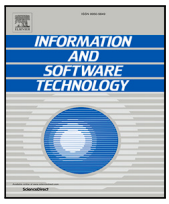
Information and Software Technology 197 (2026) 108183

Contents lists available at ScienceDirect

Information and Software Technology

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



The hands behind the agents: Understanding practitioner challenges with agentic frameworks

Rinad Hamid *, John Pangas, Ahmad Abdellatif

Department of Electrical and Software Engineering, University of Calgary, Alberta, Canada

ARTICLE INFO

Keywords:
Agentic frameworks
Large Language Models (LLMs)
Developer challenges
Stack Overflow
Crowdsource

ABSTRACT

Context: Agentic frameworks such as LangChain, CrewAI, and AutoGen have become important for building autonomous AI systems by coordinating LLM-based agents. However, despite rapid adoption, limited evidence exists about the real-world challenges practitioners face during development. Prior research has focused mainly on post-deployment issues such as runtime and communication problems. This leaves limited understanding of the obstacles that occur before deployment. These challenges hinder progress, expose documentation gaps, and reveal mismatches between framework abstractions and practitioner needs.

Objective: This study aims to identify and characterize the challenges practitioners encounter when using agentic frameworks, examine the types of questions developers ask, and quantify which technical areas are the most difficult for the community to resolve.

Methods: We analyze 2658 Stack Overflow posts related to agentic frameworks. Using co-occurring tag analysis and Latent Dirichlet Allocation (LDA), we extract recurring topics and group them into categories. We then manually code posts to classify question types, including "How", "Why", and "What". Topic difficulty is assessed using the percentage of unresolved posts and the median time to accepted answers.

Results: Practitioners most frequently struggle with framework configuration, tool integration, vector database operations, and agent execution behavior. A majority of posts, 71.1%, are procedural "How" questions, showing a need for practical guidance. LLM execution and runtime issues have the highest proportion of unanswered posts at 86%. Vector database related problems take the longest to resolve, with a median time of 100.3 h. In contrast, library and output or token issues are resolved more quickly.

Conclusion: This study offers the first large-scale empirical analysis of developer challenges with agentic frameworks. The findings highlight barriers in data management, orchestration, runtime execution, and configuration. Improving documentation, abstractions, and debugging tools can support more reliable and accessible agentic system development.

1. Introduction

The rapid evolution of Large Language Models (LLMs) has fundamentally transformed the landscape of software development, notably by catalyzing the emergence of autonomous agent systems [1]. Software agent is an autonomous entity that perceives its environment through natural language interfaces, reasons about goals and context, and acts upon that environment through tool use and multi step execution [2,3]. The commercial significance of agentic AI is substantial; the global agentic AI market was valued at approximately $5.25 billion in 2024 and is projected to reach $199.05 billion by 2034 [4]. This explosive growth reflects the critical role these systems play in enabling sophisticated automation across industries, from automated research assistants to intelligent workflow copilots.

The motivation for developing agentic systems stems from the limitations of monolithic LLM applications, which struggle with complex, multi step tasks requiring tool use, iterative reasoning, and dynamic adaptation [5]. Agentic systems address these constraints by facilitating task decomposition, external knowledge retrieval, and autonomous decision making [6]. To translate these capabilities into deployable applications, practitioners rely on specialized agentic frameworks such as LangChain, AutoGen, and CrewAI, which provide the necessary

* Corresponding author.
E-mail addresses: rinad.hamid@ucalgary.ca (R. Hamid), john.pangas@ucalgary.ca (J. Pangas), ahmad.abdellatif@ucalgary.ca (A. Abdellatif).

https://doi.org/10.1016/j.infsof.2026.108183
Received 26 November 2025; Received in revised form 10 April 2026; Accepted 5 May 2026
Available online 9 May 2026

structure for developing and orchestrating agentic workflows. However, despite the abstractions these tools provide, practitioners face significant implementation challenges when developing with these frameworks.

Prior studies have investigated the challenges in agentic system development, identifying issues such as runtime instability [7], adversarial security vulnerabilities [8], and communication inefficiencies [9]. Yet, these studies primarily focus on the execution and runtime phase, analyzing systems that are already built and deployed to understand why they perform poorly [7,10–12]. This leaves a critical gap in our understanding of the development phase: the practical, pre-deployment challenges practitioners encounter when attempting to configure, integrate, and orchestrate the complex components in agentic frameworks. Furthermore, these challenges are compounded by the rapid and continuous evolution of agentic frameworks [13]. LangChain, AutoGen, and similar platforms are frequently updated with new tools, breaking changes, and evolving best practices [14,15], often outpacing the inherent knowledge cutoff of general purpose LLMs.

To address the gaps identified in prior studies, our work examines the challenges practitioners discuss on Stack Overflow when using agentic frameworks. We analyze a comprehensive dataset of 2658 posts, extracted from the most prominent QA development website (i.e., Stack Overflow) related to major frameworks such as LangChain, CrewAI, and AutoGen to identify the primary technical difficulties and recurring conceptual themes. Our mixed-methods approach combines topic modeling to cluster related questions into coherent themes with quantitative and thematic analyses to identify patterns in question complexity and knowledge gaps. Our investigation is guided by the following research questions:

RQ1: Which topics emerge in discussions about agentic frameworks among practitioners? To answer this, we applied Latent Dirichlet Allocation (LDA) to extract latent themes, followed by a qualitative consensus process to label and synthesize these topics into a high level taxonomy. We find that agentic framework practitioners ask about 14 main topics that can be grouped into four categories: Data Management (30%), Integration (29%), Development (28%), and Prerequisites (13%).

RQ2: What types of questions are practitioners asking when working with existing agentic frameworks? To categorize practitioner intent, we manually coded a statistically representative sample of 1767 posts (95% confidence level) across the four identified categories. We find that the vast majority of questions (71.1%) are procedural "How" questions.

RQ3: Which topics present the greatest difficulty for practitioners to respond to effectively? We quantified difficulty by measuring the percentage of unresolved posts and the median time to an accepted solution. We find that the most difficult topics are those related to LLM Runtime (86% have no accepted answers) and complex data management activities such as vector database operations (median time: 100.3 h). In contrast, topics concerning Output & Token Management tend to receive faster responses (median time: 11.2 h). We did not find a statistically significant correlation between topic difficulty and topic popularity.

This study makes the following key contributions:

- We conduct the first large scale empirical analysis of the real-world challenges practitioners encounter when building with agentic frameworks.
- We quantify the severity of these challenges through community response metrics to identify the technical areas that are the hardest to resolve.
- We publish our replication package of the dataset and the code used in our analysis to facilitate further research.[1]

[1] https://zenodo.org/records/17708828

Building on these contributions, it becomes clear why addressing these gaps is essential. First, understanding the challenges practitioners face helps guide the development of more robust frameworks and improved developer tools. Second, it supports the creation of targeted educational resources and best practices. Finally, it strengthens the long term reliability and scalability of agent systems as they become increasingly central in different domains and industries.

Paper Organization. The remainder of this paper is organized as follows: Section 2 describes our methodology, Section 3 presents the results, Section 4 discusses their implications, Section 5 reviews related work, Section 6 considers threats to validity, and Section 7 concludes the paper.

2. Methodology

The primary objective of our work is to comprehend the types of challenges practitioners face when using agentic frameworks to develop LLM-based agents. We utilized Stack Overflow as our primary data source, as the platform provides a candid record of the questions and issues practitioners face in practice. Stack Overflow was selected over other sources, such as discussion forums or GitHub, due to several key advantages, consistent with prior software engineering studies [16–18]. First, Stack Overflow has a large and active developer community, which ensures diverse perspectives and a high volume of real-world programming problems. Second, the platform enforces a structured question and answer format, along with voting and reputation mechanisms, which helps surface the most relevant and high quality content. Third, unlike GitHub issues, Stack Overflow questions are generally self-contained and focused on specific technical problems rather than broader project discussions, making them well-suited for systematic analysis. One might argue that practitioners could refer to LLMs to answer their technical questions. However, Jimenez et al. [19] show that LLMs often fail to answer questions that require deep reasoning across multiple file dependencies and complex system interactions. Their findings indicate that current models struggle to maintain a correct understanding of the full software context, meaning that many of these issues still require human developers to interpret, trace, and resolve them. Furthermore, LLMs are trained on data up to specific cut-off dates, which makes them unable to address newly emerged questions. For example, if a new framework is released after an LLM has been trained, it would not be able to answer questions about it. Therefore, practitioners often turn to Stack Overflow for help with their technical challenges (as discussed in Section 4). Fig. 1 summarizes the overall methodology, which unfolds across five stages. Each of these stages is elaborated on in the remainder of this section.

Step 1: Stack Overflow dataset acquisition & preparation. We began by downloading the publicly available Stack Overflow archive, which is distributed in raw XML format. This raw data was subsequently converted into a structured relational format to enable large scale processing and analysis. The resulting dataset encompasses posts created from the platform's inception in August 2008 through January 2025 and contains approximately 60 million entries, including both questions and answers. Each entry is characterized by its core text (question or answer) alongside key metadata, such as programming language tags, vote scores, and timestamps, providing multiple dimensions for the subsequent empirical analysis.

Step 2: Collect co-occurring tags. Tags on Stack Overflow serve as user assigned labels that summarize the content of a post, indicating the technologies, frameworks, or concepts involved (e.g., *agent*, *langchain*). These tags are crucial for our study as they provide a first level indication of a post's relevance to agentic frameworks, which is necessary to effectively filter the millions of entries in the archive. Without this initial filtering, sifting through the dataset for relevant content would be infeasible.

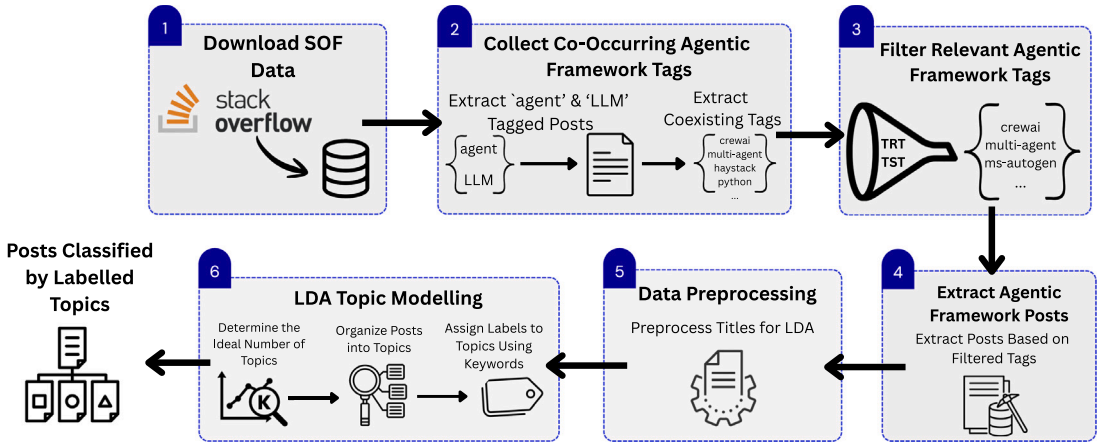


Fig. 1. Illustration of the methodology used in our study.

To identify the set of tags most relevant to agentic frameworks, we adapted the methodology of prior work on Stack Overflow post analysis [20]. This selection required a systematic procedure to ensure the resulting post set was both comprehensive and highly focused on the domain. We initially considered focusing on a broad collection of specific framework tags (e.g., *crewai*, *langchain-agents*, *llama-index*). This approach risked missing posts about emerging frameworks or introducing noise from general purpose technologies that co-occur with a wide range of tools. We also considered narrowing the selection to a single, highly specific framework tag such as *langchain-agents*. This approach was too narrow, limiting our analysis to a single ecosystem and failing to capture the broader developer discourse on agentic systems.

Ultimately, we chose to focus our initial extraction on posts requiring the co-occurrence of both the *large-language-model* and *agent* tags. This combination was selected for its high conceptual specificity, as our goal was to extract posts that truthfully reflect agentic LLM development frameworks, rather than general LLM use.

We specifically avoided using either tag as a standalone criterion because the *large-language-model* tag alone introduces excessive noise from general LLM usage (e.g., prompt engineering), while the *agent* tag alone often refers to general software agents unrelated to modern LLM-based agents.

By requiring the co-occurrence of both, we ensured the initial post set was highly focused and less noisy, directly linking the concept of an agent to the underlying Large Language Model technology.

Following this approach, we then extracted all posts containing both the *large-language-model* and *agent* tags, resulting in a total of 35 posts. From this subset, we then systematically extracted and counted all other tags that appeared alongside the required pair. This process yielded 27 unique candidate tags. These co-occurring tags capture additional concepts and technologies that practitioners frequently associate with agentic frameworks (e.g., langchain, pandas), that practitioners frequently associate with agentic frameworks, providing the necessary basis for expanding the dataset in the subsequent step.

Step 3: Filter tags for relevance and significance. To develop a focused and reliable final tag set, we recognized that the initial 27 co-occurring tags required refinement. Specifically, tags that were only weakly related to agentic frameworks or appeared too rarely to be useful needed to be excluded. For this filtering, we applied two heuristic metrics introduced in prior studies [20,21].

The first metric, Tag Relevance Threshold (TRT), captures the strength of association between a candidate tag and agentic framework related posts. TRT is calculated by dividing the number of posts containing both the candidate tag and the agentic framework tags (*large-language-model* and *agent*) by the total number of posts containing only the candidate tag. This metric helps filter out tags that are common in general contexts but have only a weak connection to agentic frameworks. Formally, the TRT is calculated as:

$$\mathrm{TRT}_{\mathrm{tag}} = \frac{No.\ of\ agentic\ framework\ posts\ for\ the\ tag}{Total\ number\ of\ posts\ for\ the\ tag}$$

Calculating TRT across all candidate tags allows us to distinguish which tags are strongly associated with agentic frameworks and which are less relevant. For example, the tag *crewai* has a TRT of 8.7%, implying that 8.7% of posts labeled with *crewai* are also tagged with both *large-language-model* and *agent*.

However, tags that appear in very few posts can sometimes show a high TRT, which may misleadingly suggest a strong association with agentic frameworks. For example, the *alpaca* tag occurs in only 11 posts, yet its TRT reaches 4.5% because five posts overlap with both *large-language-model* and *agent* tags. To mitigate the influence of such minimally relevant tags, we use a second metric called the Tag Significance Threshold (TST). TST quantifies how significant a candidate tag is within the set of posts related to agentic frameworks. TRT is calculated by counting how many posts contain both the target tag and the agentic framework tags. This count is then compared to the total number of posts containing the main tag combination, *large-language-model* and *agent* (35 posts), to see how strongly the target tag is associated with agentic frameworks. This metric is expressed as:

$$\mathrm{TST}_{\mathrm{tag}} = \frac{No.\ of\ agentic\ framework\ posts\ for\ the\ tag}{No.\ of\ agentic\ framework\ posts\ with\ the\ mostpopular\ tag}$$

This metric highlights tags that appear frequently in agentic framework discussions, excluding those that appear sporadically. For example, the *prompt* tag has a TST of 1.9%, implying that 1.9% of all agentic framework related posts tagged with *large-language-model* and *agent* also include the *prompt* tag.

To identify the final set of tags that are both relevant and significant to agentic frameworks, we established minimum thresholds for the TRT and TST. Instead of using a single cutoff, the first two authors independently explored several threshold combinations and assessed their impact on the set of co-occurring tags. For each candidate tag, the annotators examined a sample of 15 posts to determine when it started to become less related to agentic frameworks and shift toward broader, less relevant topics. This empirical approach ensured that the selected thresholds struck an appropriate balance between capturing meaningful content and excluding irrelevant noise. Following the best practices of prior empirical studies [22,23], we then compared our independent evaluations and discussed them among all authors to reach consensus on the most appropriate TRT and TST values. Based on this rigorous process, we determined that a TRT above 1.1% and a TST above 2.8% provided the best balance, preserving a representative range of agentic framework discussions while filtering out irrelevant posts. The final tag

Table 1
Final tag set for agentic framework posts with corresponding TRT and TST values (%)

Tag name	TRT (%)	TST (%)
langchain	1.2	74.3
py-langchain	1.3	17.1
crewai	8.7	11.4
langchain-agents	4.7	8.6
multi-agent	1.7	8.5
langgraph	1.9	5.7
openai-assistants-api	2.8	2.9
ms-autogen	1.9	2.9
litellm	25	2.8

set, defined using these thresholds, is shown in Table 1 alongside each tag's TRT and TST values.

Step 4: Construct agentic framework dataset. After establishing the final tags related to agentic frameworks, we extracted all Stack Overflow posts that included at least one of these tags (see Table 1) to form the primary corpus that will be used for our study. In total, we extracted 2658 agentic framework posts, each accompanied by the posts' attributes.

Step 5: Clean and Preprocess Agentic Framework Posts. To prepare the collected post dataset for in-depth qualitative analysis, we first refined the dataset to focus on elements most indicative of developer discussions.

We restricted our analysis to **post titles** because they clearly highlight the main point of each discussion. In contrast, the body of a post can be long and include extraneous details, code snippets, or unrelated information that might obscure the key patterns we are interested in.

Once all the titles were extracted, we implemented a preprocessing workflow using Python's NLTK [24] and Gensim [25] libraries, as follows:

- **Stopword Removal:** We filtered out a predefined set of common English stopwords using the NLTK corpus, as these terms lack significant semantic value for distinguishing topics.
- **Bigram Detection:** Some concepts in this domain are often expressed as multi-word phrases (e.g., "Langchain agent" or "multi agent systems"). To capture these as single units, we used a statistical model to identify pairs of words that frequently appear together in the post titles and combined them into single tokens for analysis.
- **Lemmatization:** We reduced all words to their canonical base, or lemma, to ensure that variations of the same word are treated consistently, thereby helping the model recognize patterns more accurately.

These preprocessing steps yielded a refined dataset that was subsequently used as input for the topic modeling stage.

Step 6: Apply Topic Modeling to Identify Agentic Framework Topics. After preprocessing, we set out to uncover the main themes within the agentic framework posts on Stack Overflow. For this purpose, we employed Latent Dirichlet Allocation (LDA) [26], a probabilistic method widely utilized in software engineering research [22,27,28]. LDA treats each document (post) as a mixture of multiple topics, producing a probability distribution that indicates how strongly a post aligns with each topic. We implemented this step using Mallet's version of LDA [29], which identifies the dominant topic of each post by selecting the one with the highest probability and its most representative keywords.

When applying LDA, it is necessary to determine the number of topics, denoted as K, since this parameter dictates how the posts will be grouped. Choosing an excessively high K can yield overly specific topics, whereas a very low K tends to generate broad topics that conflate multiple themes. To address this challenge, we explored K values from 5 to 25, incrementing by 1, and adopted a quantitative–qualitative hybrid approach:

1. **Quantitative Screening:** We computed the coherence score [30] for each value of K to objectively measure topic quality and previously used by similar work [16,22,27]. The coherence score assesses the semantic similarity between the most highly ranked words within a topic. A higher score indicates a better, more interpretable topic that human experts would judge as being consistently focused on a single concept. This process identified a set of candidate models ($K = 10$ to 16) with the most similar coherence values. Lower configurations ($K = 10$–11) exhibited topic conflation, where conceptually distinct themes were merged into a single topic. For example, posts on multi-agent system design (e.g., "Design pattern in multi-agent systems") and framework-specific debugging (e.g., "LangChain with Azure AttributeError") were grouped together, despite representing fundamentally different concerns, in this case agent architecture versus runtime errors. Conversely, higher configurations ($K = 16$) produced overly narrow topics that were difficult to interpret, such as isolating posts about a single library function or error type into their own topic, offering little analytical value beyond what a keyword search would provide.
2. **Qualitative Validation:** The first two authors independently manually reviewed a random sample of 50 posts from the topics generated by each candidate model. This assessment indicated that $K = 14$ produced topics that were both semantically distinct and interpretable, while also offering the best balance between breadth and specificity within our domain.

3. Results

In this section, we present the results of our study. We describe our motivation, methodology to answer the questions, and results for each research question.

3.1. RQ1: Which topics emerge in discussions about agentic frameworks among practitioners?

Motivation: Recent research has begun to explore how agentic systems powered by LLMs are being applied across various industries such as software engineering [2,31,32], science discovery [33], and finance [34]. Agentic frameworks are central to this shift because they provide the components needed to build agentic systems. Working with these systems requires skills such as prompting, task orchestration, and tool integration. As a result, practitioners often face uncertainties about how to apply these emerging practices in real-world applications. The goal of this research question is to examine the practitioners discussions on Stack Overflow to identify the main topics focus on when working with agentic frameworks. Mapping these topics provides an initial understanding of the areas that attract the most attention and the issues that commonly arise in practice.

Approach: To address this research question, we relied on the results of the Latent Dirichlet Allocation (LDA) analysis performed in Step 6 of our methodology (as detailed in Section 2). The resulting topics were then manually reviewed and refined to ensure clarity and accuracy. Specifically, each topic was reviewed independently by the first two authors, who examined the top keywords and a random sample of posts to assign a descriptive label representing its central theme. Following independent labeling, the annotators discussed all topics to reach a consensus on their final titles.

During this analysis, we observed that several topics addressed related aspects of agentic system development. For instance, some

Table 2
Topics on agentic frameworks, grouped by category, with popularity indicators. Topics are sorted within each category by number of posts (greatest to least).

Main Category	Topic	# Posts	Avg. Views	Avg. scores
Data Management	Memory & Context Retrieval	222	2485.12	1.02
	Document Parsing/Indexing	196	2022.88	1.03
	Multi-Input & Template Handling	191	1883.06	0.92
	Vector DB Operations	173	2062.62	1.01
Integration	Agent Orchestration	**245**	1715.61	1.09
	Model Deployment	185	1514.17	0.95
	API Usage	184	1705.95	0.84
	Text & Embedding Pipelines	166	1942.66	1.01
Development	LLM Execution & Runtime Issues	218	1587.57	0.85
	Output & Token Management	187	1806.30	1.12
	Advanced Chain Usage & Asynchronous Operations	181	1947.38	1.08
	Framework Libraries & Methods	155	2685.85	**1.86**
Prerequisites	Environment & Dependency Issues	220	**3350.04**	1.46
	Resource Configuration & Management	135	2664.11	1.34

topics focused on coordinating multiple agents or integrating external tools, as both these topics involve managing the interaction between different components. To provide a clearer and more organized overview, we grouped these related topics into broader categories to create a structured, hierarchical view of the discussions around agentic frameworks.

Finally, we assessed the popularity of each topic within the developer community to understand which areas attract the most attention and engagement. We selected two complementary metrics that have been used by prior work [20,35–37] to capture different aspects of popularity:

- **Average Views per Post (Avg. Views):** This metric captures a topic's overall visibility and level of developer interest. On Stack Overflow, the view count accounts for both registered and non registered users; thus, posts with higher view counts reflect topics that draw broader community attention and engagement.
- **Average Score per Post (Avg. Score):** This measures the perceived quality and value of a topic's discussions. Scores are based on community upvotes, reflecting how helpful or informative peers consider the associated content.

Results: Table 2 presents the 14 topics that emerged in Stack Overflow discussions about agentic frameworks. These 14 topics were organized into four main categories: `Data Management`, `Integration`, `Development`, and `Prerequisites`. This categorization helps illustrate which areas of agentic framework development attracted the most attention and the discourse surrounding different aspects of these frameworks. Below, we explore each category in depth to highlight the key themes in discussions on agentic frameworks.

Data Management: The posts in this category focused on handling, processing, and structuring data specifically for LLM based agents within agentic frameworks. Comprising 30% of the total dataset, this was the largest category observed. The prominence of these posts highlights the critical role of effective data handling in ensuring LLMs can access and use information accurately and efficiently across different workflows. This category includes four key topics: *Multi-Input & Template Handling*, *Document Parsing/Indexing*, *Vector DB Operations*, and *Memory & Context Retrieval*. Discussions in this area addressed the unique challenges practitioners faced in managing the context window, implementing Retrieval Augmented Generation (RAG), embedding and indexing documents for vector databases, and managing conversational memory. For example, one post asked, "How to use VectorStoreRetrieverMemory in langchain with PGVector?" [38]. The sheer volume of posts in this category shows the immense practical difficulty practitioners face in preparing, storing, and retrieving the specific information needed to power intelligent, context-aware agents.

Integration: The posts in this category focused on the challenges integrating different components of agentic frameworks. This category accounted for a substantial 29% of the dataset, which highlighted the essential role of integration in enabling LLM based agents to function effectively in real-world applications. Discussions in this area highlighted challenges practitioners faced when coordinating multiple agents, integrating external APIs, setting up pipelines for embeddings, or deploying models into production environments. For example, the post "How to apply a prompt on each retrieved document before answering, in the Langchain ConversationalRetrievalChain"[39] illustrates the practical difficulties practitioners encountered when combining multiple components into a single pipeline. Another illustrative example is "Multiple agents thread synchronization", which demonstrated the complexities of coordinating multiple agents simultaneously. Practitioners often encountered issues such as race conditions, timing conflicts, and shared resource management when multiple agents operated in parallel, underscoring the additional integration challenges that arise specifically in multi-agent systems. Notably, *Agent Orchestration* accounted for the highest number of posts across all topics, reflecting the central difficulty of coordinating multiple agents effectively.

Development: This category addressed the practical challenges practitioners faced when developing and executing agents using agentic frameworks. For instance, posts in this category included "LangChain returning answers out of context" [40] and "Authentication error when using LangChain agents" [41], illustrating the common difficulties in ensuring correct execution and reliable agent behavior. Discussions in this area covered issues such as LLM execution errors, handling asynchronous chains, and managing outputs effectively. Comprising 28% of the dataset, these posts emphasized the technical complexities inherent in developing functional, high performing LLM based agents within agentic frameworks.

Prerequisites: This category addressed the fundamental setup and initialization issues practitioners encountered when working with agentic frameworks. It encompassed posts related to establishing and managing the environment, dependencies, and computational resources required for these systems to function correctly. Discussions within this area constituted 13% of the total posts in the dataset, highlighting the challenges practitioners faced in meeting these initial requirements. Posts commonly describe problems such as installation errors, module import failures and library conflicts. For example, practitioners asked questions like "ModuleNotFoundError: No module named langchain in docker container" [42], or "LlamaIndex SIGILL on Mac M1" [43]. Collectively, these discussions highlighted the practical difficulties of fulfilling the necessary prerequisites for agentic frameworks and establishing the correct foundation of required packages, libraries, and resources for subsequent functionality.

In our analysis of topic popularity, we find that the most viewed discussions center on the `Prerequisites` and `Development` categories. Although the `Prerequisites` category represents only 13% of posts in the dataset, topics such as *Environment & Dependency Issues*

Table 3
Agent framework post types on stack overflow.

Main categories	% How	% Why	% What	% Other
Prerequisites	66.3	24.6	7.1	2.1
Development	73.2	14.0	11.5	1.4
Data Management	70.0	19.9	9.5	0.6
Integration	72.5	18.2	8.1	1.2
Agentic Frameworks (all)	71.1	18.3	9.3	1.2

attract the highest average views (3350). This suggests that setup challenges, while not frequently discussed, are highly visible and broadly relevant to the community. Its significance likely stems from the fact that environment and dependency issues often serve as blocking problems; even a single unresolved setup error can completely impede progress, leading practitioners to actively search for remedies. Within the `Development` category, which accounts for 28% of all posts, the topic *Framework Libraries & Methods* attains the highest average score (1.86). This pattern suggests that after overcoming initial setup challenges, practitioners place greater value on practical, peer generated guidance about specific frameworks than on standard documentation. Together, these trends point to a layered model of engagement: practitioners first concentrate on resolving common prerequisite issues and then shift toward acquiring more specialized knowledge within framework related topics.

3.2. *RQ2: What types of questions are practitioners asking when working with existing agentic frameworks?*

Motivation: Understanding the challenges practitioners face with agentic frameworks requires more than merely identifying popular topics. By examining the specific questions practitioners ask, we can gain insight into the practical difficulties they encounter while using agentic frameworks. Previous studies have indicated that the types of questions, such as "how", "why", or "what" often reflect different stages or kinds of problems practitioners encounter during their workflows [18,44]. Therefore, studying the patterns in these questions enables the community to uncover the specific difficulties practitioners encounter when using agentic frameworks.

Approach: To analyze discussions around agentic frameworks, we randomly sampled posts from each of the four major categories identified in RQ1. This approach allowed us to review a manageable subset of the dataset while ensuring representative coverage across all categories. The sample sizes for each category were determined to meet a 95% confidence level with a 1.5% margin of error. Our random sample yielded a total of 1767 posts, comprising 240 `Prerequisites` posts, 497 `Development` posts, 516 `Data Management` posts, and 514 `Integration` posts. Each post in the sample was independently reviewed by two authors, who assigned labels based on the following predefined question type categories:

- **How:** This category included posts where practitioners focused on the procedure or method needed to achieve a specific outcome. These questions typically outlined a goal and sought practical steps for implementation (e.g., "How to save history and restart a chat from last point in Langchain with Ollama?").
- **What:** This category represented questions that requested factual information or clarification. Practitioners raised these questions to better understand framework concepts, components, or definitions, which in turn supported more informed design choices (e.g., "In LangChain, what is an example conversation used for?").
- **Why:** Posts in this category were concerned with the underlying cause or rationale for a given behavior. They often emerged from troubleshooting scenarios, where the author expected an explanation for unexpected outcomes. (e.g., "Why does LangChain agent return the action input instead of running it?").
- **Other:** Posts that did not align with the three categories above were grouped in this category (i.e., "Other" category). These were often more open-ended, opinion seeking, or otherwise ambiguous in intent (e.g., Design Patterns).

To evaluate the consistency of our labeling process, we applied Cohen's Kappa, a statistical measure of inter-rater agreement that accounts for chance alignment. Overall, our agreement rate ($\kappa = 0.90$) reflected a strong level of consistency across both authors. For cases where no consensus was initially reached, the annotators engaged in follow up discussions until a shared decision was made. Finally, we conducted a verification step, reviewing the titles and bodies of the sampled posts to confirm their assigned categories.

Results: Table 3 summarizes the distribution of question types across the four major categories of agentic framework posts. Overall, a majority of posts (71.1%) were categorized as the "How" type, indicating that practitioners primarily sought practical guidance and procedural steps for implementing and utilizing agentic frameworks. "Why" questions made up 18.3% of the posts, reflecting troubleshooting concerns and requests for explanations of unexpected behaviors. Posts categorized as "What" represented 9.3%, showing a smaller but notable need for factual information or clarification.

When examining specific categories, `Development` posts exhibited the highest proportion of "How" questions (73.2%), which highlighted the strong demand for step-by-step implementation support during the development of LLM-based agents. `Prerequisites` posts showed the highest proportion of "Why" questions (24.6%), suggesting that practitioners frequently encountered unexpected issues related to environment setup, dependencies, or configuration before they began core development. The highest rate of conceptual questions ("What") was also found in the `Development` category (11.5%). This finding aligns with the notion that as practitioners actively build features, they require frequent clarification on the available components, definitions, or conceptual boundaries of the framework, which is critical for making informed design choices.

3.3. *RQ3: Which topics present the greatest difficulty for practitioners to respond to effectively?*

Motivation: After identifying the prevalent topics and the types of questions they included, we set out to assess the difficulty of responding to posts across these topics. Understanding which posts were more challenging to address provides valuable insights into the current gaps in community knowledge, documentation and available resources. By examining the difficulty of questions across different topics, we aim to pinpoint areas where practitioners encountered the most significant obstacles.

Approach: To assess the difficulty of questions within each topic, we adopted two measures commonly utilized in prior research [20,45,46]:

1. **Unresolved Posts (% without accepted answers).** This metric captured the proportion of posts in a topic that lacked an accepted answer. Because only the original author of the post can mark a solution as accepted, a higher percentage of unresolved posts indicated greater difficulty in addressing questions within that topic.

Table 4
Difficulty metrics by dominant topic (Sorted by % w/o Accepted Answer).

Topic	% w/o accepted answer	Median time to answer (Hrs.)
LLM Execution & Runtime Issues	**86.24**	21.35
Vector DB Operations	86.13	**100.34**
Advanced Chain Usage & Asynchronous Operations	85.64	19.16
Text & Embedding Pipelines	84.34	14.98
Memory & Context Retrieval	83.78	41.38
Model Deployment	83.24	39.51
Output & Token Management	82.89	11.26
Multi-Input & Template Handling	82.72	25.82
Framework Libraries & Methods	82.58	12.92
Agent Orchestration	82.04	31.87
Document Parsing/Indexing	81.63	20.80
API Usage	80.43	14.40
Resource Configuration & Management	80.00	20.83
Environment & Dependency Issues	76.36	22.61

Table 5
Spearman correlation of difficulty versus popularity of topics.

Correlation coeff.	Avg. views (p-value)	Avg. score (p-value)
% Without Accepted Answer	0.345 (0.227)	−0.122 (0.679)
Median Time to Answer (Hrs.)	0.314 (0.274)	−0.462 (0.096)

2. **Median Time to Get an Accepted Answer (hours).** This measure reflected the typical amount of time it took for a post to receive an accepted solution. We computed the interval between when the question was asked and when the accepted answer was submitted (not when it was later marked as accepted). Longer delays suggested that the question posed more of a challenge for the community.

To ensure a fair and reliable comparison of response times, consistent with prior Stack Overflow studies, we exclude posts created within the median response time window (21.39 h), as these posts have not yet had sufficient opportunity to receive answers. This step mitigates right censoring effects, where recent posts may appear unresolved or associated with longer response times simply due to insufficient observation time. Such an approach aligns with prior work in empirical software engineering [20,47–49], where response time statistics are computed on posts with adequate exposure or conditioned on resolved questions to ensure meaningful latency measurements.

Results: Table 4 presents the results of topic difficulty across agentic framework topics. From the table, we observe that a large share of posts related to agentic frameworks remain unresolved, and accepted answers often take substantial time to arrive. Across most topics, more than 80% of questions have no accepted solution, with the highest share observed in LLM Execution and Runtime Issues at 86.2%. This value is computed over the full curated dataset.

To better understand the reasons behind this high rate of unresolved questions in LLM Execution and Runtime Issues, we conducted a qualitative analysis on a randomly sampled set of 100 posts without accepted answers. The sample reveals consistent patterns across posts. Many questions lack critical information, such as library versions, environment configurations, or complete error traces, which limits reproducibility. In addition, a large portion of posts involve highly customized pipelines with unique combinations of prompts, tools, vector stores, and model parameters, making the issues tightly coupled to individual codebases.

We also observe that many questions are affected by rapid API evolution and version mismatches across frameworks, particularly in ecosystems such as LangChain, where breaking changes are frequent. Furthermore, several posts do not include minimal reproducible examples or present overly broad or multi-component code, which obscures the root cause of the issue. Finally, a subset of questions involves niche or less widely adopted tools, further reducing the likelihood of receiving timely and accurate responses.

In terms of responsiveness, the longest median time to receive an accepted answer was observed in *Vector DB Operations*, at 100.3 h. This value was a clear outlier compared to other topics, which generally ranged from 11 to 40 h. We hypothesize that this outlier is linked to the infrastructure dependent and opaque nature of vector database issues. Reproducing these failures often requires access to the original environment, including the vector database setup, the embedding model used to generate vectors, and in many cases the underlying dataset. Vector DB retrieval behavior depends heavily on these components [50,51]. This high barrier to reproduction leaves such questions to a smaller pool of specialized experts and thereby prolonging the time to resolution.

Other topics with notable delays included *Memory & Context Retrieval* (41.4 h) and *Model Deployment* (39.5 h). Posts concerning *Memory & Context Retrieval* [52,53] often hinge on how context windows, chunking strategies, and caching mechanisms are implemented within a specific codebase. Diagnosing such issues typically requires understanding not only the framework's API but also how memory is persisted across sessions, streaming callbacks, and concurrent executions [54]. Together, all of these factors raise the level of expertise needed to provide effective support. Furthermore, prior work [55–57] has also highlighted similar long standing challenges associated with maintaining reliable and interpretable memory components in agentic systems, which helps explain why these issues continue to be difficult to resolve in practice.

For *Model Deployment*, posts tend to be difficult to answer because they often depend on the user's exact deployment configuration including the underlying framework, cloud platform, model serving API and containerized environment [52,53]. This high degree of environment specific variability limits the number of contributors who can confidently diagnose the problem, contributing to the longer time to resolution.

Conversely, some topics received relatively faster responses from the Stack Overflow community, suggesting that greater familiarity with the subject matter reduced resolution time. For example, *Framework Libraries & Methods* and *Output & Token Management* had median times to answer of 12.92 h and 11.26 h, respectively. These topics typically involve practical, well understood tasks, such as implementing standard framework routines, handling API outputs, or managing tokens efficiently. Their comparatively shorter response times suggested that these topics required less domain expertise to address, in contrast to more specialized areas like vector database operations or memory retrieval.

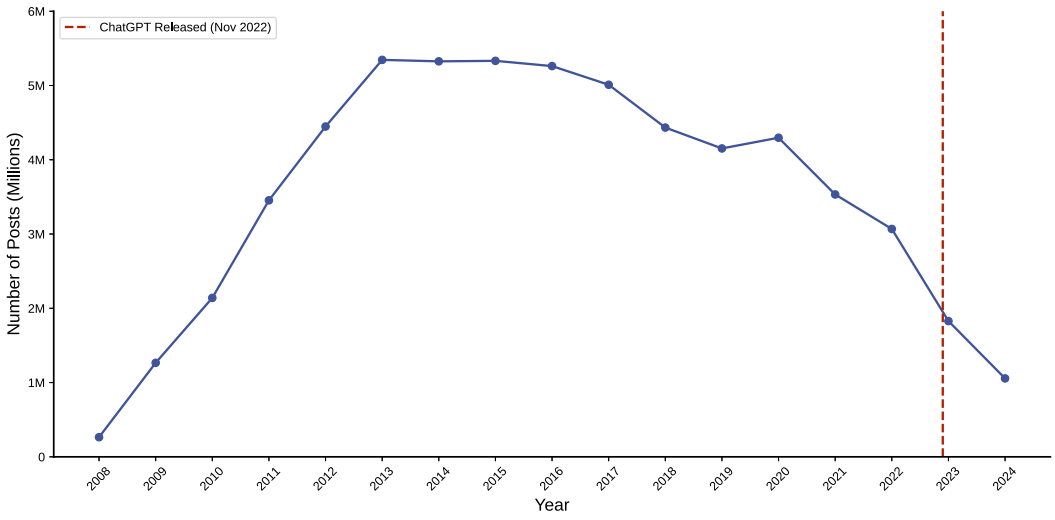


Fig. 2. Evolution of stack overflow posts per year.

To examine the relationship between topic difficulty and popularity, we computed Spearman Rank Correlation coefficients, as summarized in Table 5. Overall, the results indicated only weak associations between the difficulty metrics (percentage of unresolved posts and median time to answer) and the measures of popularity (average views and average score). Since all correlations had *p*-values greater than 0.05, none of these associations were statistically significant. These findings suggest that the most challenging topics were not necessarily the most visible or highly rated within the Stack Overflow community.

4. Discussion & implications

In this section, we discuss the evolving role of Stack Overflow in the age of AI, analyze framework-specific challenges, changes in topic resolution difficulty over time, and patterns observed in the LangChain community forum. In addition, we discuss the implications of our results.

4.1. The evolving role of stack overflow in the age of AI

The emergence of LLMs such as ChatGPT has reshaped how practitioners seek and exchange programming knowledge. As conversational systems became capable of generating code and answering routine questions, many speculated that community driven Q&A platforms like Stack Overflow were losing relevance. To investigate this, we analyzed posts created by software practitioners on Stack Overflow. Fig. 2 shows the number of posts published each year from 2008 to 2024. From the figure, we observe that overall posting volume declined from approximately four million posts in 2020 to three million in 2022. This downward trend became even more pronounced after the release of ChatGPT, with monthly posts falling from about 1.5 million in November 2022 to roughly 700,000 in 2024.

However, a different pattern emerges when examining posts related to agentic frameworks. In our analysis, questions involving frameworks such as LangChain, AutoGen, and CrewAI have risen steadily. This trend is almost entirely a post-2022 phenomenon: fewer than ten posts appeared in 2022, yet the number rose to 1049 in 2023 and then to 1274 in 2024. This surge indicates that while Stack Overflow is receiving fewer routine questions, it is becoming more important as a specialized venue for resolving complex issues associated with rapidly evolving AI tooling.

Recent research provides broader evidence of this shift in platform usage. Prior work [58,59] shows that while overall activity on Stack Overflow has declined, the remaining questions have become more complex and technical. Helic et al. [58] conclude that practitioners now rely on Stack Overflow for more advanced and larger scale issues, while delegating routine tasks to generative AI.

Complementing this, Zhong and Wang [60] demonstrate that LLM generated code is not consistently robust or reliable in practice and cannot fully replace community driven platforms such as Stack Overflow. Kabir et al. [61] further quantify these limitations by evaluating 517 programming questions and finding that 52% of ChatGPT responses contained incorrect information, 77% were overly verbose, and 78% contradicted human generated solutions. The majority of these errors stemmed from misapplied framework behaviors or references to non-existent APIs. Compounding this problem, 39% of users failed to detect the inaccuracies because the responses appeared fluent and confident.

These findings are further supported by recent industry evidence from the Stack Overflow Developer Survey 2025 [62], which collected responses from more than 32,000 developers. Notably, approximately 35% of developers report that their visits to Stack Overflow are driven, at least in part, by challenges related to AI systems. The survey also reports that more developers distrust the accuracy of AI tools (46%) than trust them (33%). In addition, 66% of developers identify "AI solutions that are almost right, but not quite" as a primary frustration, and 45% report that debugging AI-generated code can be more time consuming. Notably, 75% of respondents indicate that even in a future with advanced AI, they would still prefer to consult a human when they do not trust AI-generated answers. These findings reinforce the continued importance of human expertise and community driven platforms in supporting software development.

Furthermore, these limitations are amplified in the fast moving ecosystem of agentic frameworks [13,63,64], where libraries evolve rapidly and introduce frequent changes. In such settings, prior work shows that LLM generated code often fails to remain reliable in non trivial and evolving development scenarios. This is partly due to the reliance of LLMs on static training data and limited runtime context, which can result in outdated or incomplete guidance. As a result, developers often complement AI assisted development with community validated and up to date sources when addressing complex or evolving issues.

Our analysis of Stack Overflow posts reflects these same limitations. Many unresolved questions involve issues such as agent orchestration failures, runtime dependency management, tool integration, and vector database inconsistencies. These problems cannot be reliably addressed by generative AI alone, as they often require real time debugging, environment specific configuration, and an understanding of interactions across multiple system components. Consistent with this, over 80% of posts in most topics lacked an accepted answer, and

Table 6
Proportion of posts per topic by framework.

Topic name	Autogen	CrewAI	LangChain	Dominant framework
Agent Orchestration	10.0%	29.2%	7.5%	CrewAI
Framework Libraries	20.0%	4.2%	5.7%	Autogen
LLM Execution & Runtime Issues	10.0%	16.7%	8.0%	CrewAI
Multi-Input & Template Handling	14.0%	8.3%	7.3%	Autogen
API Usage	12.0%	0.0%	5.9%	Autogen
Memory & Context Retrieval	0.0%	8.3%	8.7%	LangChain
Environment & Dependency Issues	8.0%	8.3%	8.4%	LangChain
Model Deployment	8.0%	8.3%	7.2%	CrewAI
Document Parsing	8.0%	4.2%	7.7%	LangChain
Output & Token Management	4.0%	0.0%	7.6%	LangChain
Advanced Chain Usage	0.0%	4.2%	7.2%	LangChain
Vector DB Operations	4.0%	4.2%	6.7%	LangChain
Text & Embedding Pipelines	2.0%	0.0%	6.6%	LangChain
Resource Configuration	0.0%	4.2%	5.5%	LangChain

categories such as Vector DB Operations exhibited median response times exceeding 100 h. These long delays highlight the complexity of debugging agentic systems that depend on evolving libraries, intricate configuration patterns, and diverse execution environments. In such contexts, practitioners continue to rely on Stack Overflow for reproducible and experience grounded insights that complement documentation and current AI based tools.

4.2. Framework specific challenges

To investigate whether practitioners face platform-specific challenges, we conducted a framework-specific analysis to examine the distribution of challenge topics across LangChain, AutoGen, and CrewAI. Specifically, leveraging the LDA output, we assigned each post to its dominant topic, identified the associated framework based on tags, titles, and post content, and compared normalized topic distributions across frameworks. This analysis reveals distinct patterns in the types of challenges practitioners encounter across these frameworks. Table 6 summarizes the proportion of posts per topic for each framework, highlighting framework-specific patterns in practitioner difficulties. Percentages reflect the proportion of posts for each challenge topic within each framework, with a total of 2374 posts for LangChain, 24 posts for CrewAI, and 50 posts for AutoGen.

As shown in Table 6, multi-agent coordination challenges, captured under Agent Orchestration, are particularly prominent in CrewAI posts at 29.2%, compared to 10% in AutoGen and 7.5% in LangChain. This suggests that CrewAI users are more frequently exposed to complex orchestration scenarios requiring coordination between multiple agents. This trend likely reflects the framework's role-based design pattern, which requires developers to explicitly define hierarchies and process flows, thereby increasing the complexity of task delegation and inter-agent communication logic.

In contrast, AutoGen posts show a higher prevalence of Framework Libraries at 20% and Multi-Input and Template Handling at 14%. This indicates that library integration and template management are the primary technical hurdles for AutoGen practitioners. These findings point toward a development experience characterized by configuration complexity, where challenges arise from managing framework-specific abstractions and class structures rather than the agentic logic itself. This is consistent with documentation challenges noted in recent literature on large language model frameworks, where highly abstract configurations often lead to integration errors and syntax-related bottlenecks during initial setup.

For LangChain, Memory and Context Retrieval issues are more prominent at 8.7%, highlighting the difficulty of maintaining state and context continuity. Because LangChain is a modular ecosystem with numerous decentralized components, developers frequently encounter challenges in ensuring data flows correctly between decoupled chains. Unlike challenges driven by coordination in CrewAI or those driven by configuration in AutoGen, LangChain users face a broader distribution of issues across the entire stack.

Overall, these findings suggest that practitioner challenges are shaped by both framework design and adoption scale. These insights highlight the need for framework-specific documentation and tooling by project maintainers, as different frameworks expose distinct challenges such as orchestration, memory management, and integration complexity. Providing targeted support can help practitioners better navigate these framework-specific difficulties and reduce common implementation errors.

4.3. Temporal evolution of topic resolution difficulty

To examine whether certain topics are becoming easier or harder to resolve over time, we analyzed median resolution times (in hours) for posts across 2023 to 2025. Each topic was measured by the median time to receive an accepted answer, providing a precise indicator of resolution difficulty. Vector DB Operations was selected as a case study due to its consistently high median resolution times, which are often caused by infrastructure-dependent and opaque issues, including reliance on the original database environment, embedding model, and dataset.

Across topics, we observe substantial variation in resolution difficulty, as seen in Table 7. For Vector DB Operations, median resolution time increased sharply from 14.5 h in 2023 to 121.8 h in 2024, and further to 270.1 h in 2025, as shown in Table 7. This trend suggests that as the topic became more widely adopted, the combination of environment specific dependencies, complex model integrations, and evolving embedding methodologies led to increasingly prolonged resolution times [65,66]. These failures often stem from infrastructure dependent variables, such as semantic drift between evolving embedding models and legacy datasets, or opaque environment-specific dependencies that elude standard troubleshooting. This trend aligns with recent observations in LLM observability research, which notes that as developers move beyond simple proof-of-concepts, they encounter increasingly complex challenges regarding retrieval precision and long-term memory consistency in fragmented database environments [67–69].

Similarly, Multi-Input & Template Handling also shows a dramatic increase in 2025, reaching 572 h (Table 7), highlighting emerging challenges in constructing and managing agentic workflows. This increase suggests an emerging bottleneck in the construction of sophisticated agentic workflows. As developers move toward nested, multi-modal templates and dynamic prompt injection, the state space for potential errors expands exponentially. Unlike earlier iterations of prompt engineering, these multi-input failures often lack clear error logs, forcing developers into prolonged trial-and-error cycles.

In contrast, the data for Agent Orchestration provides evidence of successful pattern maturation, with resolution times dropping precipitously from 96.9 h in 2023 to just 1.3 h in 2025, as seen in Table

Table 7
Median resolution times (in hours) across topics from 2023 to 2025.

Topic	2023	2024	2025
API Usage	28.8	14.4	32.8
Advanced Chain Usage	21.5	15.8	16.3
Agent Orchestration	96.9	42.5	1.3
Document Parsing	21.3	13.5	134.8
Environment & Dependency Issues	36.4	14.3	72.9
Framework Libraries	22.5	26.1	29.2
LLM Execution & Runtime Issues	77.0	14.7	16.9
Memory & Context Retrieval	37.9	57.9	22.5
Model Deployment	81.1	25.8	189.5
Multi-Input & Template Handling	12.1	25.8	572.0
Output & Token Management	12.4	9.8	8.1
Resource Configuration	33.7	13.0	109.9
Text & Embedding Pipelines	8.6	18.4	25.3
Vector DB Operations	14.5	121.8	270.1

7. This rapid improvement likely reflects the community's convergence on standardized orchestration patterns. The emergence of high-level libraries has effectively codified previously erratic multi-agent behaviors into predictable, documented state machines.

Overall, the temporal analysis highlights that resolution difficulty does not evolve uniformly across topics. Some topics exhibit increasing challenge over time, while others show fluctuations that may reflect changes in community familiarity, framework updates, or feature adoption, as summarized in Table 7. Vector DB Operations and Multi-Input & Template Handling stand out as consistently high-difficulty topics within this period.

4.4. LangChain community forum analysis

To understand the nature of developer discussions within the LangChain ecosystem, we conducted an empirical analysis of posts from the official community forum, enabling us to compare these patterns against a broader general purpose developer platform. LangChain was selected due to its broad adoption and active ecosystem, as indicated by its extensive developer base (over 80 million monthly downloads) [70] and high community engagement, including more than 90,000 GitHub stars. As of March 26, 2026, the LangChain community forum contains 1277 posts across its active tags. The forum organizes discussions using a tag-based system, where tags are assigned by contributors and maintainers to reflect different aspects of developer activity.

The available tags include *python-help* (686 posts), which primarily captures implementation-related issues in Python workflows; *js-help* (147 posts), which focuses on JavaScript integrations and front-end concerns; *intro-to-langgraph* (138 posts), which covers questions related to understanding and adopting LangGraph-based agent workflows; and *cloud* (93 posts), which includes deployment, hosting, and infrastructure-related discussions. Additional tags include *self-hosted* (84 posts), which relates to running LangChain systems in custom environments; *product-feedback* (40 posts), which includes user suggestions, feature feedback, and general opinions about the framework; *langsmith-studio* (31 posts), which focuses on tooling, observability, and debugging support provided by LangSmith; *ambient-agents-with* (16 posts), which relates to experimental or emerging agent-based setups and configurations; *feature-request* (14 posts), which captures requests for new functionality or enhancements; *intro-to-langsmith* (13 posts), which includes onboarding and introductory discussions related to LangSmith tools; *hybrid* (2 posts), which involves mixed deployment or architectural setups combining multiple environments or approaches; and *guidelines* (1 post), which includes forum usage instructions and administrative guidance.

To focus on practitioner-relevant technical challenges, we excluded posts associated exclusively with tags such as *product-feedback*, *feature-request*, and *guidelines*, as these primarily relate to user feedback or forum administration rather than development issues. Since posts may be associated with multiple tags, we ensured that each post was counted only once within the retained set, resulting in a dataset of 922 unique posts.

We then analyzed this dataset using the same approach applied to the Stack Overflow data. Specifically, we used LDA to uncover latent topics by grouping posts with similar word usage patterns, and subsequently mapped these topics to the 14 predefined challenge categories. To ensure the validity of this mapping, the first two authors independently reviewed the topic outputs by examining both the top keywords and a random sample comprising 50% of posts within each category. Discrepancies were discussed and resolved through consensus, leading to the final set of topic labels and category assignments. Table 8 presents the distribution of LangChain forum posts across the challenge categories.

The distribution of posts across categories is presented in Table 8. The results indicate a clear concentration in a few dominant areas, with Framework Libraries & Methods (27.3%) and Advanced Chain/Workflow Usage (23.0%) accounting for more than half of the posts. These are followed by Agent Orchestration (10.6%), API Usage (8.35%), and Model Deployment (6.4%), while other categories appear less frequently. This pattern suggests that discussions on the LangChain forum are largely centered around framework usage, workflow construction, and coordination of agent behavior during development.

A likely explanation for this concentration is the nature of the forum itself, which is closely tied to an actively evolving framework. Developers using such platforms are often engaged in building or experimenting with applications, and therefore tend to seek guidance on composing workflows, understanding abstractions, and integrating components. As a result, discussions are more focused on usage patterns and design considerations than on lower-level system issues.

In contrast, the Stack Overflow results presented earlier show a more evenly distributed set of challenges, with prominent categories including Agent Orchestration (11%), LLM Execution & Runtime Issues (10%), Environment & Dependency Issues (10%), and Memory & Context Retrieval (10%). Compared to the LangChain forum, Stack Overflow contains a greater proportion of posts related to debugging, runtime failures, and environment configuration. This difference likely reflects the broader scope of Stack Overflow, where developers seek assistance with well-defined and reproducible issues across diverse environments.

Overall, while the LangChain forum highlights challenges related to framework usage and workflow design, Stack Overflow captures a wider range of system-level and debugging concerns. Despite these differences, both platforms consistently surface core challenges such as agent orchestration, API integration, and workflow construction, suggesting that the patterns identified in our study generalize beyond a single platform.

Table 8
Proportion of LangChain forum posts per category.

Category	Number of posts	Percentage (%)
Framework Libraries & Methods	252	27.33
Advanced Chain Usage	212	22.99
Agent Orchestration	98	10.63
API Usage	77	8.35
Model Deployment	59	6.40
Environment and Dependency Issues	55	5.97
LLM Execution & Runtime Issues	46	4.99
Output & Token Management	38	4.12
Memory & Context Retrieval	34	3.69
Vector DB Operations	17	1.84
Multi-Input & Template Handling	11	1.19
Resource Configuration	13	1.41
Text & Embedding Pipelines	7	0.76
Document Parsing	3	0.33
Total	**922**	**100**

4.5. Implications

The analysis of our study regarding developer challenges with agentic frameworks carries broader implications for how these systems are designed, adopted, and taught. By examining these challenges, we can begin to see patterns that reveal opportunities for improving tools, practices, and education.

To illustrate these patterns, Fig. 3 presents a bubble chart mapping each topic by popularity and difficulty. Each bubble represents a topic, and its size reflects how many posts fall within that category. The horizontal axis reflects difficulty, measured by the percentage of posts without accepted answers, while the vertical axis shows popularity, measured by average view counts. Together, the elements of the chart highlight which topics are widely discussed, which remain stubbornly unresolved, and where attention might be most needed. This perspective can guide the community in aligning research, practice, and education to support the sustainable growth of agentic software development.

Implications for Practitioners: Our findings, as depicted in Fig. 3, provide a roadmap for practitioners to prioritize efforts and allocate resources effectively. Fig. 3 highlights that *LLM Execution & Runtime Issues*, *Vector DB Operations*, and *Advanced Chain Usage & Asynchronous Operations* represented the most difficult topics, with a high proportion of posts lacking accepted answers. These problems are often context dependent and cannot be easily resolved through standard documentation. For example, runtime issues may stem from hardware constraints, incompatible framework versions, or subtle bugs in agent orchestration. We recommend that practitioners incorporate practical debugging utilities, standardized execution procedures, and lightweight monitoring dashboards directly into their development workflows because these issues are highly context dependent and often difficult to diagnose without targeted support. These tools can help teams quickly pinpoint runtime failures, surface configuration issues, and trace agent interactions in real time, reducing downtime and improving the reliability of deployed systems.

In contrast, *Environment & Dependency Issues* emerged as the most popular topic, reflected in their high average view counts. Practitioners frequently encountered challenges related to setting up correct runtime environments, managing package dependencies, or resolving conflicts between library versions. Interestingly, despite their popularity, these issues were comparatively less difficult than others; yet they remained significant, as 76% of posts still lacked an accepted answer. This indicates that solutions exist but are fragmented or hard to find. We recommend that practitioners create centralized, comprehensive documentation outlining setup steps, common pitfalls, and recommended configurations. They could also develop automated environment management tools, such as scripts or containerized setups that handle dependencies, version conflicts, and runtime configurations, thereby reducing manual troubleshooting.

Moreover, *Agent Orchestration* accounted for the largest volume of posts, as seen in Fig. 3, emphasizing the operational importance of multi agent coordination. To reduce repetitive troubleshooting and accelerate development cycles, practitioners could provide templates, best practice guidelines, or pre-built orchestration patterns.

Implications for Researchers: The academic community can leverage our findings to focus their research effort on the most challenging aspects of agentic frameworks. Topics such as *LLM Execution*, *Vector DB operations*, and *Advanced Chain Usage & Asynchronous Operations* remained particularly difficult in our analysis, highlighting areas where current frameworks are either insufficiently expressive or inadequately documented. These challenges suggest a need for standardized interfaces, methods to optimize runtime performance, and formal approaches to asynchronous task handling in multi-agent systems. Furthermore, the high volume of questions regarding agent orchestration indicates a gap in the available abstractions and models for managing inter-agent communication and dependencies. Future research could investigate patterns in orchestration failures, propose architectural frameworks that simplify coordination, or develop formal verification methods to ensure correctness in complex agent interactions. In particular, new architectural models could provide higher level abstractions that allow developers to specify agent goals and dependencies declaratively, rather than manually wiring message passing, state transitions, and failure handling. Such frameworks would shift orchestration from procedural logic to structured execution models. These models would automatically manage tool invocation, resource sharing, and error recovery. Together, these features would help reduce brittleness and improve reliability in deployment. Advancing research in these areas has the potential to make agentic frameworks more robust, accessible, and reliable for practitioners in real-world applications.

While the current study leverages posts authored by developers on Stack Overflow to capture real-world practitioner challenges, future work can complement this dataset with direct validation through structured surveys or interviews. Such approaches would provide additional confirmation of the findings and offer deeper insights into developer experiences. These studies would require ethical approval, participant recruitment, and careful planning, and are therefore considered as part of future research.

Implications for Educators: Educators can leverage these findings to design curricula that closely reflect the challenges of developing with agentic frameworks. The majority of questions were categorized as "How" posts, where practitioners sought step-by-step guidance to achieve specific goals. This suggests that students would benefit from experiential learning opportunities where they implement agentic frameworks themselves, rather than only studying theory. Specifically, students should gain hands on experience with frameworks such as LangChain, LlamaIndex, and AutoGen and be exposed to the common challenges practitioners face, including agent orchestration,

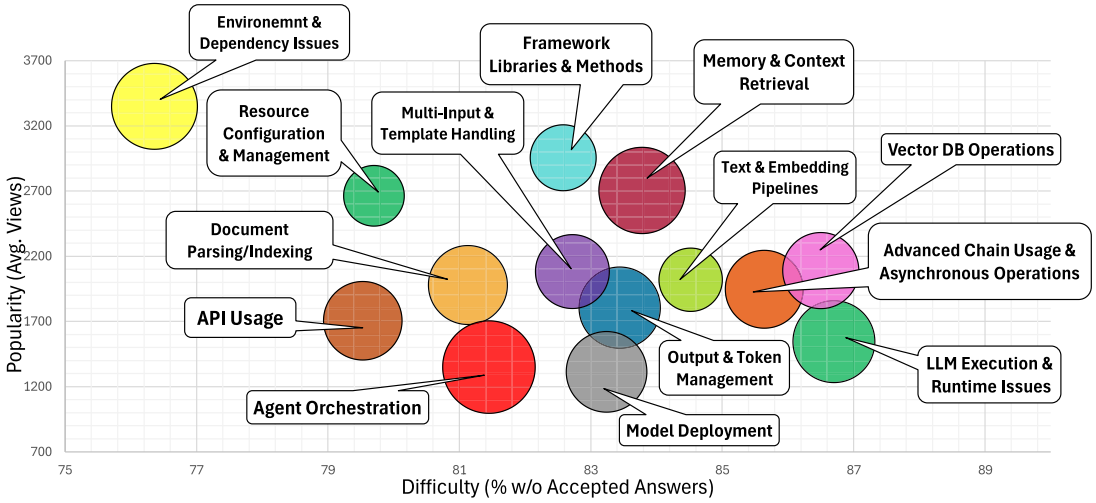


Fig. 3. Agentic framework topics' popularity vs. difficulty.

memory and context retrieval, and framework environment issues. Instruction should highlight strategies to mitigate these challenges, such as designing modular agent workflows, implementing robust memory management, and understanding framework specific constraints.

The Data Management category accounted for the largest share of posts, covering critical topics such as *Memory & Context Retrieval*, *Document Parsing/Indexing*, and *Vector DB Operations*. This indicates that managing data and context is a central challenge for practitioners. Curricula should therefore move beyond basic agent design to emphasize advanced techniques for efficient data handling, retrieval, and memory management. Similarly, the prominence of environment and dependency issues highlights the importance of teaching foundational setup and configuration practices early in coursework, ensuring that students can navigate the practical complexities of agentic framework development effectively.

5. Related work

In this section, we present the studies related to the work that leverages and analyze Stack Overflow data to have more insights from developers perspectives and discuss in Agentic AI.

5.1. Agents and agentic frameworks

An agent is an autonomous system powered by a Large Language Model (LLM) that can perceive inputs, reason about tasks, and take actions to achieve specific goals [1]. Agents interact with their environment by processing user inputs, maintaining context, and executing decisions through tools or external systems [71]. For example, an agent can receive a user query, retrieve relevant information from a database, reason about the task, and generate an appropriate response by invoking external tools or APIs.

Agentic frameworks are designed to support the development and orchestration of such agents. They provide abstractions that help developers manage the full agent lifecycle, including task decomposition, reasoning, tool execution, and coordination across multiple steps [13]. These frameworks simplify the process of building complex applications by handling common challenges such as prompt management, tool integration, and workflow control [72].

Several frameworks have emerged in this space. For example, LangChain provides modular components for chaining LLM calls, integrating external tools, and managing memory for context-aware interactions [73]. AutoGen focuses on multi-agent coordination, enabling agents to communicate, collaborate, and execute tasks through structured interaction patterns [6], while CrewAI emphasizes role-based agent orchestration through explicitly defined agent hierarchies and workflows [74]. These frameworks expose developers to different types of challenges depending on their design, such as memory management, coordination complexity, and integration-related issues.

From a technical perspective, agentic frameworks define execution and interaction patterns that govern how agents are instantiated, interact, and progress through tasks. These frameworks structure multi-step execution through control flow mechanisms, communication protocols, and lifecycle management [75]. Additionally, agentic frameworks also support integration with external systems and data sources, allowing agents to operate within complex and dynamic environments [76]. Together, these elements provide the foundation for building scalable and flexible agent-based systems.

5.2. Using stack overflow data

Prior work has examined Stack Overflow from multiple perspectives, ranging from mapping broad developer discussions and trends [77] to analyzing specific software engineering practices and methodologies [78]. In parallel, researchers have evolved topic modeling techniques moving from standard LDA to hierarchical concept modeling [79] and semantic web approaches [80], while others have tuned LDA with algorithms to better surface security questions [81] and continuous integration trends [23]. Continuing this mindset, domain specific studies have applied similar methods to particular ecosystems: Rosen and Shihab [47] used topic modeling to categorize mobile development discussions, a domain further explored by Gurcan [82] and Alanazi and Alfayez [83] specifically for the Flutter framework; Zou et al. [84,85] analyzed non functional requirements through developers' perspectives; Abdellatif et al. [20] studied chatbot development challenges; Mahmood et al. [18] examined web services topics; and Haque et al. [46] documented Docker related issues. Further granular studies include Han et al. [86] regarding Deep Learning frameworks; Abid et al. [87] surfacing refactoring topics; Ahmed and Bagherzadeh [17] on concurrency; Şilbir [88] on bioinformatics code; and Bouaziz et al. [36], who applied BERTopic to unveil microservices deployment trends.

Prior studies have examined developer challenges in emerging technologies using Stack Overflow data. In chatbot development, challenges are primarily centered around integration and development tasks, such as API usage, messaging platform integration, and intent and entity management [20]. These issues largely focus on connecting components and managing conversational logic.

In contrast, studies on challenges in developing machine learning applications highlight challenges related to data preprocessing, model deployment, and environment configuration [48]. These problems are often interdisciplinary and difficult to debug, as errors can propagate across stages. More recent Stack Overflow studies on large language models further identify limitations such as hallucinations, outdated knowledge, and difficulties handling complex frameworks and environment-specific configurations [89].

Unlike prior challenges in chatbots and machine learning systems, which primarily focus on integration, data processing, deployment, and the operationalization of single model pipelines, our findings show that agentic frameworks introduce a distinct set of challenges. These include multi-agent coordination (e.g., managing interactions and dependencies between agents), persistent memory management (e.g., maintaining and updating shared context over time), and dynamic workflow orchestration (e.g., handling non-deterministic execution paths and tool usage). These challenges arise from the system level complexity of agentic LLM applications, particularly the need to manage interactions and dependencies across multiple agents over time. Together, they introduce new coordination, state, and workflow complexities, highlighting that agentic systems require design patterns and debugging strategies not needed in chatbots or traditional machine learning applications [90,91].

To the best of our knowledge, there is no work that has studied the challenges practitioners face when using agentic frameworks. We believe that our study complements prior work on Stack Overflow by analyzing agentic related posts, systematically extracting and categorizing discussion topics surrounding these frameworks. In addition, we explore the nature of these discussions by classifying them into the "how", "what" and "why" types to better understand the underlying challenges developers face. Finally, we assess the difficulty of these topics by analyzing community responsiveness, specifically using the metrics of unresolved posts and time-to-answer. We believe that our work sheds light for the research community on the areas that require further investigation and where additional tooling, documentation, or research support could be most beneficial.

5.3. Challenges in agentic AI

Research into agentic AI systems has uncovered significant challenges ranging from foundational ethical concerns to intricate engineering limitations inherent to agents. Hughes et al. [92] and Raza et al. [93] emphasize the critical need for robust governance frameworks to address attribution, accountability, and security vulnerabilities specifically within agent decision making processes, while Ranjan et al. [94] highlight the systemic risks of emergent bias and fairness propagation unique to multi agent systems. On the architectural front, Derouiche et al. [13] and Mascardi et al. [95] identify scalability, memory management for long term agent operation, and the lack of standardized inter agent communication protocols as primary barriers to engineering robust multi agent systems. These structural challenges are compounded by operational complexities; for instance, Epperson et al. [96] note the severe lack of tool support for reviewing and debugging long, complex inter agent conversations, and Krishnan [97] points to context management across agents as a key bottleneck. Furthermore, the dynamic nature of agent interaction introduces unique risks: Tian et al. [98] and Hammond et al. [99] warn of miscoordination, collusion, and unforeseen adversarial behaviors arising between agents, while Han et al. [90] and Tran et al. [100] focus on the difficulties of optimizing task allocation among agents and fostering effective reasoning through agent debates. Finally, bridging the gap between autonomous agents and human users remains a distinct challenge, with Gal and Grosz [101] and Naik et al. [102] stressing the importance of designing for effective mixed human–agent collaboration and maintaining appropriate oversight to balance agent autonomy with human control.

Unlike prior work that focuses predominantly on the execution phases of agentic systems and analyzes deployed systems to understand their limitations, our study is the first that primarily investigates the challenges practitioners face before deployment. We examine real-world issues derived from Stack Overflow discussions of agentic frameworks. This pre-deployment perspective is important because it highlights obstacles that may hinder system construction and preparation for deployment, which are often overlooked in studies that focus only on post-deployment behavior.

6. Threats to validity

6.1. Internal validity

Our study relies on analyzing posts and discussions about agentic frameworks, which introduces the possibility that some relevant content may have been overlooked or miscategorized. Prior software engineering studies on Stack Overflow have adopted tag-based filtering as a practical and effective approach for constructing focused datasets [20–23]. These tags serve as a primary indicator of a post's technical domain and are widely relied upon by practitioners to categorize and discover relevant content. While tag-based filtering ensures a manageable and focused dataset, it may exclude posts that use unconventional or emerging terminology not captured by the selected tags. Stack Overflow contains more than 60 million posts and approximately 80 million comments, making exhaustive analysis computationally expensive in terms of time and resources. To mitigate potential omissions, we followed a multi-step approach. First, we systematically derived an initial set of general tags by requiring the co-occurrence of *large-language-model* and *agent*. This combination was chosen for its high conceptual specificity to agentic LLM systems: using the tag *large-language-model* alone would introduce noise from general LLM usage (e.g., prompt engineering), while *agent* alone often refers to unrelated software agents. Specific framework tags (e.g., LangChain or CrewAI) were considered, but they risked missing emerging frameworks and limiting the analysis to a single ecosystem. All authors discussed these alternatives and reached consensus on using the co-occurrence of both tags as the base for dataset construction. From this starting set, we extracted all co-occurring tags and applied a hybrid approach combining quantitative thresholds (TRT and TST) with manual qualitative validation. Specifically, the first two authors reviewed samples of posts for each candidate tag to ensure relevance to agentic frameworks and reduce noise from unrelated topics. This process balances precision and coverage while maintaining a focused and reliable dataset. Future work can further complement this approach with semantic or embedding retrieval methods to capture a broader range of relevant discussions.

Additionally, determining the optimal number of clusters for topic modeling can influence the granularity of the topics extracted. We addressed this by testing multiple cluster sizes and evaluating their coherence to ensure that the resulting topics meaningfully represented discussions on agentic frameworks. Finally, labeling posts according to type is an inherently subjective process. To mitigate this, multiple authors independently classified the data, and interrater reliability was measured to ensure consistency across annotations.

6.2. Construct validity

Construct validity concerns whether the variables we measured truly reflected the challenges practitioners face with agentic frameworks. Topic labels, for instance, may not perfectly capture the nuanced content of the posts. To minimize this issue, several authors independently reviewed keywords and representative posts for each topic, followed by group discussions to reach agreement on meaningful labels. Metrics used to assess the complexity and popularity of issues were chosen based on prior research in software engineering, which helped ensure that they aligned with the theoretical constructs of developer challenges.

6.3. External validity

Threats to external validity involve the extent to which our findings hold in other contexts. The generalizability of our findings is limited by the focus on a single platform. While other forums and developer communities, such as GitHub Discussions and Discord, may provide additional insights into agentic frameworks, we chose Stack Overflow because it attracts a wide range of practitioners and continues to host discussions on complex, high difficulty issues that go beyond what current LLM-based assistants can reliably address. Stack Overflow hosts over 24 million questions and remains widely used with 31 million registered users [103].

To mitigate this limitation, we complemented our analysis with an additional study of the LangChain community forum, which showed consistent patterns in the types of challenges developers encounter, as discussed in Section 4. Furthermore, we plan in the future to extend our study by incorporating multiple platforms, such as GitHub Discussions, Discord communities, and specialized forums, or by conducting surveys and interviews to capture a more comprehensive view of developer challenges. Such work could also explore how LLM-based assistance and community-driven platforms complement each other in practice.

7. Conclusion

In this study, we analyzed Stack Overflow posts to uncover the challenges practitioners face when working with agentic frameworks. We identified 14 topics of discussion across four main categories: `Configuration`, `Development`, `Data Management`, and `Integration`. Practitioners showed the greatest interest in questions about setting up the framework and managing its dependencies, while working with vector databases proved to be the most difficult topic. Furthermore, our analysis of question types revealed that a majority of posts (71.1%) were procedural "How" questions, underscoring the strong developer need for practical, step-by-step guidance. These challenges are not typical of traditional software development but arise from the unique complexities of agentic frameworks, where coordinating multiple agents and managing large, dynamic contexts introduce novel difficulties. Hence, there is a strong need for practical support, such as standardized execution workflows, debugging aids, and tools to simplify orchestration and data management, and tailored educational resources that provide practitioners with practical experience using these frameworks. Addressing these gaps would not only ease the learning curve for newcomers but also enable experienced practitioners to build more efficient and resilient agentic systems. Overall, our findings highlight the areas where the community can focus efforts to make agentic frameworks easier to adopt and more effective in practice. Future research could explore designing higher-level abstractions for agent orchestration, formal verification techniques to ensure correct interactions, and optimized methods for managing runtime execution and vector database operations. Investigating these areas could lead to frameworks that are more robust, less error-prone, and easier for practitioners to deploy effectively in complex, real-world scenarios.

CRediT authorship contribution statement

Rinad Hamid: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **John Pangas:** Writing – review & editing, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ahmad Abdellatif:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Rinad Hamid reports was provided by University of Calgary. 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.

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Z. Chen, J. Tang, X. Chen, Y. Lin, A survey on large language model based autonomous agents, 2023, arXiv.

[2] W. Huanting, G. Jingzhi, Z. Huawei, X. Jie, W. Zheng, AI agentic programming: A survey of techniques, challenges, and opportunities, 2025, http://dx.doi.org/10.48550/arXiv.2508.11126, arXiv preprint arXiv:2508.11126.

[3] Y. Wang, W. Zhong, Y. Huang, E. Shi, M. Yang, J. Chen, H. Li, Y. Ma, Q. Wang, Z. Zheng, Agents in software engineering: Survey, landscape, and vision, Autom. Softw. Eng. 32 (2) (2025) 1–36.

[4] Precedence Research, Agentic AI Market Grows as Autonomous AI Agents Redefine Productivity and Task Automation, 2025, https://www.precedenceresearch.com/agentic-ai-market.

[5] J. He, C. Treude, D. Lo, LLM-based multi-agent systems for software engineering: Literature review, vision, and the road ahead, ACM Trans. Softw. Eng. Methodol. 34 (5) (2025) http://dx.doi.org/10.1145/3712003.

[6] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, et al., Autogen: Enabling next-gen LLM applications via multi-agent conversations, in: First Conference on Language Modeling, 2024.

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

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

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

[10] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, et al., Agentbench: Evaluating llms as agents, 2023, arXiv preprint arXiv:2308.03688.

[11] Z. Liu, W. Yao, J. Zhang, L. Xue, S. Heinecke, R. Murthy, Y. Feng, Z. Chen, J.C. Niebles, D. Arpit, et al., Bolaa: Benchmarking and orchestrating llm-augmented autonomous agents, 2023, arXiv preprint arXiv:2308.05960.

[12] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, et al., Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023, arXiv preprint arXiv:2307.16789.

[13] H. Derouiche, Z. Brahmi, H. Mazeni, Agentic ai frameworks: Architectures, protocols, and design challenges, 2025, arXiv preprint arXiv:2508.10146.

[14] A. Fourney, A. Awadallah, C. Tan, E. Zhu, F. Niedtner, G. Bansal, J. Gerrits, J. Alber, et al., AutoGen v0.4: Reimagining the foundation of agentic AI for scale, extensibility, and robustness, 2025, https://www.microsoft.com/en-us/research/blog/autogen-v0-4-reimagining-the-foundation-of-agentic-ai-for-scale-extensibility-and-robustness/ . (Accessed 22 November 2025).

[15] F. Both, Why we no longer use LangChain for building our AI agents, 2024, Octomind Engineering Blog. https://octomind.dev/blog/why-we-no-longer-use-langchain-for-building-our-ai-agents. (Accessed 22 November 2025).

[16] X. Chen, J. Wang, C. Gao, X. Ju, Z. Cui, An empirical study of openai API discussions on stack overflow, 2025, http://dx.doi.org/10.48550/arXiv.2505.04084, arXiv preprint arXiv:2505.04084.

[17] S. Ahmed, M. Bagherzadeh, 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, ESEM'18, ACM, 2018, http://dx.doi.org/10.1145/3239235.3239524.

[18] K. Mahmood, G. Rasool, F. Sabir, A. Athar, An empirical study of web services topics in web developer discussions on stack overflow, IEEE Access 11 (2023) 9627–9655, http://dx.doi.org/10.1109/ACCESS.2023.3238813.

[19] C.E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, K. Narasimhan, SWE-bench: Can language models resolve real-world GitHub issues? in: The Twelfth International Conference on Learning Representations, 2024.

[20] A. Abdellatif, D. Costa, K. Badran, R. Abdalkareem, E. Shihab, Challenges in chatbot development: A study of stack overflow posts, in: 2020 IEEE/ACM 17th International Conference on Mining Software Repositories, MSR, 2020, pp. 174–185, http://dx.doi.org/10.1145/3379597.3387472.

[21] M. Bagherzadeh, R. Khatchadourian, 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 (ESEC/FSE 2019), Association for Computing Machinery, New York, NY, USA, 2019, pp. 432–442, http://dx.doi.org/10.1145/3338906.3338939.

[22] G. Uddin, F. Sabir, Y.G. Guéhéneuc, An empirical study of IoT topics in IoT developer discussions on stack overflow, Empir. Softw. Eng. 26 (2021) http://dx.doi.org/10.1007/s10664-021-10021-5.

[23] O. Ali, S. Islem, A. Eman, W.M. Mohamed, An empirical study on continuous integration trends, topics and challenges in stack overflow, in: Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering, EASE'23, ACM, 2023, pp. 141–151, http://dx.doi.org/10.1145/3593434.3593485.

[24] Natural Language Toolkit (NLTK), Natural language toolkit - NLTK 3.9.2 documentation, 2025, https://www.nltk.org/. (Accessed on 29 July 2025).

[25] Gensim, Gensim: Topic modelling for humans, 2025, https://radimrehurek.com/gensim/. (Accessed on 27 July 2025).

[26] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet Allocation, J. Mach. Learn. Res. 3 (March 2003) 993–1022, http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993.

[27] M.A. Al Alamin, S. Malakar, G. Uddin, S. Afroz, T.B. Haider, A. Iqbal, An empirical study of developer discussions on low-code software development challenges, in: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories, MSR, 2021, pp. 46–57, http://dx.doi.org/10.1109/MSR52588.2021.00018.

[28] A.M. Alghamdi, W. Aljedaani, H. Jalali, et al., Understanding developer challenges and trends in web accessibility: a Stack Overflow analysis, Univ. Access Inform. Soc. 24 (2024) 1701–1717, http://dx.doi.org/10.1007/s10209-024-01174-3.

[29] Andrew Kachites McCallum, MALLET: A machine learning for language toolkit, 2002, http://mallet.cs.umass.edu/. (Accessed 27 July 2025).

[30] M. Röder, A. Both, A. Hinneburg, Exploring the space of topic coherence measures, in: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM'15, 2015, pp. 399–408, http://dx.doi.org/10.1145/2684822.2685324.

[31] F. Angela, G. Beliz, H. Mark, L. Mitya, S. Shubho, Y. Shin, M.Z. Jie, Large language models for software engineering: Survey and open problems, in: International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE 2023), IEEE, 2023, pp. 31–53.

[32] A. Novikov, N. Vu, M. Eisenberger, E. Dupont, P.-S. Huang, A.Z. Wagner, S. Shirobokov, B. Kozlovskii, F.J.R. Ruiz, A. Mehrabian, et al., AlphaEvolve: A coding agent for scientific and algorithmic discovery, 2025, http://dx.doi.org/10.48550/arXiv.2506.13131, arXiv preprint arXiv:2506.13131.

[33] M.C. Ramos, C.J. Collison, A.D. White, A review of large language models and autonomous agents in chemistry, Chem. Sci. (2025).

[34] Y. Xiao, E. Sun, D. Luo, W. Wang, TradingAgents: Multi-agents LLM financial trading framework, 2024, arXiv preprint arXiv:2412.20138.

[35] E. Sesari, F. Sarro, A. Rastogi, Understanding fairness in software engineering: Insights from stack exchange sites, in: Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2024, pp. 269–280.

[36] A. Bouaziz, M.A. Saied, M. Sayagh, A. Ouni, M.W. Mkaouer, An empirical study on microservices deployment trends, topics and challenges in stack overflow, in: 2025 IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER, IEEE, 2025, pp. 113–123.

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

[38] A. Bency, How to use VectorStoreRetrieverMemory in langchain with pgvector?, 2023, Available at URL https://stackoverflow.com/q/76968299. (Accessed 21 November 2025).

[39] watermelodic, How to apply a prompt on each retrieved document before answering, in the langchain ConversationalRetrievalChain, 2023, Available at URL https://stackoverflow.com/q/77043572. (Accessed 21 November 2025).

[40] questioner9928, Langchain returning answers out of context, 2024, Available at URL https://stackoverflow.com/q/78942031. (Accessed 21 November 2025).

[41] user20777937, Authentication error when using LangChain agents, 2024, Available at URL https://stackoverflow.com/q/77742697. (Accessed 21 November 2025).

[42] Karthik, LangChain ModuleNotFoundError: No module named 'langchain', 2023, Available at URL https://stackoverflow.com/q/76726419. (Accessed 21 November 2025).

[43] user13133644, LlamaIndex SIGILL on mac M1, 2023, Available at URL https://stackoverflow.com/q/76259887. (Accessed 21 November 2025).

[44] K. Alam, K. Mittal, B. Roy, C. Roy, Developer challenges on large language models: A study of stack overflow and openai developer forum posts, in: CoRR Abs/2411.10873, 2024, http://dx.doi.org/10.48550/arXiv.2411.10873.

[45] M. Ahmed, M.N.I. Opu, C. Roy, S.I. Suhi, S. Chowdhury, Exploring challenges in test mocking: Developer questions and insights from stack overflow, 2025, http://dx.doi.org/10.48550/ARXIV.2505.08300, CoRR. arXiv:2505.08300.

[46] M.U. Haque, L.H. Iwaya, M.A. Babar, 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, 2020, pp. 1–11, http://dx.doi.org/10.1145/3382494.3410683.

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

[48] M. Alshangiti, H. Sapkota, P.K. Murukannaiah, X. Liu, Q. Yu, Why is developing machine learning applications challenging? A study on stack overflow posts, in: ESEM, 2019, pp. 1–11.

[49] M. Asaduzzaman, A.S. Mashiyat, C.K. Roy, K.A. Schneider, Answering questions about unanswered questions of stack overflow, in: Proceedings of the 10th Working Conference on Mining Software Repositories (MSR 2013), 2013, pp. 97–100.

[50] B. Wang, D. Zhao, N.R. Tallent, L. Guo, On the reproducibility limitations of RAG systems, 2025, arXiv:2509.18869.

[51] DigitalOcean, Beyond vector databases: RAG architectures without embeddings, 2025, https://www.digitalocean.com/community/tutorials/beyond-vector-databases-rag-without-embeddings.

[52] User77719186, How to create isolated session for ConversationBufferMemory per user in langchain, 2024, Available at URL https://stackoverflow.com/q/77719186. (Accessed 22 November 2025).

[53] User78097971, Langchain: How to store memory with streaming, 2024, Available at URL https://stackoverflow.com/q/78097971. (Accessed 22 November 2025).

[54] Z. Xiong, Y. Lin, W. Xie, P. He, Z. Liu, J. Tang, H. Lakkaraju, Z. Xiang, How memory management impacts LLM agents: An empirical study of experience-following behavior, 2025, arXiv:2505.16067.

[55] Y. Hu, Y. Wang, J. McAuley, Evaluating memory in LLM agents via incremental multi-turn interactions, 2025, http://dx.doi.org/10.48550/arxiv.2507.05257, ArXiv. arXiv:2507.05257.

[56] Z. Zhang, X. Bo, C. Ma, R. Li, X. Chen, Q. Dai, J. Zhu, Z. Dong, J.-R. Wen, A survey on the memory mechanism of large language model based agents, 2024, http://dx.doi.org/10.48550/arxiv.2404.13501, ArXiv. arXiv:2404.13501.

[57] Y. Wang, X. Chen, MIRIX: Multi-agent memory system for LLM-based agents, 2025, http://dx.doi.org/10.48550/arxiv.2507.07957, ArXiv. arXiv:2507.07957.

[58] D. Helic, T. Santos, Stack overflow is not dead yet: Crowd answers still matter, 2025, http://dx.doi.org/10.48550/arXiv.2509.05879, arXiv Preprint arXiv:2509.05879 [Cs.CY].

[59] J. Liu, X. Tang, L. Li, P. Chen, Y. Liu, ChatGPT vs. Stack overflow: An exploratory comparison of programming assistance tools, in: 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security Companion (QRS-C), 2023, pp. 364–373, http://dx.doi.org/10.1109/QRS-C60940.2023.00105.

[60] L. Zhong, Z. Wang, Can LLM replace stack overflow? A study on the robustness and reliability of large language model code generation, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, (19) 2024, pp. 21841–21849.

[61] S. Kabir, D.N. Udo-Imeh, B. Kou, T. Zhang, Is stack overflow obsolete? An empirical study of the characteristics of ChatGPT answers to stack overflow questions, in: Conference on Human Factors in Computing Systems, CHI'24, ACM, 2024, http://dx.doi.org/10.1145/3613904.3642596.

[62] Stack Overflow, 2025 stack overflow developer survey, 2025, URL https://survey.stackoverflow.co/2025/. (Accessed 26 March 2026).

[63] A. Fourney, A. Awadallah, C. Tan, E. Zhu, F. Niedtner, G. Bansal, J. Gerrits, J. Alber, et al., AutoGen v0.4: Reimagining the foundation of agentic AI for scale, extensibility, and robustness, 2025, URL https://www.microsoft.com/en-us/research/blog/autogen-v0-4-reimagining-the-foundation-of-agentic-ai-for-scale-extensibility-and-robustness/. (Accessed 22 November 2025).

[64] F. Both, Why we no longer use LangChain for building our AI agents, 2024, URL https://octomind.dev/blog/why-we-no-longer-use-langchain-for-building-our-ai-agents. (Accessed 22 November 2025).

[65] E. Dritsas, M. Trigka, Machine learning in modern database systems: Techniques, architectures, and deployment challenges, Eng. Appl. Artif. Intell. 170 (2026) 114225.

[66] H. Vejendla, Drift-adapter: A practical approach to near zero-downtime embedding model upgrades in vector databases, in: Proceedings of EMNLP 2025, 2025, pp. 15938–15949.

[67] D. Chen, et al., HaluMem: Evaluating hallucinations in memory systems of agents, 2026, ArXiv Preprint.

[68] N.F. Liu, K. Lin, J. Hewitt, et al., Lost in the middle: How language models use long contexts, 2023, ArXiv. URL https://arxiv.org/abs/2307.03172.

[69] S. Barnett, S. Kurniawan, S. Thudumu, et al., Seven failure points when engineering a retrieval augmented generation system, in: CAIN 2024, 2024, pp. 194–199, http://dx.doi.org/10.1145/3644815.3644945.
[70] H. Chase, Reflections on three years of building LangChain, 2025, https://blog.langchain.com/three-years-langchain/. (Accessed 26 March 2026).
[71] W. Zhang, et al., AgentOrchestra: Orchestrating multi-agent intelligence with the tool-environment-agent (TEA) protocol, 2025.
[72] Y. Wang, et al., An empirical study of agent developer practices in AI agent frameworks, 2025, arXiv preprint arXiv:2512.01939.
[73] LangChain, LangChain documentation, 2026, URL https://docs.langchain.com/.
[74] CrewAI, Crewai documentation, 2026, URL https://docs.crewai.com/. (Accessed 26 March 2026).
[75] Z. Durante, et al., Agent AI: Surveying the horizons of multimodal interaction, 2024, http://dx.doi.org/10.48550/arxiv.2401.03568, arXiv preprint arXiv:2401.03568.
[76] R. Sapkota, K.I. Roumeliotis, M. Karkee, AI agents vs. Agentic AI: A conceptual taxonomy, applications and challenges, Inf. Fusion 126 (2026) 103599, http://dx.doi.org/10.1016/j.inffus.2025.103599.
[77] A. Barua, S.W. Thomas, A.E. Hassan, What are developers talking about? an analysis of topics and trends in stack overflow, Empir. Softw. Eng. 19 (3) (2014) 619–654.
[78] A.A. Khan, J.A. Khan, M.A. Akbar, P. Zhou, M. Fahmideh, Insights into software development approaches: mining q &a repositories, Empir. Softw. Eng. 29 (1) (2024) 8.
[79] H. Chen, J. Coogle, K. Damevski, Modeling stack overflow tags and topics as a hierarchy of concepts, J. Syst. Softw. 156 (2019) 283–299.
[80] Z. Anwar, H. Afzal, S. Kadry, X. Cheng, Semantic web approaches in stack overflow: Research trends and technological insights, Int. J. Semant. Web Inf. Syst. (IJSWIS) 20 (1) (2024) 1–61.
[81] X. Yang, D. Lo, X. Xia, Z. Wan, J. Sun, What security questions do developers ask? a large-scale study of stack overflow posts, J. Comput. Sci. Tech. 31 (5) (2016) 910–924.
[82] F. Gurcan, Identification of mobile development issues using semantic topic modeling of stack overflow posts, PeerJ Comput. Sci. 9 (2023) e1658.
[83] A. Alanazi, R. Alfayez, What is discussed about flutter on stack overflow (SO) question-and-answer (q&a) website: An empirical study, J. Syst. Softw. 215 (2024) 112089.
[84] J. Zou, L. Xu, M. Yang, X. Zhang, D. Yang, Towards comprehending the non-functional requirements through developers' eyes: An exploration of stack overflow using topic analysis, Inf. Softw. Technol. 84 (2017) 19–32.
[85] J. Zou, L. Xu, W. Guo, M. Yan, D. Yang, X. Zhang, Which non-functional requirements do developers focus on? an empirical study on stack overflow using topic analysis, in: 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, IEEE, 2015, pp. 446–449.
[86] J. Han, E. Shihab, Z. Wan, A.E. Hassan, What do programmers discuss about deep learning frameworks? Empir. Softw. Eng. 25 (2020) 2694–2747, http://dx.doi.org/10.1007/s10664-020-09819-6.
[87] C. Abid, K. Gaaloul, M. Kessentini, V. Alizadeh, What refactoring topics do developers discuss? a large scale empirical study using stack overflow, IEEE Access 10 (2021) 56362–56374.
[88] G.M. Şilbir, Evaluation of posts by bioinformatics code developers on stack overflow platform: topic modeling and community detection, Politek. Derg. 1–1.
[89] L. Da Silva, J. Samhi, F. Khomh, LLMs and stack overflow discussions: Reliability, impact, and challenges, J. Syst. Softw. 230 (2025) 112541.
[90] S. Han, Q. Zhang, Y. Yao, W. Jin, Z. Xu, LLM multi-agent systems: Challenges and open problems, 2025, arXiv:2402.03578.
[91] S.S. Chowa, et al., From language to action: A review of large language models as autonomous agents and tool users, Artif. Intell. Rev. 59 (2) (2026) 1–59.
[92] L. Hughes, Y.K. Dwivedi, T. Malik, M. Shawosh, M.A. Albashrawi, et al., AI agents and agentic systems: A multi-expert analysis, J. Comput. Inf. Syst. (2025) 1–29.
[93] S. Raza, R. Sapkota, M. Karkee, C. Emmanouilidis, Trism for agentic AI: A review of trust, risk, and security management in LLM-based agentic multi-agent systems, 2025, arXiv:2506.04133.
[94] R. Ranjan, S. Gupta, S.N. Singh, Fairness in agentic AI: A unified framework for ethical and equitable multi-agent system, 2025, arXiv:2502.07254.
[95] V. Mascardi, D. Weyns, A. Ricci, C.B. Earle, A. Casals, M. Challenger, et al., Engineering multi-agent systems: State of affairs and the road ahead, SIGSOFT Softw. Eng. Notes 44 (1) (2020) 18–28, http://dx.doi.org/10.1145/3310013.3322175.
[96] W. Epperson, G. Bansal, V.C. Dibia, A. Fourney, J. Gerrits, E.E. Zhu, S. Amershi, Interactive debugging and steering of multi-agent AI systems, in: Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, CHI '25, Association for Computing Machinery, New York, NY, USA, 2025, http://dx.doi.org/10.1145/3706598.3713581.
[97] N. Krishnan, Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications, 2025, arXiv:2504.21030.
[98] F. Tian, A. Luo, J. Du, X. Xian, R. Specht, G. Wang, X. Bi, J. Zhou, et al., An outlook on the opportunities and challenges of multi-agent ai systems, 2025, arXiv:2505.18397.
[99] L. Hammond, A. Chan, J. Clifton, J. Hoelscher-Obermaier, A. Khan, E. McLean, C. Smith, et al., Multi-agent risks from advanced AI, 2025, arXiv:2502.14143.
[100] K.-T. Tran, D. Dao, M.-D. Nguyen, Q.-V. Pham, B. O'Sullivan, H.D. Nguyen, Multi-agent collaboration mechanisms: A survey of LLMs, 2025, arXiv:2501.06322.
[101] K. Gal, B.J. Grosz, Multi-agent systems: Technical & ethical challenges of functioning in a mixed group, Daedalus 151 (2) (2022) 114–126.
[102] S. Naik, A. Snellinger, A.L. Toombs, S. Saponas, A.K. Hall, Exploring early adopters' use of AI driven multi-agent systems to inform human-agent interaction design: Insights from industry practice, in: CHI EA '25, Association for Computing Machinery, New York, NY, USA, 2025, http://dx.doi.org/10.1145/3706599.3706693.
[103] Stack Exchange Network, Stack exchange sites list, 2024, URL https://stackexchange.com/sites. (Accessed 2024).