Overflow may not encompass all developer questions and may exhibit bias toward specific technologies, particularly LangChain and Python. As a result, insights from alternative communities and development venues may be underrepresented. Nevertheless, Stack Overflow is among the largest and most widely used developer communities, engaging both practitioners and experts, and thus serves as a valuable resource for understanding real-world challenges. To address this limitation, we also analyzed GitHub Issues (Braiek et al., 2018; Huq et al., 2020; Lin et al., 2025; Silva et al., 2021), which offer a complementary perspective on practical development challenges across a broader spectrum of agent frameworks, tools, and programming languages. This expanded coverage helps to reduce ecosystem-specific bias. Additionally, several challenges identified on Stack Overflow were also partially present in GitHub Issues, which increases confidence in the consistency of our findings.

**Selection of tags.** Relying on tags to collect Stack Overflow questions and answers is a validity threat, as tagging can be incomplete, and some relevant posts may be missed. To reduce this risk, we followed well-established techniques (Ahmed and Bagherzadeh, 2018; Rosen and Shihab, 2016; Yang et al., 2016; Haque et al., 2020). The construction of the tag set may introduce bias, as some AI Agent–related questions may not use the selected tags. Tag selection and validation were performed by the first two authors based on manual inspection of sampled posts. We reduced this risk by applying tag expansion heuristics followed by manual inspection of sampled posts during tag selection.

A primary threat to validity concerns the potential for false positives, false negatives, and the limited representativeness of public developer discussions. On Stack Overflow, the term *agent* may refer to concepts unrelated to AI Agent systems, including agent-based simulation, CI/CD execution agents, reinforcement-learning agents, or robotics and control systems. To reduce this risk, we applied a multi-stage filtering process. First, we used tag-based retrieval with the Tag Relevance Threshold (TRT) and Tag Significance Threshold (TST), which are heuristics widely adopted in Stack Overflow mining studies (Rosen and Shihab, 2016; Abdellatif et al., 2020; Uddin et al., 2021; Yang et al., 2023). We then manually inspected the retrieved posts and iteratively refined the exclusion criteria.

Nevertheless, no filtering strategy can fully eliminate false positives or false negatives in large-scale mining studies. Following common practice in empirical software engineering, we therefore treat filtering as a multi-stage reduction process rather than as a guarantee of complete removal (Abdellatif et al., 2020; Bagherzadeh and Khatchadourian, 2019; Haque et al., 2020; Sayyadnejad et al., 2025). Consequently, the dataset should be regarded as a carefully filtered approximation of AI Agent-related developer discussions. Since our analysis focuses on corpus-level challenges, a small residual fraction of off-topic posts is less likely to dominate the topic structure identified by LDA, which captures recurrent word co-occurrence patterns across the corpus (Barua et al., 2014). In addition, any clearly off-topic topic emerging from the model could be identified and excluded during manual labeling. As an additional safeguard, stratified sampling was employed: records were sampled by topic size, and records within each topic were selected by considering the document's topic contribution score ($Topic_Perc_Contrib$), rather than relying solely on random sampling.

For GitHub, false positives are less likely because issues were mined from repositories explicitly identified as AI Agent repositories, rather than through keyword searches over arbitrary issue text. Nevertheless, given the dataset's scale (64,098 issues), exhaustive manual validation was infeasible, and some non-relevant issues may remain. Finally, public Stack Overflow posts and GitHub issues may underrepresent proprietary, enterprise-scale, or closed-source AI Agent deployments, where challenges related to security, scalability, monitoring, compliance, and production operations may be more prominent. This limitation may affect the generalizability of the findings.

**GitHub Issue Metrics and Repository-Size Bias.** Git-Hub issue-based difficulty metrics can be influenced by repository-specific practices, maintainer priorities, project maturity, and community size. For instance, larger repositories often receive a greater number of issues and may triage them differently compared to smaller repositories. To address this limitation, raw issue volume is not interpreted as a direct measure of difficulty. Instead, issue volume is considered a proxy for importance or maintenance load, while resolution-oriented indicators such as open-issue rate, median time-to-close, median age of open issues, and activity span are used as proxies for difficulty. A repository-normalized sensitivity analysis, in which each repository is given equal weight, was also conducted to determine whether topic patterns are primarily influenced by large repositories. However, these

metrics should be regarded as empirical proxies for persistence and maintenance burden rather than as absolute measures of technical complexity.

A potential threat to validity also arises from comparing difficulty across Stack Overflow and GitHub, as the two platforms capture distinct forms of difficulty. Stack Overflow reflects help-seeking difficulty, measured by unresolved questions and time to accepted answers. In contrast, GitHub reflects maintenance difficulty, indicated by open-issue rates, time-to-close, open issue age, and activity span. Consequently, we do not combine these metrics into a single cross-platform difficulty score. Instead, we interpret them as complementary indicators, presented side by side. Therefore, our synthesis should be regarded as an empirical interpretation of recurring difficulty signals across platforms, rather than a direct one-to-one comparison between Stack Overflow questions and GitHub issues.

**Construction of topic taxonomy**. We adopted LDA-MALLET to identify recurring themes in the Stack Overflow and GitHub corpora. Although more recent embedding-based and LLM-based topic modeling methods have been proposed, LDA remains widely used in empirical software engineering research on developer-generated text (Jin et al., 2026; Silva et al., 2021; Bajaj et al., 2014; Barua et al., 2014; Yang et al., 2016; Rosen and Shihab, 2016; Sayyadnejad et al., 2025). This choice is appropriate for our study because our goal is to identify broad and interpretable developer challenges, rather than to optimize predictive performance. LDA also supports manual validation because it provides explicit distributions of words over topics and topics over documents, making the resulting topics easier to inspect, interpret, and compare with prior studies (Jin et al., 2026; Chen et al., 2026). In addition, LDA-MALLET is a mature and reproducible method that has been widely adopted in software engineering studies of Stack Overflow, GitHub issues, and other developer artifacts (Barua et al., 2014; Ahmed and Bagherzadeh, 2018; Haque et al., 2020; Abdellatif et al., 2020; Alanazi and Alfayez, 2024).

We acknowledge that embedding-based and LLM-based approaches, such as BERTopic, have gained increasing attention (Azher et al., 2024; Bu et al., 2023; Naghshzan and Ratte, 2023; Ahmed et al., 2025). These methods can capture richer semantic relationships through transformer embeddings. However, prior studies have reported mixed results when applying such methods to software engineering text, particularly in the presence of technical terminology, code snippets, domain-specific jargon, smaller corpora, clustering instability, outlier handling, and topic consistency issues across domains and languages (Azher et al., 2024; Z. Zhang et al., 2023; Ahmed et al., 2025; Fan et al., 2023; de Groot et al., 2022; Medvecki et al., 2023; Grootendorst and contributors, 2022; Zve et al., 2025). Thus, there is not yet a single standard topic modeling approach that is clearly preferable for software engineering datasets of this size and nature (Chang et al., 2024).

To improve the stability and interpretability of the LDA results, we ran multiple seeds and selected configurations based on topic coherence (Ahmed et al., 2025). Fig. 8(a) and 8(b) show that the resulting topics are well separated and support manual interpretation. The discovered Stack Overflow and GitHub topics were also independently reviewed and validated by the authors, achieving substantial agreement ($\kappa \approx 0.833$, $P_o = 85.7\%$) and near-perfect agreement ($\kappa \approx 0.916$, $P_o = 92.3\%$), respectively. The subsequent grouping into five high-level challenge families also achieved substantial agreement ($\kappa \approx 0.825$, $P_o = 85.7\%$). These validation steps support the consistency and interpretability of the identified topics. We leave a systematic comparison with more recent topic modeling techniques, such as BERTopic, for future work.

**Topic Modeling.** Our topic taxonomies rely on LDA-based modeling combined with manual validation. While we followed established practices for topic selection, coherence evaluation, and stratified sampling, topic modeling remains sensitive to pre-processing choices and parameter settings (Panichella et al., 2013). Topic boundaries should therefore be interpreted as approximate groupings rather than strict classifications. To mitigate this threat, we complemented automated modeling with manual inspection, inter-rater agreement analysis, and consensus-based labeling.

**Manual labeling**. Manual labeling necessarily involves a degree of subjectivity, particularly when assigning concise labels to topics and organizing them into higher-level categories. To mitigate this threat, we followed a structured labeling and reconciliation procedure commonly used in empirical software engineering studies that construct taxonomies from mined developer artifacts (Humbatova et al., 2020; Sayyadnejad et al., 2025; Uddin et al., 2021; Mahmood et al., 2023; Chen et al., 2020). Specifically, two authors independently analyzed and labeled the sampled posts and issues. Disagreements were discussed to reach consensus, and a third author validated the final labels and resolved any remaining conflicts. Moreover, our sample was stratified by topic size, and records within each topic were selected by considering the model's per-record topic contribution score, rather than relying solely on random sampling. Although this process cannot eliminate all subjectivity, it reduces individual-author bias and improves the consistency and traceability of the final labels.

**Temporal Validity.** The AI Agent ecosystem evolves rapidly, with frequent changes in frameworks, APIs, and deployment practices. Our analysis reflects the state of developer challenges during the studied time window. While specific tools and failure manifestations may change over time, we expect the higher-level challenge families (e.g., orchestration, retrieval, interaction contracts, and runtime reliability) to remain relevant as long as Agent systems rely on autonomous control flow, persistent state, and tool-mediated execution.

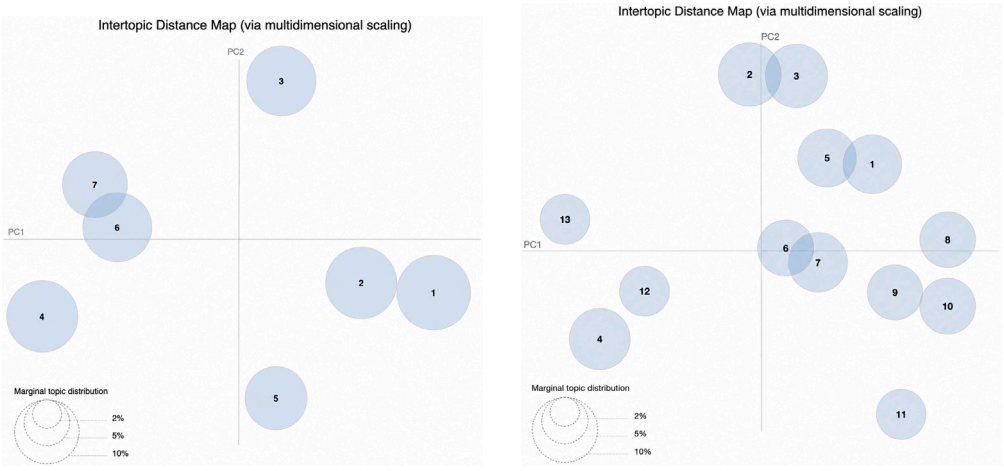
## 8. Conclusion

This study examined the technical challenges developers face when building AI Agent systems using large-scale data from Stack Overflow and GitHub. The results show that many of the difficulties encountered in practice arise from systems-level concerns rather than limitations of individual LLMs. Analysis of Stack Overflow questions identified seven recurring challenge topics, comprising 28 fine-grained subtopics related to installation, retrieval pipelines, orchestration, interaction contracts, and runtime reliability.

A complementary analysis of GitHub issues identified 13 Agent-related challenge topics that capture failures and maintenance burdens encountered during framework development, integration, and deployment. Across both platforms, configuration and prompting issues attract the most attention and are typically resolved quickly. In contrast, challenges related to retrieval pipelines, orchestration semantics, and long-running execution are less visible but consistently harder to resolve, exhibiting higher rates of unresolved questions and longer resolution times.

At the domain level, infrastructure-oriented frameworks report fewer issues overall, but these issues tend to remain active for longer periods. UI and workflow-centric platforms generate a higher volume of reports that are usually resolved more quickly. End-user Agent applications surface failures originating across multiple layers of the stack, revealing how design choices in orchestration, memory handling, and interaction contracts propagate to user-facing behavior. These findings suggest that progress in AI Agent systems depends less on improving model capability in isolation and more on strengthening orchestration mechanisms, stabilizing interaction contracts, and supporting persistent state in complex, tool-mediated workflows.

This study opens several directions for future research. First, incorporating additional data sources, such as proprietary issue trackers or industry incident reports, could provide a broader view of Agent failures in production settings. Second, longitudinal studies could examine how specific challenges evolve as Agent frameworks mature and as new abstractions emerge. Third, combining mining-based analysis with controlled empirical evaluations may help establish causal links between design decisions and observed failure/issues. Fourth, the feature requests and enhancement issues above show that developers




(a) Stack Overflow topics (b) GitHub Issues topics

**Fig. 8.** Inter-topic distance maps generated with pyLDAvis for the LDA-MALLET topics. The maps provide a qualitative view of topic separation and support the manual interpretation of the Stack Overflow and GitHub Issues topics.

are not just reporting problems but also sharing their expectations for the future of Agent frameworks. Our analysis points to this need, but further study is needed to systematically understand these requests and to see how they can guide the next generation of Agent framework designs.

## CRediT authorship contribution statement

**Ali Asgari:** Writing – original draft, Resources, Methodology, Investigation, Formal analysis, Data curation. **Annibale Panichella:** Writing – review & editing, Validation, Supervision, Formal analysis, Conceptualization. **Pouria Derakhshanfar:** Conceptualization. **Mitchell Olsthoorn:** Writing – review & editing, Validation, Supervision, Formal analysis.

## Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Ali Asgari reports financial support was provided by JetBrains Research. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work was conducted as part of the AI for Software Engineering (AI4SE) collaboration between JetBrains and Delft University of Technology. The authors gratefully acknowledge the financial support provided by JetBrains, which made this research possible.

## Data availability

The data will be published on zenodo after acceptance. Now also it is available for your view by using the link mentioned in the paper.

## References

Abdellatif, A., Costa, D., Badran, K., Abdalkareem, R., Shihab, E., 2020. Challenges in chatbot development: A study of stack overflow posts. In: Proceedings of the 17th International Conference on Mining Software Repositories. pp. 174–185.

Acharya, D.B., Kuppan, K., Divya, B., 2025. Agentic ai: Autonomous intelligence for complex goals–a comprehensive survey. IEEE Access.

Ahmed, S., Bagherzadeh, M., 2018. What do concurrency developers ask about? A large-scale study using stack overflow. In: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. pp. 1–10.

Ahmed, M., Opu, M.N.I., Roy, C., Suhi, S.I., Chowdhury, S., 2025. Exploring challenges in test mocking: Developer questions and insights from StackOverflow. ArXiv preprint arXiv:2505.08300.

Akbarpour, N., Mirza, A.S., Raoofian, E., Fard, F., Rodríguez-Pérez, G., 2025. Unveiling ruby: Insights from stack overflow and developer survey. ArXiv preprint arXiv:2503.19238.

Alamin, M.A.A., Uddin, G., Malakar, S., Afroz, S., Haider, T., Iqbal, A., 2023. Developer discussion topics on the adoption and barriers of low code software development platforms. Empir. Softw. Eng. 28 (1), 4.

Alanazi, A., Alfayez, R., 2024. What is discussed about Flutter on Stack Overflow (SO) question-and-answer (Q&A) website: An empirical study. J. Syst. Softw. 215, 112089.

Allamanis, M., Sutton, C., 2013. Why, when, and what: analyzing stack overflow questions by topic, type, and code. In: 2013 10th Working Conference on Mining Software Repositories. MSR, IEEE, pp. 53–56.

Alshangiti, M., Sapkota, H., Murukannaiah, P.K., Liu, X., Yu, Q., 2019. Why is developing machine learning applications challenging? A study on stack overflow posts. In: 2019 Acm/Ieee International Symposium on Empirical Software Engineering and Measurement. Esem, IEEE, pp. 1–11.

Asgari, A., de Koning, M., Derakhshanfar, P., Panichella, A., 2026. Metamorphic testing of deep code models: A systematic literature review. ACM Trans. Softw. Eng. Methodol.

Asgari, A., Guerriero, A., Pietrantuono, R., Russo, S., 2024. From testing to evaluation of NLP and LLM systems: An analysis of researchers and practitioners perspectives through systematic literature review and developers' community platforms mining.

Asgari, A., Guerriero, A., Pietrantuono, R., Russo, S., et al., 2025. Adaptive probabilistic operational testing for large language models evaluation.

Azher, I.A., Seethi, V.D.R., Akella, A.P., Alhoori, H., 2024. Limtopic: Llm-based topic modeling and text summarization for analyzing scientific articles limitations. In: Proceedings of the 24th ACM/IEEE Joint Conference on Digital Libraries. pp. 1–12.

Bagherzadeh, M., Khatchadourian, R., 2019. Going big: a large-scale study on what big data developers ask. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 432–442.

Bajaj, K., Pattabiraman, K., Mesbah, A., 2014. Mining questions asked by web developers. In: Proceedings of the 11th Working Conference on Mining Software Repositories. pp. 112–121.

Barua, A., Thomas, S.W., Hassan, A.E., 2014. What are developers talking about? An analysis of topics and trends in stack overflow. Empir. Softw. Eng. 19 (3), 619–654.

Bhandari, G., Naseer, A., Moonen, L., 2021. Cvefixes: automated collection of vulnerabilities and their fixes from open-source software. In: Proceedings of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering. pp. 30–39.

Bi, T., Liang, P., Tang, A., Xia, X., 2021. Mining architecture tactics and quality attributes knowledge in stack overflow. J. Syst. Softw. 180, 111005.

Braiek, H.B., Khomh, F., Adams, B., 2018. The open-closed principle of modern machine learning frameworks. In: Proceedings of the 15th International Conference on Mining Software Repositories. pp. 353–363.

Bu, W., Shu, H., Kang, F., Hu, Q., Zhao, Y., 2023. Software subclassification based on bertopic-bert-bilstm model. Electronics 12 (18), 3798.

Cemri, M., Pan, M.Z., Yang, S., Agrawal, L.A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., et al., 2025. Why do multi-agent llm systems fail? ArXiv preprint arXiv:2503.13657.

Chang, S., Wang, R., Ren, P., Wang, Q., Huang, H., 2024. A large language model guided topic refinement mechanism for short text modeling. ArXiv preprint arXiv:2403.17706.

Chen, Z., Cao, Y., Liu, Y., Wang, H., Xie, T., Liu, X., 2020. A comprehensive study on challenges in deploying deep learning based software. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 750–762.

Chen, X., Wang, J., Gao, C., Ju, X., Cui, Z., 2026. What developers ask about openai APIs: An empirical study on stack overflow. J. Syst. Softw. 112867.

Cochran, W.G., 1977. Sampling Techniques. John Wiley & Sons.

Cohen, J., 1960. A coefficient of agreement for nominal scales. Educ. Psychol. Meas. 20 (1), 37–46.

de Groot, M., Aliannejadi, M., Haas, M.R., 2022. Experiments on generalizability of BERTopic on multi-domain short text. ArXiv preprint arXiv:2212.08459.

de Witt, C.S., 2025. Open challenges in multi-agent security: Towards secure systems of interacting ai agents. ArXiv preprint arXiv:2505.02077.

Deng, Z., Guo, Y., Han, C., Ma, W., Xiong, J., Wen, S., Xiang, Y., 2025. Ai agents under threat: A survey of key security challenges and future pathways. ACM Comput. Surv. 57 (7), 1–36.

Du, W., Ding, S., 2021. A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications. Artif. Intell. Rev. 54 (5), 3215–3238.

Du, C., Wang, C., Chao, Y., Xie, X., Cui, Y., 2025. AI agent communication from internet architecture perspective: Challenges and opportunities. ArXiv preprint arXiv:2509.02317.

Elshan, E., Bruhin, O., Schmidt, N., et al., 2024. Unveiling challenges and opportunities in low code development platforms: A StackOverflow analysis. In: Proceedings of the 57th Hawaii International Conference on System Sciences. HICSS.

Fan, Y., Shi, L., Yuan, L., 2023. Topic modeling methods for short texts: A survey. J. Intell. Fuzzy Systems 45 (2), 1971–1990.

Fincher, S., Tenenberg, J., 2005. Making sense of card sorting data. Expert Syst. 22 (3), 89–93.

Gabriel, I., Keeling, G., Manzini, A., Evans, J., 2025. We need a new ethics for a world of AI agents. ArXiv preprint arXiv:2509.10289.

Grootendorst, M., contributors, 2022. BERTopic Issue #461: Discussion on outliers / UMAP–HDBSCAN stability. https://github.com/MaartenGr/BERTopic/issues/461. (Accessed 13 October 2025).

Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N.V., Wiest, O., Zhang, X., 2024. Large language model based multi-agents: A survey of progress and challenges. In: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence. IJCAI-24, pp. 8048–8057, Survey Track. http://dx.doi.org/10.24963/ijcai.2024/890.

Guo, J., Huang, S., Li, M., Huang, D., Chen, X., Zhang, R., Guo, Z., Yu, H., Yiu, S.-M., Jensen, C., Lam, K.-Y., 2025. A comprehensive survey on benchmarks and solutions in software engineering of LLM-empowered agentic system. ArXiv preprint arXiv:2510.09721.

Han, J., Shihab, E., Wan, Z., Deng, S., Xia, X., 2020. What do programmers discuss about deep learning frameworks. Empir. Softw. Eng. 25 (4), 2694–2747.

Haque, M.U., Iwaya, L.H., Babar, M.A., 2020. Challenges in docker development: A large-scale study using stack overflow. In: Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM, pp. 1–11.

Hindle, A., Alipour, A., Stroulia, E., 2016. A contextual approach towards more accurate duplicate bug report detection and ranking. Empir. Softw. Eng. 21 (2), 368–410.

Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A., et al., 2020. spaCy: Industrial-Strength Natural Language Processing in Python. Zenodo, Honolulu, HI, USA.

Humbatova, N., Jahangirova, G., Bavota, G., Riccio, V., Stocco, A., Tonella, P., 2020. Taxonomy of real faults in deep learning systems. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. pp. 1110–1121.

Huq, S.F., Galib, A.H., Jamil, A.R.M., Kabir, A., 2020. On the characteristics and applicability of topic modeling in predicting lifetime of GitHub issues.

Jin, B., Li, H., Nie, P., Zou, Y., 2026. Energy-efficient software development: A multi-dimensional empirical analysis of stack overflow. In: Proceedings of the IEEE/ACM International Conference on Software Engineering. ICSE, Rio de Janeiro, Brazil, http://dx.doi.org/10.1145/3744916.3773177.

Kolt, N., 2025. Governing AI agents. ArXiv preprint arXiv:2501.07913.

Kubiak, A.P., Kawalec, P., 2022. Prior information in frequentist research designs: The case of Neyman's sampling theory. J. Gen. Philos. Sci. 53 (4), 381–402.

Landis, J.R., Koch, G.G., 1977. The measurement of observer agreement for categorical data. Biometrics 33 (1), 159–174.

Li, H., Khomh, F., Openja, M., et al., 2021. Understanding quantum software engineering challenges an empirical study on stack exchange forums and github issues. In: 2021 IEEE International Conference on Software Maintenance and Evolution. ICSME, IEEE, pp. 343–354.

Lin, H.Y., Thongtanunam, P., Treude, C., Charoenwet, W., 2024. Improving automated code reviews: Learning from experience. In: Proceedings of the 21st International Conference on Mining Software Repositories. pp. 278–283.

Lin, Z., Zhang, N., Liu, C., Zheng, Z., 2025. An empirical study of ChatGPT-related projects and their issues on GitHub. Expert Syst. Appl. 267, 126113.

Liu, J., Wang, K., Chen, Y., Peng, X., Chen, Z., Zhang, L., Lou, Y., 2024. Large language model-based agents for software engineering: A survey. ArXiv preprint arXiv:2409.02977.

Loper, E., Bird, S., 2002. Nltk: The natural language toolkit. ArXiv Preprint Cs/0205028.

Luo, J., Zhang, W., Yuan, Y., Zhao, Y., Yang, J., Gu, Y., Wu, B., Chen, B., Qiao, Z., Long, Q., et al., 2025. Large language model agent: A survey on methodology, applications and challenges. ArXiv preprint arXiv:2503.21460.

Mahmood, K., Rasool, G., Sabir, F., Athar, A., 2023. An empirical study of web services topics in web developer discussions on stack overflow. IEEE Access 11, 9627–9655.

McCallum, A.K., 2002. Mallet: A machine learning for languagetoolkit. http://mallet.cs.umass.edu.

McHugh, M.L., 2012. Interrater reliability: the kappa statistic. Biochem. Medica 22 (3), 276–282.

Medvecki, D., Bašaragin, B., Ljajić, A., Milošević, N., 2023. Multilingual transformer and bertopic for short text topic modeling: The case of serbian. In: Conference on Information Technology and Its Applications. Springer, pp. 161–173.

Mo, T., Jiang, Z., Zheng, Q., 2025. Interactive AI agent for code refactoring assistance: A study on decision-making strategies and human-agent collaboration effectiveness. Acad. Nexus J. 4 (1).

Naghshzan, A., Ratte, S., 2023. Enhancing api documentation through bertopic modeling and summarization. ArXiv preprint arXiv:2308.09070.

Neyman, J., 1938. Contribution to the theory of sampling human populations. J. Amer. Statist. Assoc. 33 (201), 101–116.

OBrien, D., Biswas, S., Imtiaz, S., Abdalkareem, R., Shihab, E., Rajan, H., 2022. 23 shades of self-admitted technical debt: an empirical study on machine learning software. In: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 734–746.

Openja, M., Adams, B., Khomh, F., 2020. Analysis of modern release engineering topics:–a large-scale study using stackoverflow–. In: 2020 IEEE International Conference on Software Maintenance and Evolution. ICSME, IEEE, pp. 104–114.

Panichella, A., Dit, B., Oliveto, R., Di Penta, M., Poshynanyk, D., De Lucia, A., 2013. How to effectively use topic models for software engineering tasks? An approach based on genetic algorithms. In: 2013 35th International Conference on Software Engineering. ICSE, pp. 522–531. http://dx.doi.org/10.1109/ICSE.2013.6606598.

Plaat, A., Wong, A., Verberne, S., Broekens, J., van Stein, B., Back, T., 2025. Agentic large language models, a survey. ArXiv preprint arXiv:2503.23037.

Raheem, T., Hossain, G., 2025. Agentic AI systems: Opportunities, challenges, and trustworthiness. In: 2025 IEEE International Conference on Electro Information Technology. EIT, IEEE, pp. 618–624. http://dx.doi.org/10.1109/eIT64391.2025.11103638.

Řehůřek, R., Sojka, P., 2010. Software Framework for Topic Modelling with Large Corpora. University of Malta.

Renthlei, Z., Lallawmkima, C., 2025. Choosing the right sample size in social science studies: A methodological review. Mizoram Educ. J. 1.

Rizk, Y., Awad, M., Tunstel, E.W., 2018. Decision making in multiagent systems: A survey. IEEE Trans. Cogn. Dev. Syst. 10 (3), 514–529.

Röder, M., Both, A., Hinneburg, A., 2015. Exploring the space of topic coherence measures. In: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining. pp. 399–408.

Rosen, C., Shihab, E., 2016. What are mobile developers asking about? A large scale study using stack overflow. Empir. Softw. Eng. 21 (3), 1192–1223.

Sapkota, R., Roumeliotis, K.I., Karkee, M., 2025. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. ArXiv preprint arXiv:2505.10468.

Sapkota, R., Roumeliotis, K.I., Karkee, M., 2026. AI agents vs. Agentic AI: A conceptual taxonomy, applications and challenges. Inf. Fusion 126, 103599.

Sayyadnejad, M.M., Asgari, A., Sami, A., Tahayori, H., 2025. Exploring the black box: analysing explainable AI challenges and best practices through stack exchange discussions. Empir. Softw. Eng. 30 (6), 176.

Schneider, J., 2025. Generative to agentic ai: Survey, conceptualization, and challenges. ArXiv preprint arXiv:2504.18875.

Sengupta, S., Haythornthwaite, C., 2020. Learning with comments: An analysis of comments and community on stack overflow.

Shetty, M., Chen, Y., Somashekar, G., Ma, M., Simmhan, Y., Zhang, X., Mace, J., Vandevoorde, D., Las-Casas, P., Gupta, S.M., et al., 2024. Building ai agents for autonomous clouds: Challenges and design principles. In: Proceedings of the 2024 ACM Symposium on Cloud Computing. pp. 99–110.

Silva, C.C., Galster, M., Gilson, F., 2021. Topic modeling in software engineering research. Empir. Softw. Eng. 26 (6), 120.

Tahir, A., Yamashita, A., Licorish, S., Dietrich, J., Counsell, S., 2018. Can you tell me if it smells? A study on how developers discuss code smells and anti-patterns in stack overflow. In: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018. pp. 68–78.

Tian, F., Luo, A., Du, J., Xian, X., Specht, R., Wang, G., Bi, X., Zhou, J., Kundu, A., Srinivasa, J., et al., 2025. An outlook on the opportunities and challenges of multi-agent ai systems. ArXiv preprint arXiv:2505.18397.

Uddin, G., Sabir, F., Guéhéneuc, Y.-G., Alam, O., Khomh, F., 2021. An empirical study of IoT topics in IoT developer discussions on stack overflow. Empir. Softw. Eng. 26 (6), 121.

Wang, C., Chen, Z., Zhou, M., 2023. Automl from software engineering perspective: Landscapes and challenges. In: 2023 IEEE/ACM 20th International Conference on Mining Software Repositories. MSR, IEEE, pp. 39–51.

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., Zhao, W.X., Wei, Z., Wen, J.-R., 2024. A survey on large language model based autonomous agents. Front. Comput. Sci. 18 (6), 186345.

Wang, et al., 2025. A survey on AgentOps: Categorization, challenges, and future directions. Verify full author list before use. ArXiv preprint arXiv:2508.02121.

Woolson, R.F., Bean, J.A., Rojas, P.B., 1986. Sample size for case-control studies using Cochran's statistic. Biometrics 927–932.

Yang, J., Jimenez, C.E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., Press, O., 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. Adv. Neural Inf. Process. Syst. 37, 50528–50652.

Yang, X.-L., Lo, D., Xia, X., Wan, Z.-Y., Sun, J.-L., 2016. What security questions do developers ask? A large-scale study of stack overflow posts. J. Comput. Sci. Tech. 31 (5), 910–924.

Yang, W., Zhang, C., Pan, M., 2023. Understanding the topics and challenges of GPU programming by classifying and analyzing stack overflow posts. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 1444–1456.

Zhang, Z., Chen, C., Liu, B., Liao, C., Gong, Z., Yu, H., Li, J., Wang, R., 2023. Unifying the perspectives of nlp and software engineering: A survey on language models for code. ArXiv preprint arXiv:2311.07989.

Zhang, B., Liang, P., Zhou, X., Ahmad, A., Waseem, M., 2023. Practices and challenges of using github copilot: An empirical study. ArXiv preprint arXiv:2303.08733.

Zhu, X., Wu, J., Zhang, X., Li, T., Mu, Y., Zhai, J., Shen, C., Fang, C., Liu, Y., 2026. An empirical study of bugs in modern LLM agent frameworks. In: Companion Proceedings of the 34th ACM Symposium on the Foundations of Software Engineering. FSE Companion '26.

Zou, A., Lin, M., Jones, E., Nowak, M., Dziemian, M., Winter, N., Grattan, A., Nathanael, V., Croft, A., Davies, X., et al., 2025. Security challenges in ai agent deployment: Insights from a large scale public competition. ArXiv preprint arXiv:2507.20526.

Zve, E., Icard, B., Breton, A., Sainero, L., Bourgne, G., Ganascia, J.-G., 2025. From outliers to topics in language models: Anticipating trends in news corpora. ArXiv preprint arXiv:2509.22030.