

# A Large-Scale Study on the Development and Issues of Multi-Agent AI Systems

Daniel Liu\*, Krishna Upadhyay\*, Vinaik Chhetri\*, A.B. Siddique†, Umar Farooq\*

\* Louisiana State University, † University of Kentucky

Email: dliu26@lsu.edu, kupadh4@lsu.edu, vchhet2@lsu.edu, ab.siddique@uky.edu, ufarooq@lsu.edu

**Abstract**—The rapid emergence of multi-agent AI systems (MAS), including LangChain, CrewAI, and AutoGen, has shaped how large language model (LLM) applications are developed and orchestrated. However, little is known about how these systems evolve and are maintained in practice. This paper presents the first large-scale empirical study of open-source MAS, analyzing over 42K unique commits and over 4.7K resolved issues across eight leading systems. Our analysis identifies three distinct development profiles: sustained, steady, and burst-driven. These profiles reflect substantial variation in ecosystem maturity. Perfective commits constitute 40.8% of all changes, suggesting that feature enhancement is prioritized over corrective maintenance (27.4%) and adaptive updates (24.3%). Data about issues shows that the most frequent concerns involve bugs (22%), infrastructure (14%), and agent coordination challenges (10%). Issue reporting also increased sharply across all frameworks starting in 2023. Median resolution times range from under one day to about two weeks, with distributions skewed toward fast responses but a minority of issues requiring extended attention. These results highlight both the momentum and the fragility of the current ecosystem, emphasizing the need for improved testing infrastructure, documentation quality, and maintenance practices to ensure long-term reliability and sustainability.

**Index Terms**—Multi-Agent Software, Software Repositories, Software Mining, Software Maintenance.

## I. INTRODUCTION

The emergence of large language models (LLMs) has significantly influenced the design of intelligent systems, giving rise to multi-agent systems (MAS) that support collaboration among autonomous agents on complex tasks. Frameworks such as AutoGen [22], CrewAI [3], and LangChain [11] provide abstractions for agent creation, communication, and coordination. These systems support the construction of workflows that integrate reasoning, planning, and tool use, making them a central component of modern LLM-based software development.

Despite their growing adoption, little is known about how these systems evolve and are maintained. Prior work on MAS has focused primarily on algorithmic and architectural advances, while the empirical understanding of their software development practices remains limited. This gap is critical because these systems must evolve rapidly to support new models, APIs, and orchestration mechanisms, while maintaining stability and long-term sustainability.

In software engineering, mining software repositories has proven effective for identifying development and maintenance trends at scale. However, such analyses have not yet been applied to multi-agent AI systems, which involve a mix of

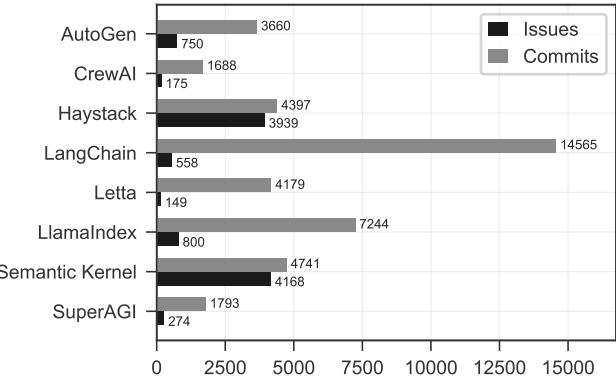


Fig. 1: Development and maintenance activity across eight major multi-agent AI systems, illustrating the commits and issues used in our large-scale study.

neural and symbolic reasoning, distributed execution, and frequent integration with external services. Understanding how these systems evolve and how issues are handled in practice can provide valuable insights into the engineering of complex AI systems.

To address this gap, we conduct a large-scale empirical study of the most active open-source multi-agent AI systems on GitHub (ranging from 16K to 119K stars). Our dataset includes eight representative projects: AutoGen [22], CrewAI [3], Haystack [8], LangChain [11], Letta [12], LlamaIndex [14], Semantic Kernel [9], and SuperAGI [21]. Table I provides their architecture and usage details. From these repositories, we collect 42,267 unique commits and 4,731 resolved issues, enabling a detailed examination of development patterns and maintenance activities.

This study is structured around two core research questions (RQs). The first asks how development patterns differ across MAS, aiming to characterize variations in activity levels, growth trajectories, and commit types. The second investigates issue reporting and resolution, focusing on the types of issues encountered, their frequency, and how efficiently they are addressed across projects. Together, these RQs provide a comprehensive view of the development and maintenance practices within the multi-agent AI ecosystem.

As shown in Figure 1, the selected repositories vary significantly in both development activity and issue volume. LangChain exhibits the highest development intensity with over 14K commits, followed by LlamaIndex and Semantic Kernel with approximately 7K and 4.7K commits, respec-

TABLE I: Overview of the eight multi-agent AI systems analyzed in this study, including their architecture types, primary purposes, and GitHub popularity.

MAS	Architecture	Purpose	GitHub Stars
Autogen [22]	Conversational Workflow	Multi-agent collaboration through automated conversational workflows with support for sequential, group, and nested chat patterns	51.3K
CrewAI [3]	Role-based Hierarchy	Orchestrating autonomous agents with defined roles working collaboratively on tasks through sequential or manager-led execution	40K
Haystack [8]	Graph-based Pipeline	Building Retrieval-Augmented Generation (RAG) applications, document search, question-answering, and agentic systems with modular, composable components	23.2K
LangChain [11]	Modular Chain	General-purpose LLM orchestration enabling chains, agents, and tool integrations through composable building blocks	119K
Letta [12]	Memory-centric Agent	Building stateful AI agents with persistent memory, enabling long-term context retention and personalized interactions across conversations	19K
Llama Index [14]	Data-centric Pipeline	Context-augmented LLM applications focusing on connecting LLMs to external data sources through RAG patterns	45K
Semantic Kernel [9]	Plugin-based Orchestrator	Enterprise AI integration bridging LLMs with existing code through plugins, enabling AI orchestration in business applications	26.6K
SuperAGI [21]	Extensible Agent Toolkit	Building, deploying, and managing production-ready autonomous AI agents with concurrent execution and tool integration capabilities	16.8K

tively. Haystack and AutoGen demonstrate steady, long-term activity, while CrewAI, Letta, and SuperAGI contribute at smaller but consistent scales. On the issue side, Semantic Kernel and Haystack each report more than 4K issues, indicating large and active developer communities. This variation highlights a maturing ecosystem in which a few MAS account for most of the development and maintenance work.

Our findings identify three development profiles. *Sustained* systems like Haystack show consistent growth and periodic refactoring. *Steady* ones such as Semantic Kernel and LlamaIndex maintain balanced activity with moderate fluctuation. *Burst-driven*, including SuperAGI, undergo intense but short-lived development. LangChain stands out with over 14K commits, reflecting its central influence. These profiles reflect differing maturity, contributor engagement, and stability.

Commit-level analysis shows that perfective commits dominate (40.8%), indicating a strong focus on feature enhancement over corrective (27.4%) and adaptive (24.3%) changes. Code churn patterns since 2023 show increasing deletions matching insertions, signaling a shift from expansion to architectural refinement. Issue analysis reveals that bugs (22%), infrastructure (14%), and agent coordination (10%) are the most frequent concerns, highlighting the complexity of orchestrating modular agents. Median resolution times range from under a day to two weeks, but high variance points to uneven responsiveness and maintenance capacity.

This work provides the first quantitative analysis of the development and maintenance of multi-agent AI systems. It is characterized by fast iteration, feature-oriented development, and uneven maturity across systems. Together, these findings highlight both the momentum and fragility of the ecosystem. To support future research in this domain, we release our curated dataset.

## II. BACKGROUND AND RELATED WORK

### A. Multi-Agent AI System

The rise of LLMs has fundamentally changed how AI systems are designed and built. Instead of relying on a single, all-

purpose model, developers are increasingly turning to multi-agent architectures, where multiple specialized agents work together to solve complex problems. Each agent has its own role, capabilities, and objectives, and they collaborate through structured communication and coordination. This shift allows systems to reason more effectively and distribute work across agents, mimicking teamwork in human organizations.

Several open-source frameworks have emerged to make the construction of such MAS more accessible. AutoGen [22] provides an environment for building conversational agents that engage in multi-turn dialogues, enabling them to negotiate, plan, and cooperate to achieve shared objectives. CrewAI [3] takes inspiration from human organizations by modeling agents as members of a hierarchical “crew”, each with specific roles and responsibilities, facilitating structured collaboration and delegation. LangChain [11] offers composable building blocks or “chains”, that allow developers to design complex reasoning workflows, while LlamaIndex [14] focuses on augmenting agents with data retrieval capabilities, enabling them to access and reason over external knowledge sources efficiently. Microsoft’s Semantic Kernel [9] introduces an enterprise-grade orchestration layer that blends symbolic and neural reasoning. It enables developers to define reusable “skills” and connect LLM-based reasoning with conventional code execution. Together, these frameworks illustrate the growing sophistication of modern AI development, where systems are not just powered by a single intelligent model but by networks of coordinated agents that interact dynamically to achieve goals.

Despite differences in design philosophy, these frameworks share common architectural foundations. Most provide abstractions for agent definition (roles, personas, capabilities), inter-agent communication (message passing, shared memory, context sharing), task decomposition (planning and delegation), and execution control (sequential, parallel, or conditional orchestration). The choice of framework typically reflects trade-offs between flexibility, simplicity, and coordination complexity, balancing how much autonomy individual agents

have against how tightly their interactions are managed.

### B. Related Work

In software engineering, software repository mining has long been a crucial method for understanding how software is built and maintained at scale. A study on language adoption trends [19] have shown how ecosystem choices influence code quality. Bug predication research has leveraged historical data to identify defect-prone components [10], [18]. Eyolfson et al. [4] revealed that even the timing of code commits, such as late-night activity, can impact software reliability. Domain-specific analyses have provided further insights, identifying recurring problems such as API misuse in mobile apps [13], vulnerability patterns in blockchain-based smart contracts [17], and documentation drift detected through commit message analysis [1].

Building on this tradition, recent work on multi-agent AI systems has focused largely on their architectural and algorithmic innovations. Surveys such as those by Guo et al. [7] and Xi et al. [23] categorize advances in agent architectures, communication protocols, coordination mechanisms, and memory management. Benchmarking efforts like AgentBench [15] provide standardized tasks for evaluating agent collaboration, reasoning, and adaptability. Other studies have begun to explore the social dynamics that emerge when multiple agents interact, including cooperation and coordination behaviors in simulated environments [16], as well as challenges related to maintaining consistent communication and shared goals in dialogue-based collaboration.

This work addresses this gap through a large-scale empirical analysis of GitHub repositories that focus on MAS. By examining their structures, evolution patterns, and development activities, we aim to capture the current landscape of multi-agent software engineering in practice. The study highlights emerging trends in framework design and maintenance challenges, offering new insights into how these complex systems are engineered and evolved. Our findings contribute to a clearer understanding of MAS development and lay the groundwork for future research on improving tools and practices in this rapidly growing field.

## III. METHODOLOGY

### A. Study Objective and Research Questions

This study investigates how multi-agent AI systems are developed and maintained within open-source communities. We analyze repositories that implement agent-based architectures to gain insight into their development patterns and maintenance practices. Two main RQs arise from this goal.

#### RQ1. How do development patterns vary among the MAS repositories?

- RQ1.1: What are the distinct commit activity patterns across repositories?
- RQ1.2: What is the distribution of commit types in MAS repositories?

TABLE II: Overview of dataset composition.

MAS	Issues				Commits
	Total	w/ PRs	Labeled	w/ Both	
AutoGen	750	709	560	530	3,660
CrewAI	175	89	167	82	1,688
Haystack	3,939	1,405	2,720	1,062	4,397
LangChain	558	511	238	220	14,565
Letta	149	135	75	67	4,179
LlamaIndex	800	770	780	762	7,244
Semantic Kernel	4,168	1,108	3,897	1,069	4,741
SuperAGI	274	4	82	1	1,739

**RQ2.** *What are the issue reporting patterns and resolution characteristics in MAS frameworks?*

- RQ2.1: How do issue patterns evolve over time across frameworks?
- RQ2.2: What types of issues are most prevalent across frameworks?

### B. Dataset Construction

1) *Data Collection:* We selected eight popular open-source GitHub repositories with diverse architectural patterns, as detailed in Table I. Using the GitHub GraphQL API [5], we extracted a total of 10,813 closed issues across these repositories. In addition to the issues dataset, we compiled the commit history by cloning each repository and extracting all commit records. This process yielded an initial dataset of 44,041 commits spanning the development history of the selected repositories.

2) *Preprocessing:* Our analysis focused specifically on issues whose resolution involved code modifications. We therefore filtered out issues that lacked associated pull requests (PRs), reducing the dataset from 10,813 to 4,731 issues. We also examined labeled issues to understand issue categorization patterns in these repositories. Of the closed issues, 8,519 contained labels, and when combined with the PR requirement, 3,793 labeled issues with associated PRs remained in our dataset.

For the commit dataset, we addressed data quality issues stemming from Git operations such as cherry-picking and rebasing, which can introduce duplicate commits with identical content but different commit hashes. Through systematic duplicate detection and removal, we refined the dataset from 44,041 to 42,267 unique commits, ensuring data integrity for subsequent code change analyses.

Different subsets of these preprocessed datasets were used for different analyses throughout this study, with the specific dataset selection determined by the RQs being addressed. The breakdown for each repository is presented in Table II.

## IV. RESULTS

### A. Development and Contribution Patterns (RQ1)

1) *What are the distinct commit activity patterns across repositories?:* Commit activity across the analyzed MAS repositories reveals three distinct development profiles. As

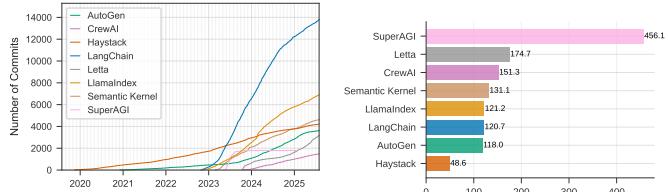


Fig. 2: Commit activity patterns across multi-agent AI frameworks. (a) Cumulative development growth. (b) Variation in monthly commit regularity, higher means irregular patterns.

shown in Figure 2 (a), LangChain leads with around 14,000 commits, reflecting rapid growth starting in mid-2023 and stabilization by 2025.

Haystack shows the most consistent trajectory, with steady contributions since 2020 and the lowest coefficient of variation (coefficient of variation (CV)) at 48.6%, as shown in Figure 2 (b). In contrast, SuperAGI follows a burst-driven pattern, marked by a sharp spike in mid-2023 and minimal activity afterward, resulting in a CV of 456.1%.

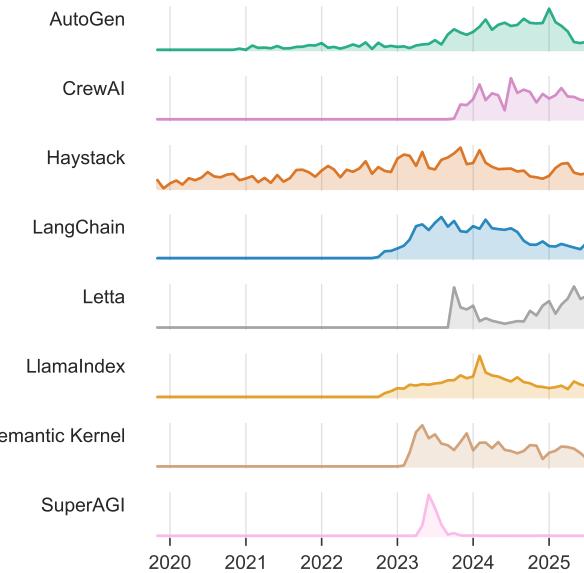


Fig. 3: Monthly commit activity sparklines for each repository showing temporal patterns.

Other systems fall between these extremes, with patterns ranging from steady growth to sporadic surges. As Figure 3 shows, most projects accelerated in 2023, marking a broader inflection point aligned with rising interest in AI agents.

To understand how the three development profiles translate into code evolution, we analyzed code churn patterns, which closely reflect the observed commit trends. SuperAGI added nearly 3 million lines in early 2023, followed by minimal maintenance, suggesting rapid prototyping with limited follow-up (Figure 4). Haystack exhibits deliberate refactoring, marked by two large deletion events ( $\sim 140K$  and  $\sim 290K$  lines) in 2023 and 2024, each affecting over 1.5K files. LangChain shows the most dynamic evolution, with repeated churn peaks (300K – 400K lines) and a major deletion spike ( $\sim 400K$  lines) in mid-2025, indicating ongoing architectural restructuring.

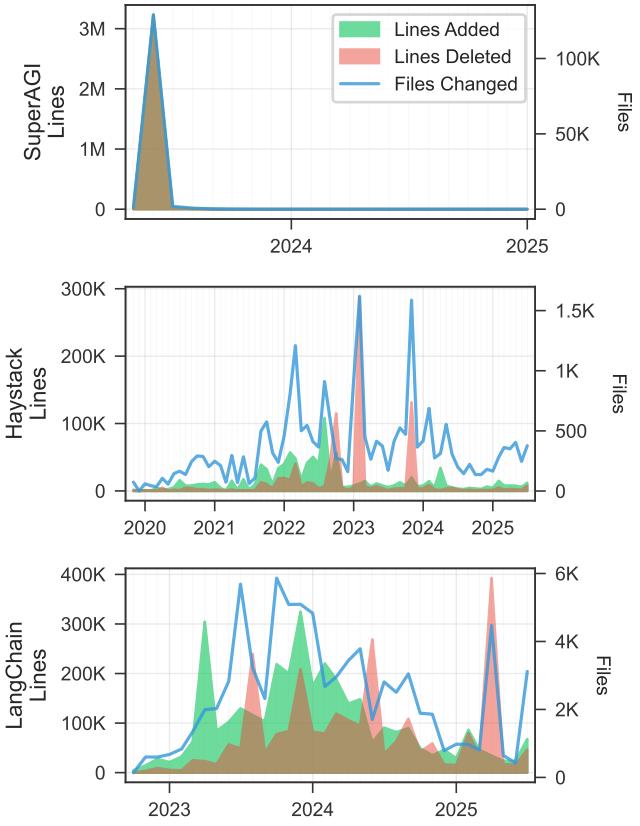


Fig. 4: Code churn patterns showing lines added, lines deleted, and files changed across different development profiles

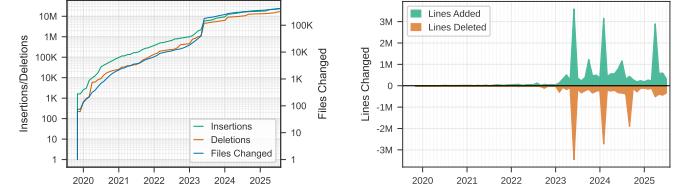


Fig. 5: Ecosystem-level code evolution in multi-agent frameworks. (a) Cumulative growth shows overall expansion in code and file changes. (b) Temporal distribution highlights the balance between code additions and deletions over time.

At the ecosystem level, Figure 5 (a) shows cumulative insertions and deletions reaching 10–20 million lines and over 100K files changed by 2025. The steepest growth occurred between 2020 and 2023. Figure 5 (b) highlights a shift toward maintainability: from 2023 onward, large-scale deletions were often balanced by equivalent insertions, suggesting increased focus on code restructuring rather than expansion.

**Finding 1.1.** *The MAS framework ecosystem exhibits three distinct development profiles: sustained high-intensity development characterized by continuous elevated commit activity, steady consistent development with predictable patterns over time, and burst/sporadic development marked by concentrated periods of intense activity followed by minimal engagement. Most repositories intensified their development starting*

TABLE III: Distribution of commit types across the entire dataset based on maintenance classification.

Commit Type	Percentage
Perfective	40.83%
Corrective	27.36%
Adaptive	24.30%
Adaptive Perfective	5.76%
Corrective Perfective	1.36%
Corrective Adaptive	0.39%

in 2023, suggesting a critical growth period for the ecosystem, though development consistency varies dramatically across projects with CV ranging from 48.6% to 456.1%.

2) *What is the distribution of commit types in MAS repositories?*: Characterizing the distribution of commit types provides valuable insights into the development of MAS. We used a fine-tuned DistilBERT model [2], trained on GitHub commit messages [20], to classify 42,266 commits in our dataset into three fundamental maintenance types: perfective commits that enhance system functionality, adaptive commits that address evolving requirements, and corrective commits that resolve defects and bugs, as well as their combinations.

As shown in Table III, perfective maintenance activities account for 40.83% of all commits, which significantly exceeds both corrective (27.36%) and adaptive (24.30%) maintenance types. The three single-maintenance commits account for 92.49% of all commits, while combined maintenance commits only account for 7.51%.

Our analysis reveals distinct maintenance patterns across AI frameworks as shown in Table IV. Perfective maintenance dominates all frameworks (34.2-51.5%), indicating these systems prioritize feature development over corrective maintenance, characteristic of rapidly evolving domains. Semantic Kernel exhibits the highest perfective ratio (51.5%) with the lowest corrective maintenance (18.3%), suggesting a mature, stable codebase. Conversely, SuperAGI and Letta demonstrate higher corrective ratios (32.8% and 33.5% respectively), potentially indicating less stable architectures or more experimental features. Frameworks demonstrate considerable corrective and adaptive maintenance activity (18.3-33.5% and 17.2-27% respectively), indicating active responses to both defects and changing requirements.

**Finding 1.2.** *The prevalence of perfective commits indicates that the development of MAS prioritizes continuous improvement over reactive bug fixing, indicating a proactive approach to software evolution. Mixed-category commits remain minimal (<8%), revealing atomic, single-purpose development practices where commits typically address focused maintenance tasks rather than combining multiple objectives. These patterns suggest the ecosystem is in an active growth phase, with development efforts concentrated on feature enhancement rather than traditional maintenance.*

TABLE IV: Distribution of commit classifications showing maintenance activity patterns across frameworks.

P: Perfective, C: Corrective, A: Adaptive, AP: Adaptive Perfective, CP: Corrective Perfective, CA: Corrective Adaptive

MAS	P (%)	C (%)	A (%)	AP (%)	CP (%)	CA (%)
AutoGen	43.4	23.7	25.2	5.8	1.4	0.5
CrewAI	44.2	24.5	24.0	6.0	0.9	0.5
Haystack	42.6	26.4	24.3	5.5	0.9	0.4
LangChain	39.2	26.9	25.3	6.8	1.3	0.4
Letta	38.1	33.5	23.2	4.2	0.8	0.2
LlamaIndex	34.2	32.4	27.0	4.3	1.5	0.5
Semantic Kernel	51.5	18.3	20.0	7.5	2.4	0.4
SuperAGI	46.0	32.8	17.2	2.7	1.2	0.1

### B. Issue Landscape (RQ2)

1) *How do issue patterns evolve over time across frameworks?*: The MAS repositories demonstrate heterogeneous issue reporting patterns with significant temporal variations in both volume and activity intensity.

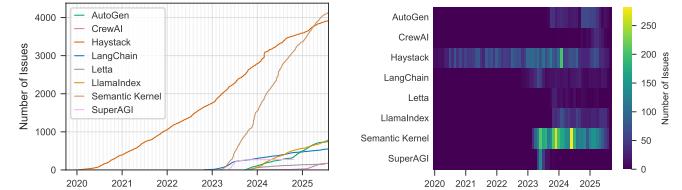


Fig. 6: Issue reporting activity across multi-agent frameworks. (a) Cumulative issue growth illustrates long-term reporting trends. (b) The monthly heatmap highlights bursts of issue creation and maintenance activity over time.

As shown in Figure 6 (a), Haystack and Semantic Kernel emerge as the dominant repositories with approximately 4,000 cumulative issues each by 2025, while other frameworks maintain substantially lower volumes below 1,000 issues. SuperAGI demonstrates a concentrated burst pattern with a dramatic spike in mid-2023 before declining sharply, contrasting with the more gradual emergence of LangChain, LlamaIndex, and other frameworks starting in 2023.

Figure 6 (b) further illustrates these patterns through temporal clustering, showing intense activity periods for Haystack and Semantic Kernel during 2023-2024 with values exceeding 200 issues per month, with Haystack exhibiting sustained high activity since 2020 and Semantic Kernel showing explosive growth starting in late 2023, while repositories like Letta and CrewAI display minimal activity.

Figure 7 reveals substantial variation in issue resolution efficiency across frameworks, with median resolution times ranging from approximately 1 day (LlamaIndex) to over 10 days (Semantic Kernel and Haystack). The boxplot whiskers extend to 1.5 times the interquartile range (IQR), indicating that outliers beyond this threshold are not displayed; thus, actual maximum resolution times may extend considerably beyond the visible range of approximately 80 days. Notably, frameworks like LlamaIndex, Letta, and AutoGen show lower median resolution times with relatively compact IQR, while

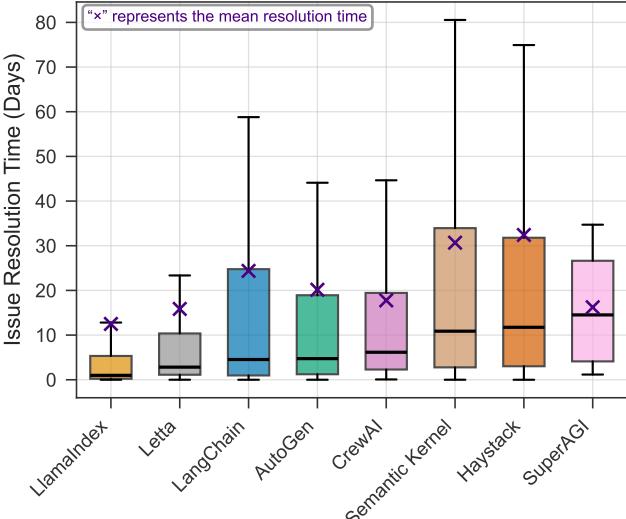


Fig. 7: Distribution of issue resolution times across repositories. Box plots show median (center line), IQR (box), whiskers extending to  $1.5 \times \text{IQR}$ , and mean values (x). Outliers are not shown.

Semantic Kernel and Haystack demonstrate higher medians and greater spread. Mean resolution times (marked with “x”) consistently exceed medians across all frameworks, suggesting right-skewed distributions where a subset of issues requires substantially longer resolution times.

**Finding 2.1.** *The MAS framework ecosystem exhibits highly heterogeneous issue reporting patterns with three distinct profiles: sustained high-volume activity (Haystack, Semantic Kernel with 4,000+ issues), moderate steady growth (LangChain, LlamaIndex with under 1,000 issues), and concentrated burst activity (SuperAGI). Issue reporting intensified dramatically across most frameworks starting in 2023, reflecting rapid ecosystem expansion. Resolution efficiency varies substantially across frameworks, with median times ranging from approximately 1 day to over 10 days. The consistent positioning of mean values above medians across all frameworks indicates right-skewed distributions where most issues resolve relatively quickly while a subset requires extended resolution times that may exceed the displayed range.*

2) *What types of issues are most prevalent across frameworks?:* The analysis of normalized issue labels reveals a clear hierarchy of concern types across the MAS framework ecosystem, with technical implementation challenges dominating over process management and community engagement.

As shown in Table V, Language/Framework-specific tags represent the most frequent category at 31% (3,301 issues), though these labels (e.g., .NET, Java, Python) primarily serve as descriptive markers indicating the technology stack and are typically combined with other labels to provide meaningful context about the underlying issue type. Project Workflow issues constitute 24% (2,563 issues) and Triage labels account

TABLE V: Distribution of issue labels across all repositories (an issue can have multiple labels).

Label	Count	Percentage
Language/Framework	3,301	31%
Project Workflow	2,563	24%
Triage	2,439	23%
Bug	2,355	22%
Infrastructure	1,467	14%
Data Processing	1,161	11%
Agent Issues	1,049	10%
Documentation	734	7%
Feature	727	7%
Community	682	6%
Excluded	266	3%
User Experience	195	2%
Other	434	4%

for 23% (2,439 issues), collectively representing 47% of all issue metadata; however, these categories reflect repository management processes (e.g., priority assignment, PR status, stale issue handling) rather than substantive product concerns, indicating substantial overhead in issue tracking and maintenance coordination. Bug reports represent the most prevalent product-specific concern at 22% (2,355 issues), encompassing defects in functionality and unexpected behavior that require remediation. Infrastructure issues account for 14% (1,467 issues), spanning build systems, deployment configurations, CI/CD pipelines, testing frameworks, dependencies, and platform-specific concerns (Docker, Windows, Azure), reflecting the operational complexity of maintaining MAS frameworks across diverse environments. Data Processing concerns constitute 11% (1,161 issues), addressing challenges in file conversion, indexing, metadata management, vector stores, and integration with various data backends (Elasticsearch, Pinecone, Weaviate), highlighting the centrality of data handling in agent workflows.

Agent-related issues comprise 10% (1,049 issues), focusing specifically on agent behavior, multi-agent coordination, chat history management, planning mechanisms, function calling, and tool usage which is the core differentiating capabilities of MAS frameworks. Documentation and Feature requests each represent 7% (734 and 727 issues respectively), while Community engagement issues account for 6% (682 issues), encompassing help requests, good first issues, and contribution-seeking labels. User experience (UX) concerns constitute only 2% (195 issues), covering command line interface (CLI) interfaces, REST APIs, installation processes, and developer tooling, suggesting either robust UX design or underreporting of usability challenges. Figure 8 illustrates the cumulative growth trajectories of major issue categories, revealing that Bug reports have accumulated over 2,200 issues with accelerated growth starting in 2023, followed by Infrastructure (approximately 1,400 issues) and Agent Issues (approaching 1,000 issues), all showing similar inflection points around 2023 that coincide with the ecosystem’s rapid expansion phase. The high frequency of Bug (22%) and Infrastructure (14%) labels, combined with the relatively modest proportion of Agent-specific

TABLE VI: Description of the key features in the issues for MAS repositories.

Topic	Components	Frequency (%)
Agent Framework Development	agent, declarative, framework, assistant, autogen, collaboration	14.58%
Chat System & Group Communication	chat, message, groupchat, user, agent, chathistory	10.50%
Planning & Sequential Execution	planner, stepwise, sequential, actionplanner, stepwiseplanner	9.79%
AI Service Provider Integration	openai, plugin, api, sdk, endpoint, assistants	9.48%
Evaluation & Performance Metrics	evaluation, metric, pipeline, evaluator, score, output, calculate	8.97%
Model Training & Fine-tuning	model, transformer, training, finetune, tinybert, gpu, distillation, cuda	6.83%
Function Calling & Exception Handling	functioncallingstepwiseplanner, throw, error, exception, functioncallcontent	6.52%
Cloud Agent Orchestration	azureaiagent, azure, agent, streaming, message, thread, service	5.81%
Prompt Engineering & Token Management	prompt, handlebar, token, template, tokenizer, truncation, model, query, generate	5.81%
Question Answering & Text Processing	answer, question, table, text, document, generation, tableqa	4.49%
Kernel Services & Orchestration	kernel, semantic, mcp, openaiassistantagent, server, handofforchestration, orchestration	4.38%
Documentation & Search Features	page, documentation, search, matching, context, doc, bing, highlight, retrieval	3.77%
Other	support, method, model, question, granularity, experiment, tool, image, test, implement	9.07%

labels (10%), indicates that foundational software quality and operational stability challenges are reported as frequently or more frequently than issues related to the unique multi-agent capabilities these frameworks aim to provide. Furthermore, the low prevalence of Documentation (7%), Community (6%), and UX (2%) issues, totaling only 15% collectively, suggests either effective support mechanisms in these areas or systematic underreporting of user-facing concerns relative to technical implementation problems.

MAS introduce unique coordination and interaction challenges that differ from traditional software. Understanding the specific nature of these agent-related issues represents a core focus of our investigation. We analyze agent issues in MAS repositories, using BERTopic [6] to extract key discussion topics and validate their coherence through qualitative analysis. Manual inspection of representative issues confirmed that BERTopic produced meaningful, semantically consistent clusters, revealing several issue categories listed in Table VI.

The distribution indicates that Agent Systems & Intelligence attracts the majority of developer attention (58.42%), encompassing Agent Framework Development (14.58%), Chat System & Group Communication (10.50%), Planning & Sequential Execution (9.79%), AI Service Provider Integration (9.48%), Cloud Agent Orchestration (5.81%), Question Answering & Text Processing (4.49%), and Documentation & Search Features (3.77%). Meanwhile, Technical Implementation & Operations (32.51%) also receives substantial focus, underscoring the significant engineering challenges in deploying MAS. This category includes Evaluation & Performance Metrics (8.97%), Model Training & Fine-tuning (6.83%), Function Calling & Exception Handling (6.52%), Prompt Engineering & Token Management (5.81%), and Kernel Services & Orchestration (4.38%).

**Finding 2.2.** Technical concerns dominate the MAS framework issue landscape, with Bugs (22%), Infrastructure (14%), Data Processing (11%), and Agent-specific issues (10%) as the most prevalent categories. Project management metadata appears frequently, indicating substantial maintenance overhead. All major categories show growth inflection points starting in

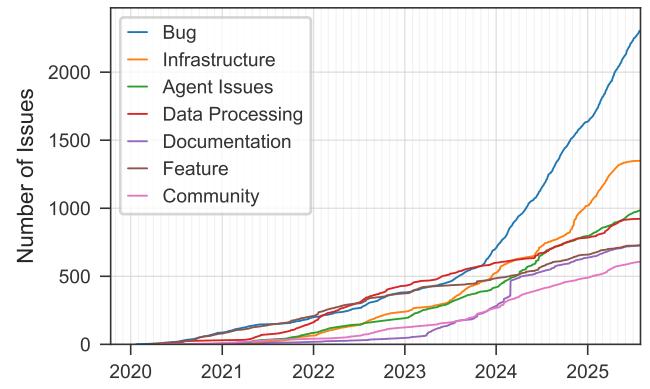


Fig. 8: Cumulative growth trajectories of major issue label categories

2023, correlating with ecosystem expansion. Topic analysis reveals developers prioritize agent capabilities (58%) while contending with operational challenges (33%), reflecting a feature-driven community facing persistent deployment barriers.

## V. THREATS TO VALIDITY

**Internal Validity.** Potential threats to internal validity stem from the repository selection process. Although we selected the most active open-source multi-agent AI frameworks to ensure relevance, relying on GitHub metadata and project activity could still include peripheral or prototype repositories. To mitigate this, we applied filtering and cleaning steps to remove inactive forks and retain only repositories with consistent commit and issue activity. While this improves representativeness, some smaller yet meaningful projects may have been excluded.

**External Validity.** Our findings are based solely on open-source projects hosted on GitHub. As a result, they may not fully generalize to proprietary multi-agent systems. Nevertheless, open-source frameworks dominate this emerging ecosystem, making GitHub an appropriate setting for studying early development and maintenance trends.

**Construct Validity.** Metrics derived from commits and issues provide quantitative evidence of development and maintenance

activity, but may overlook qualitative aspects such as code quality, design rationale, or developer expertise. We mitigated this limitation by combining multiple indicators, including commit types, code churn, issue categories, and responsiveness metrics, to strengthen construct alignment with real development practices.

**Reliability.** Reliability depends on the reproducibility of our data collection and analysis procedures. Changes to GitHub’s API or repository structures could affect future replication. Automated classification of commit and issue text may also introduce minor labeling errors. To reduce these risks, we documented all data extraction and filtering steps and verified a subset of automated classifications for consistency.

By recognizing these limitations, we aim to ensure transparency about the study’s scope and provide confidence in the validity and reproducibility of our results.

## VI. CONCLUSIONS

This study presents the first large-scale empirical analysis of development and maintenance practices in multi-agent AI systems. We examine eight of the most prominent open-source projects, covering over 42K unique commits and 4.7K resolved issues. By analyzing commit activity, code evolution, and issue handling, we offer a quantitative perspective on how these frameworks are built and sustained within open-source communities. Our findings reveal an ecosystem undergoing rapid growth but still in the process of stabilizing. Development is largely driven by feature enhancement, with perfective commits comprising the majority of changes. Issue data indicate that bugs, infrastructure concerns, and agent coordination challenges are the most common problems reported. Resolution times vary widely across projects, reflecting differences in maturity and maintainer capacity. A sharp rise in commit and issue activity after 2023 signals increased adoption and community engagement, fueled by the rising use of LLMs in MAS.

## REFERENCES

- [1] Raymond PL Buse and Westley R Weimer. Automatically documenting program changes. In *Proceedings of the 25th IEEE/ACM international conference on automated software engineering*, pages 33–42, 2010.
- [2] J. Castano, S. Martinez-Fernandez, X. Franch, and J. Bognier. Analyzing the evolution and maintenance of ml models on hugging face. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*, pages 607–618, Los Alamitos, CA, USA, apr 2024. IEEE Computer Society.
- [3] CrewAI. CrewAI Documentation - CrewAI — docs.crewai.com. <https://docs.crewai.com/>. [Accessed 27-10-2025].
- [4] Jon Eyolfson, Lin Tan, and Patrick Lam. Do time of day and developer experience affect commit bugginess? In *Proceedings of the 8th Working Conference on Mining Software Repositories*, pages 153–162, 2011.
- [5] Github. GitHub GraphQL API documentation - GitHub Docs — docs.github.com. <https://docs.github.com/en/graphql>. [Accessed 27-10-2025].
- [6] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*, 2022.
- [7] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8048–8057. International Joint Conferences on Artificial Intelligence Organization, 8 2024. Survey Track.
- [8] Haystack. Haystack Introduction — docs.haystack.deepset.ai. <https://docs.haystack.deepset.ai/docs/intro>. [Accessed 27-10-2025].
- [9] Semantic Kernel. Introduction to Semantic Kernel — learn.microsoft.com. <https://learn.microsoft.com/en-us/semantic-kernel/overview/>. [Accessed 27-10-2025].
- [10] Sunghun Kim, Thomas Zimmermann, E James Whitehead Jr, and Andreas Zeller. Predicting faults from cached history. In *29th International Conference on Software Engineering (ICSE’07)*, pages 489–498. IEEE, 2007.
- [11] LangChain. Home - Docs by LangChain — docs.langchain.com. <https://docs.langchain.com/>. [Accessed 27-10-2025].
- [12] Letta. Home — Letta — docs.letta.com. <https://docs.letta.com/>. [Accessed 27-10-2025].
- [13] Li Li, Tegawendé F Bissyandé, Haoyu Wang, and Jacques Klein. Cid: Automating the detection of api-related compatibility issues in android apps. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 153–163, 2018.
- [14] Jerry Liu. LlamaIndex. [https://github.com/jerryliu/llama\\_index](https://github.com/jerryliu/llama_index), 11 2022.
- [15] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.
- [16] Joon Sung Park, Joseph O’Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST ’23*, New York, NY, USA, 2023. Association for Computing Machinery.
- [17] Peng Qian, Zhenguang Liu, Qinming He, Butian Huang, Duanzheng Tian, and Xun Wang. Smart contract vulnerability detection technique: A survey. *arXiv preprint arXiv:2209.05872*, 2022.
- [18] Foyuzr Rahman, Daryl Posnett, Israel Herranz, and Premkumar Devanbu. Sample size vs. bias in defect prediction. In *Proceedings of the 2013 9th joint meeting on foundations of software engineering*, pages 147–157, 2013.
- [19] Baishakhi Ray, Daryl Posnett, Vladimir Filkov, and Premkumar Devanbu. A large scale study of programming languages and code quality in github. In *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, pages 155–165, 2014.
- [20] Muhammad Usman Sarwar, Sarim Zafar, Mohamed Wiem Mkaouer, Gursimran Singh Walia, and Muhammad Zubair Malik. Multi-label classification of commit messages using transfer learning. In *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 37–42. IEEE, 2020.
- [21] SuperAGI. Introduction to SuperAGI — SuperAGI Docs — superagi.com. <https://superagi.com/docs/>. [Accessed 27-10-2025].
- [22] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryan W White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.
- [23] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, Qi Zhang, and Tao Gui. The rise and potential of large language model based agents: a survey. *Science China Information Sciences*, 68(2):121101, 2025.