Triangle: Empowering Incident Triage with Multi-Agent

Experience Paper

***Abstract*—As cloud service systems grow in scale and complexity, incidents that indicate unplanned interruptions and outages become unavoidable. Rapid and accurate triage of these incidents to the appropriate responsible teams is crucial to maintain service reliability and prevent significant financial losses. However, existing incident triage methods relying on manual operations and predefined rules often struggle with efficiency and accuracy due to the heterogeneity of incident data and the dynamic nature of domain knowledge across multiple teams.**

To solve these issues, we propose Triangle, an end-to-end incident triage system based on a Multi-Agent framework. Triangle leverages a semantic distillation mechanism to tackle the issue of semantic heterogeneity in incident data, enhancing the accuracy of incident triage. Additionally, we introduce multi-role agents and a negotiation mechanism to emulate human engineers' workflows, effectively handling decentralized and dynamic domain knowledge from multiple teams. Furthermore, our system incorporates an automated troubleshooting information collection and mitigation mechanism, reducing the reliance on human labor and enabling fully automated end-to-end incident triage. Extensive experiments conducted on a real-world cloud production environment demonstrate that TRIANGLE significantly improved incident triage accuracy (up to 97%) and reduced Time to Engage (TTE) by as much as 91%, demonstrating substantial operational impact across diverse cloud services.

I. INTRODUCTION

As cloud service systems continue to expand and become increasingly complex, system incidents are inevitable [1–6]. These incidents often signify unplanned interruptions and even system outages. Therefore, when an incident occurs, prompt handling by the responsible team is crucial to prevent further failures and avoid significant financial losses [4]. However, in today's large-scale cloud service systems, a single system may involve numerous teams with different functions. Thus, incidents should be assigned to the correct responsible team, a process known as **Incident Triage** [3, 7]. If an incident is misassigned, it usually cannot be properly resolved and needs to be reassigned based on feedback from that team until the correct team is identified. Poor incident triage can significantly extend the Time To Engage (TTE), increasing system risk exposure. Therefore, rapid and accurate incident triage is critical for shortening recovery time and ensuring service quality.

Traditional incident triage processes typically rely on manual operations combined with predefined rules, where engineers use various tools to further investigate incident-related issues. This process often involves ad hoc meetings across multiple relevant teams, consuming substantial human resources and time, making rapid fault resolution difficult.

To automate the incident triage process, it can be directly compared to the bug triage problem. Academia has recently conducted extensive research on bug triage [8–13]. A typical approach involves a unified model pre-trained on historical datasets to assign bugs to various teams through a one-time classification. However, this method has limitations and cannot meet the performance requirements for incident triage. The core reason is that, unlike bug reports which often contain detailed reproduction steps and context provided by developers, the original information for operational incidents is typically automatically generated alerts lacking deep semantic context (e.g., "CPU utilization high on server X"), or user-submitted phenomenological descriptions (e.g., "Cannot log in"). Such information often lacks direct pointers to the root cause or the responsible team. The scarcity and ambiguity of information make it difficult for traditional methods reliant on keyword matching or shallow feature learning to achieve the accuracy required for practical applications.

As shown in Fig. 1, this process involves three teams and multiple engineers, and can be both time-consuming and complex. Without a thorough investigation by Teams A and B, directly identifying Team C as the correct responsible party is not feasible. To develop an incident triage system that meets real-world requirements for efficiency and accuracy, we conducted an in-depth investigation into the incident management practices of a leading global technology company's cloud services. Based on our practical experience, we have identified three key challenges in achieving efficient and accurate incident triage:

- **Incident Semantic Heterogeneity:** Incident data exhibits significant variation in how semantically similar issues are described. Key phrases crucial for triage are scattered and lack standardized templates, hindering traditional methods. LLMs' contextual understanding can better capture these underlying associations despite diverse phrasings.
- **Decentralized and Dynamic Domain Knowledge:** Effective incident triage often requires integrating knowledge from multiple, independently evolving teams. A team's responsibilities and associated domain knowledge change over time, necessitating a flexible approach like a multi-agent framework where agents with specific team knowledge can collaborate.
- **High Human Labor:** Relying heavily on manual injection of domain knowledge for incident triage incurs substantial human effort and cost, impeding end-to-end automation and increasing the time to engage.

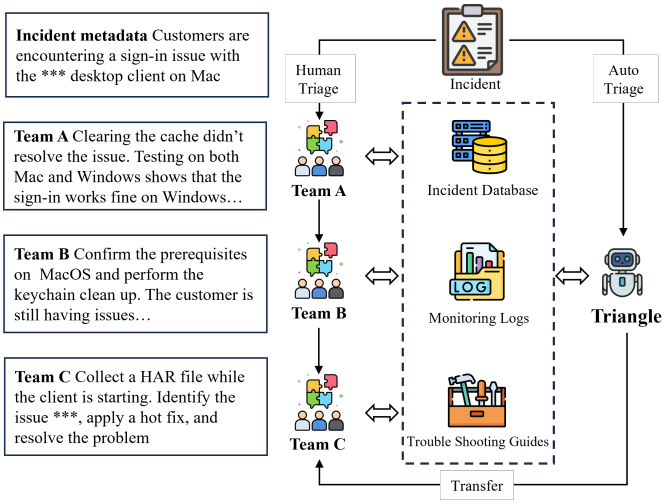

Fig. 1: Comparison of manual incident triage and TRIANGLE. In the manual process, a sign-in issue on Mac devices is escalated across Teams A, B, and C before resolution. Each team applies their domain knowledge and tools in sequence. In contrast, TRIANGLE streamlines this multi-team workflow by using agents to simulate human collaboration, reducing handoffs and accelerating resolution.

To address these issues, we design the TRIANGLE system, an end-to-end incident triage system based on a Multi-Agent framework. We opt for a "multi-agent" approach, rather than a monolithic agent, to better simulate the real-world collaborative problem-solving patterns of multiple teams. This allows different agents to specialize in domain knowledge and tools specific to certain areas, and then negotiate to reach a consensus, thereby enhancing the robustness and interpretability of the triage process.

We develop a semantic distillation mechanism, which leverages LLMs to extract core, actionable semantic information relevant to triage decisions from raw, potentially noisy, and redundant incident data. This mechanism then effectively aligns this distilled information with the domain knowledge of relevant teams, thereby addressing incident semantic heterogeneity and improving the accuracy of incident triage. Additionally, we emulate the workflow of human engineers in solving incident triage problems by innovatively designing multi-role agents and proposing an effective negotiation mechanism. This allows for the distributed and dynamic handling of multi-team domain knowledge, effectively mitigating the impact of decentralized and dynamic domain knowledge on triage performance. By utilizing the robust semantic understanding of LLMs, we design an automated Team Information Enrichment mechanism throughout the entire incident triage process, enabling end-to-end triage even in scenarios requiring reassignment, without additional human labor costs.

We conduct extensive experiments with TRIANGLE using incident triage data collected from a real-world production environment serving tens of millions of users. TRIANGLE has significantly improved incident triage accuracy (up to 97%) while reducing Time to Engage (TTE) by up to 91%, demonstrating substantial operational impact across diverse cloud services. In our offline experiments, TRIANGLE outperforms the state-of-the-art method (DeepCT [3]) by 26%–42% relative improvement in hop accuracy across all hops, achieving up to 91.7% accuracy at hop 5 without relying on manually enriched discussions. To show the general capabilities of TRIANGLE, we also conducted experiments on publicly available datasets. The results show that TRIANGLE generalizes beyond incident triage, achieving 63.2% accuracy on the MSR 2013 Bug Dataset—outperforming all baselines by an average of 51.0%, with gains ranging from 15.3% to 134.9%. Our model has been successfully deployed in a system with tens of millions of users at a leading global technology company.

Our contributions are summarized as follows:

- To the best of our knowledge, TRIANGLE is the first end-to-end incident triage system leveraging a Multi-Agent framework to automate triage in large-scale cloud environments, enhancing both efficiency and accuracy.
- We design a multi-role agent framework with an effective negotiation mechanism that mimics human engineer workflows in incident triage, dynamically managing multi-team domain knowledge to overcome challenges of decentralization and dynamism.
- We develop an automated Team Information Enrichment mechanism, integrated throughout the triage process, enabling TRIANGLE to achieve end-to-end triage, including reassignments, without additional human labor.
- TRIANGLE is deployed on real-world systems serving tens of millions of users, where it has received consistently positive feedback from on-call engineers. This real-world adoption underscores the system's practical effectiveness and has yielded valuable insights that inform and motivate further research in the domain of automated incident triage.

II. BACKGROUND

A. Incident Data

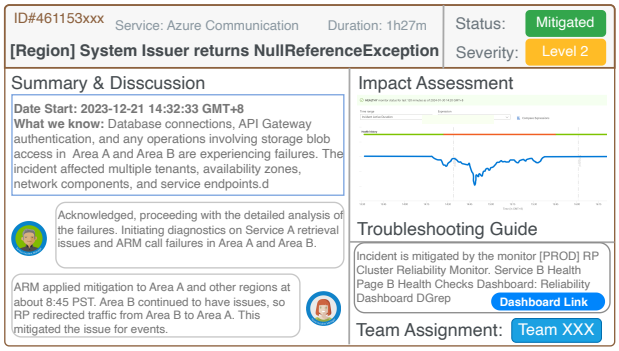

Fig. 2: Example of incident data post-triage, including metadata (title, duration, status, severity), a content body (summary and two related discussions), and additional fields such as impact assessment, consulted troubleshooting guide, and final team assignment.

Incident data serves as a critical indicator of service quality in large-scale cloud service systems. An incident is often triggered by anomalies, faults, or unplanned interruptions within a system, which could lead to significant service degradation or outages. The rapid identification and resolution of incidents are essential to maintaining high service quality and minimizing financial losses. However, modern cloud environments are highly complex, often involving numerous interconnected components and teams with distinct functions. This complexity amplifies the challenges associated with understanding and processing incident data effectively.

Incident data typically comprises system-generated logs, telemetry, alerts, and user reports, which can vary widely in their format, granularity, and relevance. Incident data are typically semi-structured, with natural-language descriptions being most critical, as shown in Fig. 2. The same incident pattern might correspond to different teams depending on subtle contextual factors, while different patterns might lead to the same team. This variability complicates the classification and assignment of incidents to the correct teams, impacting the effectiveness of incident triage systems.

B. Incident Triage

Incident triage is the process of quickly and accurately assigning incidents to the appropriate teams to ensure timely mitigation. In large-scale cloud service environments, a poorly managed triage can significantly increase Time to Engage (TTE), which is detrimental to both service quality and customer trust. Traditional approaches to incident triage rely heavily on predefined rules, human expertise, and manual operations. Engineers often need to investigate incident details using various tools, collaborate across multiple departments, and adjust triage decisions based on evolving insights and feedback. Even for human engineers, incident triage remains a very challenging task. The triage process involves extensive discussions across multiple teams, which results in a very long TTE in real-world industrial settings, sometimes even stretching to several weeks. We present the results of a real-world empirical study in Fig. 3. This study covers over 3,000 teams and hundreds of services over a 12-month period. We analyze two key metrics: the *median time units* spent on incident triage and the *median number of discussions* triggered per incident. The findings highlight a clear and pressing need for an automated end-to-end incident triage system in real-world industrial settings.

Incident triage shares similarities with bug triage, both being multi-class, single-label classification problems where different teams represent different classes. However, unlike bug triage, incident data is often generated automatically by system components or manually by users, lacking the richness needed for one-time classification models to perform effectively. This lack of context-rich information necessitates more sophisticated methods that can dynamically incorporate evolving domain knowledge and adapt to changing system states. For current incident triage systems, the initial assignment is often based on static rules or simple heuristics. In practice, incidents frequently need multiple reassignments before they reach the right team, increasing Transfer Hop Counts, TTE, and TTM. This reliance on manual routing and human experience further exacerbates inefficiencies and inconsistencies, underscoring the need for more automated and intelligent triage systems.

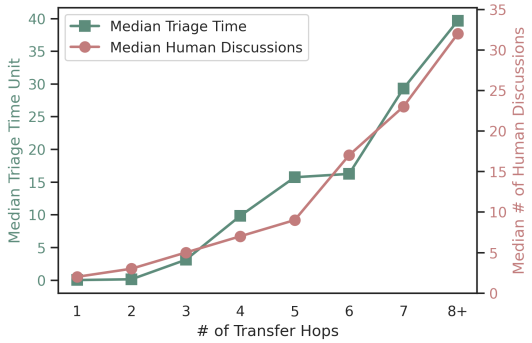

Fig. 3: Median triage time units (green) and median human discussion turns (red) per incident, segmented by number of transfer hops, aggregated across over 3,000 teams and hundreds of services over 12 months.

C. Multi-Agent

Large Language Models (LLMs) exhibit impressive planning and reasoning, enabling their use as autonomous agents [14–16]. However, single LLM agents face limitations with highly complex problems, indicating that even advanced individual models may not be universally optimal [14]. This has spurred the development of LLM-based multi-agent systems, which coordinate multiple LLMs, often specialized by distinct profiles, capabilities, and roles for specific tasks or problem facets [14, 17, 18]. These systems leverage collective intelligence and specialized skills to enhance complex problem-solving and create more adaptable simulations, such as multiple agents assuming roles in software development [19, 20], by mirroring human teamwork in collaborative decision-making.

While scaling multi-agent systems presents challenges like increased computational demands, advancements such as dynamic agent generation [21] and sophisticated orchestration methodologies are vital for improving resource utilization and coordination. The inherent modularity facilitates flexible integration of new capabilities, enabling agents to learn and evolve through mechanisms like memory and self-evolution [14, 22]. Notable frameworks like MetaGPT (structuring collaboration for software development [19]), CAMEL (focusing on autonomous cooperation via inception prompting [23]), and AutoGen (a versatile framework for customizable multi-agent applications [24]) demonstrate significant potential. Applications span automating complex coding [19, 20], creating interactive simulacra [25], and improving LLM factuality and reasoning through multi-agent discussions [26], showcasing their ability to yield more robust and effective outcomes.

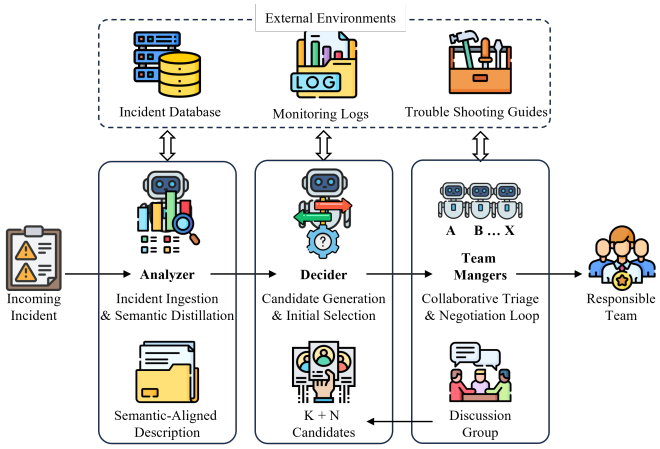

Fig. 4: TRIANGLE framework. The *Analyzer* Agent ensures semantic clarity of incoming incidents. The *Decider* Agent proposes candidate teams based on historical data and functional expertise, initiating a discussion group with relevant *Team Manager* Agents. These agents access external tools to contribute insights. Through discussion and voting, the incident is assigned; if unresolved, the process iterates with a refined candidate pool.

III. APPROACH

A. Overview

The core of TRIANGLE is a multi-agent system designed to automate and optimize the incident triage process. This system emulates human expert collaboration through distinct agent roles, each contributing specialized functions from incident understanding to final team assignment. The overall workflow, illustrated in Fig. 4, involves three main phases: incident ingestion and semantic distillation (Section III-B), team candidate generation and initial selection (Section III-C), and a collaborative triage and negotiation loop (Section III-D). Three primary agent types orchestrate this workflow: the Analyzer Agent, the Decider Agent, and the Team Manager Agent.

This multi-agent architecture is designed based on two key principles. First, it mirrors the collaborative workflow of human engineering teams, where specialists with different expertise work together through analysis, discussion, and consensus-building. Second, the differentiated agent roles maintain knowledge at multiple granularities. The Analyzer Agent holds fine-grained technical knowledge, the Decider Agent maintains strategic selection criteria, and Team Manager Agents possess team-specific operational knowledge. This knowledge stratification enables the negotiation phase to leverage domain expertise from technical, strategic, and operational perspectives simultaneously, leading to more comprehensive and robust triage decisions that emulate the collaborative reasoning process of effective human expert teams.

B. Phase 1: Incident Ingestion and Semantic Distillation

Upon entering the incident tracking system, a new incident first undergoes semantic distillation, a critical process orchestrated by the Analyzer Agent. The primary objective of this phase is to extract and refine the semantic information most pertinent to accurate triage, thereby mitigating issues that arise from inconsistent or noisy incident descriptions. The Analyzer Agent is tasked with preprocessing new incidents and then activating the semantic distillation mechanism. This mechanism transforms raw incident data into a structured and semantically aligned format, optimized for subsequent interpretation and action by other agents.

The semantic distillation mechanism itself is designed to identify key phrases essential for triage and to ensure conceptual consistency between the incident's description and the organization's *team functional documents*, which constitute a key knowledge environment. This is achieved through two coordinated steps: semantic alignment and key phrase extraction.

Initially, semantic alignment is performed to harmonize the incident's terminology with that used within the team functional documents. This multi-step alignment begins with *Initial Keyword Identification*, where the incident text is analyzed using TF-IDF to pinpoint statistically significant terms (e.g., *"latency"*, *"API error"*). Following this, during *Terminology Matching*, these identified terms are meticulously cross-referenced with a domain-specific glossary derived from the team functional documents. This step helps find preliminary matches and standardize terminological variations. The alignment process culminates in *LLM-based Semantic Refinement*. Here, an LLM-powered component of the Analyzer Agent actively engages with the team functional documents as an interactive knowledge source. Guided by the original incident text and the previously identified keywords, this component queries and navigates the document corpus to dynamically retrieve relevant contextual information and terminological standards. It then leverages this retrieved knowledge to rephrase the incident description, ensuring it aligns with the established vocabulary and conceptual framework of the teams while preserving the original meaning, ultimately producing an incident description that is semantically harmonized with internal team knowledge.

Subsequent to successful alignment, the Analyzer Agent proceeds with key phrase extraction to distill core triage information. In this step, words within the aligned incident data are assigned weights based on their TF-IDF scores relative to the entire collection of team functional documents. These weights, along with the semantically aligned incident text, are then presented to another LLM-driven component tasked with expert-level summarization. This component, acting as a triage expert, analyzes the provided information—drawing upon its understanding of relevant contexts (potentially informed by the same team functional documents it can conceptually access) to extract three pivotal types of key phrases: those related to the failure location, phrases describing the symptoms of the failure, and phrases indicating the capabilities likely required to resolve the incident. These extracted key phrases are then appended to the original incident data, serving as a refined and focused input for the Triage Decider Agent in the next phase.

Algorithm 1 Triage Decider for Incident Team Assignment

Require: Incident I, Historical Incidents $\mathcal{H}$, Team Documents $\mathcal{D}$

Ensure: Candidate Teams $\mathcal{T}^*$

// Step 1: Compute similarity with historical incidents

$\mathbf{V} \leftarrow \text{TFIDF}(\mathcal{H})$ ▷ Vectorize $\mathcal{H}$ using TF-IDF

$S(I, \mathcal{H}) \leftarrow \text{cosine}(\text{TFIDF}(I), \mathbf{V})$ ▷ Compute similarity

$\mathcal{T}_1 \leftarrow \{\text{teams of top-}K(S(I, \mathcal{H}))\}$ ▷ Select top K team candidates

// Step 2: Candidate teams refinement using LLM

$\mathcal{D}' \leftarrow \text{LLMcompress}(\mathcal{D})$ ▷ Compress team documents, high compress rate

$\mathcal{T}_2 \leftarrow \{\text{top-}N(\text{LLMmatch}(I, \mathcal{D}'))\}$ ▷ Retrieve top N team candidates

// Step 3: Final ranking of candidates

$\mathcal{D}'' \leftarrow \text{LLMcompress}(\mathcal{D})$ ▷ Compress team documents, low compress rate

$\mathcal{T}^* \leftarrow \text{LLMrank}(I, \mathcal{T}_1 + \mathcal{T}_2, \mathcal{D}'', \text{key phrases})$ ▷ Rank candidates

return $\mathcal{T}^*$

C. Phase 2: Candidate Generation and Initial Selection

With the semantically distilled incident information, the Triage Decider Agent identifies an initial set of suitable team candidates. The Triage Decider Agent is central to selecting appropriate teams. For initial assignment, it employs a two-pronged approach to generate candidates, as detailed in Algorithm 1.

Historical Incident Matching. The current incident is vectorized using TF-IDF and compared against a vectorized history of past incidents ($\mathcal{H}$) using cosine similarity. The teams that handled the top-K most similar historical incidents are selected as $\mathcal{T}_1$.

Team Document Matching. To assess relevance against team capabilities, an LLM-powered component within the Triage Decider Agent first interacts with the team functional documents ($\mathcal{D}$). It strategically processes and compresses these documents (e.g., through summarization) into more concise representations ($\mathcal{D}'$) suitable for efficient matching while aiming to preserve core functional information. Subsequently, this or another LLM-component actively consults these condensed representations ($\mathcal{D}'$), evaluating the current incident against each team's summarized profile to identify and retrieve the top-N most relevant teams as $\mathcal{T}_2$.

The two sets of candidates, $\mathcal{T}_1$ and $\mathcal{T}_2$, are combined. To refine this combined list, the Triage Decider may activate a further LLM-driven analysis. This component then undertakes a more in-depth interaction with less compressed versions of the team documents ($\mathcal{D}''$) corresponding to the combined candidates, and cross-references the incident specifics, including the key phrases extracted in Phase 1, against each team's detailed capabilities to establish a final ranking and select the top M teams for the discussion group. M can be configured by engineers according to the actual service requirements.

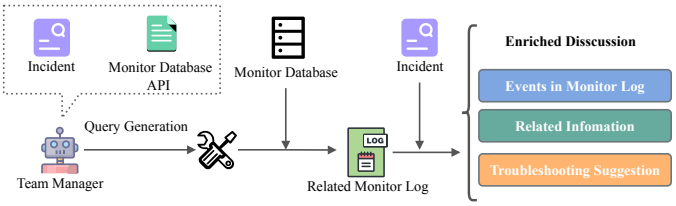

Fig. 5: Team Information Enrichment Mechanism, allowing Team Manager Agents to access and summarize external data sources, such as monitoring logs, to support triage decisions.

D. Phase 3: Collaborative Triage and Negotiation Loop

The top M candidate teams selected by the Triage Decider form a discussion group to collaboratively determine the most appropriate team. This phase involves Team Manager Agents and employs a voting-based negotiation mechanism. Each team within the organization has a corresponding Team Manager Agent. Its primary role is to assess if an incident falls within its team's responsibility, leveraging its specific domain knowledge and tools. When a team's responsibilities change, only its functional documentation needs updating, which the agent dynamically uses.

1) Team Information Enrichment Mechanism: A key capability of the Team Manager Agent is to augment the incident data with relevant contextual information. This mechanism, illustrated in Fig. 5, automates the retrieval and summarization of such information:

- *Information Retrieval:* The Team Manager Agent extracts entities like time ranges and component names from the incident. It uses these to automatically generate and execute queries against its team-specific monitoring databases (e.g., for logs). It can infer missing query parameters based on the incident description and database interface documentation.
- *LLM-based Summarization:* The retrieved logs, often voluminous and noisy, are not directly used. Instead, an LLM-powered analytical component within the Team Manager Agent is activated to interact with this retrieved data. This component intelligently sifts through the logs, treating them as a dynamic information environment, and correlates log events with the incident description it has access to. Drawing on its reasoning capabilities, it then synthesizes these findings into "enriched discussion" points, covering: a. Potential events in the logs related to the incident. b. Correlation between log information and the incident description. c. Troubleshooting suggestions derived from the logs.

This enriched information, tagged with the providing team, is added to the incident.

2) Voting-based Negotiation: Once all Team Manager Agents in the discussion group have had a chance to provide enriched information, the aggregated details are shared among them. Each Team Manager Agent then votes for the team it deems most suitable to handle the incident from the current discussion group. If a single team receives a majority of votes (e.g., more than half), the incident is assigned to that team, and the triage process concludes.

3) Reassignment Process: If the voting does not result in a consensus, the negotiation is considered to have failed for that round. The incident, now augmented with the collective enriched discussion from all participating teams, is sent back to the Triage Decider Agent. For reassignment, the Triage Decider:

- Removes the teams that were part of the failed negotiation from the immediate candidate pool for the next round to avoid immediate loops (or specifically the team that might have been voted for but didn't reach consensus if applicable, or applies other heuristics).
- Leverages the newly acquired discussion information.
- Activates an LLM-component to select a new set of candidate teams. This component primarily engages with the team functional documents as its knowledge environment, performing a nuanced matching of the enriched incident (now containing insights from the previous negotiation round) against the documented capabilities of various teams. Historical TF-IDF based selection (Step 1 in Algorithm 1) is typically not reused in reassignment rounds to focus on the fresh insights.

This negotiation loop can repeat. To prevent infinite cycles, a maximum number of reassignment loops is set (e.g., 5 loops in our settings). If no consensus is reached after the maximum loops, TRIANGLE assigns the incident based on the last voting result, which could involve human engineers at this stage.

This multi-agent approach allows TRIANGLE to dynamically adapt its triage strategy based on evolving information and collaborative insights, mirroring complex decision-making processes performed by human expert teams.

IV. EVALUATION

In this study, to fully evaluate the performance of TRIANGLE in incident triage within a real-world production environment, we aim to address the following research questions:

- RQ1: **Business Impact** - How effective is TRIANGLE in terms of time savings during incident triage in real-world industry scenarios?
- RQ2: **Overall Performance** - What is the accuracy of TRIANGLE in the continuous incident triage process within large-scale cloud service systems?
- RQ3: **Ablation Study** - What is the contribution of each key component to the overall performance of TRIANGLE?
- RQ4: **General Capabilities** - How does the general capability of TRIANGLE perform? Can it achieve good performance in tasks similar to incident triage (*e.g.* bug traige)?

A. Dataset

To evaluate the performance of TRIANGLE in real-world scenarios, we collected 15 months of real incident data from large-scale cloud service systems serving tens of millions of users at a leading global technology company. These cloud services involve hundreds of engineering teams. To ensure a quantitative and objective experiment, we concentrated solely on incidents that had been resolved, as their confirmed assignments facilitate an accurate assessment of the incident triage process. Specifically, we split the data into a 12-month period for historical incident data and a subsequent 3-month period for evaluating the performance of TRIANGLE. For our experimental analysis, we concentrated solely on incidents that had been resolved, as their confirmed assignments facilitate an accurate assessment of the incident triage process. Specifically, we split the data into a 12-month period for building the knowledge base and a subsequent 3-month period for testing the performance of TRIANGLE.

B. Experiment Setup

1) Metrics: Accuracy and Time to Engage are the two most crucial evaluation metrics in incident triage. Below, we will provide a detailed introduction to these two metrics.

Accuracy. Accuracy is a widely used metric in classification tasks and is a core indicator for evaluating the end-to-end performance of incident triage. However, in incident triage, due to the involvement of reassignment, we further refine the concept of accuracy. We introduce Hop Accuracy. Its calculation is the same as traditional accuracy, but with a restriction on the number of hops for reassignment. Hop N Accuracy ($N \geq 1$) represents the accuracy when the number of assignments does not exceed N by the time the model completes the final assignment. This places a higher requirement on the model's capabilities.

Time to Engage (TTE). TTE refers to the time elapsed from when an incident is reported to when it is assigned to the correct team. TTE is a key factor in measuring the efficiency of incident triage. In practical scenarios, the model's runtime accounts for a minimal portion of the entire triage process. This is because, during triage, engineers from different teams may conduct further analysis of the incident, and there may also be meetings between teams. The time spent by human engineers in these activities constitutes the majority of the triage process.

2) Baselines: To evaluate the performance of TRIANGLE, we introduce several baseline methods.

- ContentBased [7]: Uses locality-sensitive hashing to find suitable teams, helping mitigate cold start issues by identifying patterns in new or sparse data.
- InvertedIndex [27]: Builds a inverted index table re-ranked by IDF scores to rank teams.
- LGBM [28]: Employs a one-vs-all LightGBM model to handle sparse and unstructured data.
- MART [29]: Utilizes a multiple additive regression tree (MART) model, trained with a one-vs-all FastTree [30] classifier to assign incidents.
- DeepCT [3]: The state-of-the-art incident triage method based on deep learning.

The first four methods are traditional machine and statistical learning methods, that are widely used in the industry. DeepCT [3] is a state-of-the-art incident triage method based on deep learning. DeepCT utilizes Convolutional Neural Networks (CNNs) to encode domain-specific discussions. It then leverages Gated Recurrent Units (GRUs) to capture temporal

TABLE I: Average triage accuracy per services after deploying TRIANGLE in large-scale service systems, along with the percentage reduction in Time to Engage (TTE), comparing the three months before and after deployment to ensure significant observability and unbiased evaluation.

Team	A	B	C	D	E	F
Triage Accuracy (%)	92	82	96	64	96	97
TTE Reduction (%)	18	91	48	72	61	67

relations and applies attention mechanisms to reduce the impact of noise. This method relies heavily on the availability of extensive discussions from engineers.

C. RQ1: Business Impact

Adoption. TRIANGLE has been running in production and is actively used to triage cloud incidents at a leading global technology company that serves tens of millions of users worldwide. Two organizations within the company have adopted TRIANGLE as their primary approach to incident triage: one is a cloud platform provider operating multiple services, and the other manages a large-scale, customer-facing service. Both organizations integrate their incident management systems with TRIANGLE to supply relevant data. Despite differences in system architecture and domain knowledge, TRIANGLE has demonstrated strong robustness and scalability across these varied environments. Feedback from service teams highlights the impact of the system:

> *"*TRIANGLE*'s automated routing capabilities help reduce engineering toil and enhance customer experience by accelerating mitigation through more efficient incident handling."*

Scale. While the exact number of incidents processed by TRIANGLE is sensitive and cannot be disclosed, it operates under high-scale production conditions. Specifically, it utilizes approximately 600 million logs per day and analyzes over 2,000 distinct fault types. In total, more than 20 TB of data is processed daily. This scale of operation underscores the robustness and efficiency of TRIANGLE in handling diverse, large-volume telemetry in real-world cloud environments.

TTE saving. Time to Engage (TTE) is a key business metric for evaluating the efficiency of incident triage models. To assess the real-world performance of TRIANGLE, we select the six most recently updated services (designated A through F) and compare data from the three months before and after its deployment. The evaluation focuses on two primary metrics: average triage accuracy after deployment and the percentage reduction in TTE. Triage accuracy is measured by comparing the team initially assigned by TRIANGLE with the final resolving team. TTE reduction is calculated by comparing the average TTE in the three months following TRIANGLE's deployment with the average from the three months prior.

The empirical results, summarized in Table I, demonstrate the significant positive impact of TRIANGLE on incident management workflows. Across the six production teams, TRIANGLE generally shows high triage accuracy. Notably, Teams C, E, and F achieve outstanding accuracy rates of 96%, 96%, and 97%, respectively. Team A also performs well, with a triage accuracy of 92%. While Team B's accuracy is slightly lower at 82%, it still reflects a competent level of automated triage. Team D shows a more modest accuracy of 64%. A manual inspection reveals that the relatively modest accuracy for Team D (64%) was primarily due to the templated nature of its incident descriptions, which are typically generated by monitoring tools. These descriptions often lack sufficient contextual information, such as detailed incident logs, making it challenging for TRIANGLE to perform accurate triage.

In terms of operational efficiency, measured by TTE reduction, TRIANGLE delivers substantial improvements across all teams. Team B experiences the most dramatic impact, with a 91% reduction in TTE. This suggests that even with slightly lower triage accuracy, automation by TRIANGLE significantly streamlines the initial engagement process. Team D also sees a strong reduction of 72%, followed by Team F (67%), Team E (61%), and Team C (48%). Even Team A, which records the smallest improvement, still benefits from an 18% reduction.

These results highlight a key strength of TRIANGLE: its ability to not only accurately route incidents but also to drastically shorten the critical initial period before an incident receives attention.

D. RQ2: Overall Performance

To verify the effectiveness of TRIANGLE in a real-world scenario, we compare the end-to-end incident triage performance of TRIANGLE with other baseline methods. Based on the maximum hop count of manual triage in historical incident data, we evaluated the accuracy for hop counts ranging from 1 to 5. It is worth noting that triage models based on traditional machine learning methods in DeepTriage [7] (ContentBased, InvertedIndex, LGBM and MART) are unable to perform continuous triage. In contrast, we use the same discussion text as DeepCT to achieve continuous triage results. As a result, their hop accuracy does not vary across different hops. Additionally, because DeepCT requires manually provided enriched discussions, we sequentially provided DeepTriage with the manually added enriched discussions from the incident data in chronological order. In contrast, our method did not use manually provided enriched discussions, but instead utilized the Team Manager for automatic generation. The experimental results are shown in Table II.

According to the experimental results shown in Table II, our proposed model, TRIANGLE, shows outstanding performance compared to traditional machine learning methods and the state-of-the-art method, DeepCT in end-to-end incident triage. Besides, TRIANGLE shows a significant improvement in accuracy as the hop count increases. This highlights its capability to handle complex scenarios involving multiple reassignment hops effectively.

For hop counts up to 1, TRIANGLE achieves an accuracy of 54.7%, surpassing all other methods, including DeepCT, which

TABLE II: End-to-end performance comparison of TRIANGLE and baseline methods in a maximum of 5 transfer hops.

Method	Hop Accuracy [%]				
	Hop 1	Hop 2	Hop 3	Hop 4	Hop 5
ContentBased	9.43	17.3	21.9	25.4	29.6
InvertedIndex	14.4	24.8	34.4	42.8	49.4
LGBM	3.12	3.65	4.66	5.11	5.96
MART	4.23	6.17	7.28	10.22	13.56
DeepCT	43.4	54.6	60.4	64.4	67.6
TRIANGLE	**54.7**	**70.4**	**80.5**	**86.0**	**91.7**

TABLE III: Ablation study results of different components of TRIANGLE on Hop Accuracy.

Method	Hop Accuracy [%]				
	Hop 1	Hop 2	Hop 3	Hop 4	Hop 5
w/o ST	49.1	62.2	76.8	81.1	86.1
w/o MAT	42.8	54.6	63.4	67.5	70.4
w/o TIE	49.6	60.1	61.7	63.8	65.8
TRIANGLE	**54.7**	**70.4**	**80.5**	**86.0**	**91.7**

stands at 43.4%. As the hop count increases to 5, TRIANGLE maintains its superior performance, reaching an accuracy of 91.7%, a substantial improvement over DeepCT's 67.6%. This demonstrates TRIANGLE's robustness and effectiveness in continuous triage without the need for manually enriched discussions. The automatic generation of enriched discussions by the Team Manager in TRIANGLE plays a crucial role in achieving this enhanced performance, making it a highly effective solution for real-world incident triage scenarios.

Further in-depth analysis reveals that as the Hop Count increases, the performance improvement of DeepCT is less than that of TRIANGLE. We believe this is due to the forgetting phenomenon caused by the GRU model in DeepCT when the sequence length increases. In contrast, TRIANGLE benefits from the powerful memory and comprehension capabilities of the Transformer model in LLM for long sequences. Therefore, its performance is not affected by the increased sequence length when the Hop Count increases.

To assess the robustness and generalization of TRIANGLE, we randomly selected nine different services from the system and evaluated the performance of various incident triage models across these services. The results are illustrated in Fig. 6.

The experimental findings reveal that TRIANGLE consistently achieves superior Hop Accuracy across the majority of services, with notable improvements over baseline methods observed at the 2nd or 3rd hop. This enhancement is attributed to TRIANGLE's multi-agent negotiation mechanism, which effectively aggregates information from multiple teams. This process introduces substantial external information to incidents that initially lack sufficient details, thereby significantly improving triage performance.

Our experiments demonstrate that TRIANGLE excels in end-to-end incident triage performance in real-world scenarios.

E. RQ3: Ablation Study

To evaluate the contribution of each key components in our approach, we conducted an ablation study following the experimental setup of Section IV-D. We removed the semantic distillation (w/o ST), Multi-Agent negoTiation mechanism (w/o MAT), and Team Information Enrichment mechanism (w/o TIE) respectively. Since multi-agent negotiation is the core operation of incident triage, to ensure the normal operation of TRIANGLE after removing the multi-agent negotiation mechanism, we allowed the Triage Decider to directly assign based on the ranking of team candidates. Table III shows our experimental results.

From the experimental results presented in Table III, it is evident that each of the key components in our proposed approach contributes significantly to the overall performance. The results show that removing any of the components leads to a decrease in Hop Accuracy across all Hop count (Hop 1 to Hop 5). Specifically, without the Semantic Distillation (w/o ST), the performance drops notably, achieving only 49.1% Hop 1 accuracy, which is a 5.6% decrease compared to the full model. This drop in performance indicates that semantic distillation is crucial for accurate hop prediction, allowing the system to make more informed decisions based on enriched semantic information.

The most substantial performance degradation is observed when the multi-agent negotiation mechanism is removed(w/o MAT). The accuracy drops to 42.8% for Hop 1, which is almost a 12% reduction compared to the full model. The Hop 5 accuracy also sees a significant decline to 70.4%. This demonstrates that the negotiation mechanism is vital for optimizing the triage decision-making process through collaborative decision-making among agents, rather than relying on a naive ranking approach.

The absence of the Team Information Enrichment mechanism (w/o TIE) also results in a significant reduction in performance. The model's Hop 2 and Hop 5 accuracies decrease by 10.3% and 25.9%, respectively, compared to TRIANGLE. These results confirm that enriched discussion is a key factor for effective incident triage, as it provides crucial context that enhances the decision-making capability of the multi-agent system.

Notably, Team Information Enrichment has the greatest impact on the performance of TRIANGLE. This is because the key reason for the inaccuracy in incident triage is the insufficient amount of information in raw incidents. The role of Team Information Enrichment is to automatically obtain external relevant information through agents, so the introduction of external information has a decisive effect on the performance of incident triage.

In contrast, our proposed method, TRIANGLE, consistently outperforms all ablated versions across all metrics, achieving the highest Hop Accuracy at every threshold. This indicates that the combined use of semantic distillation, multi-agent negotiation, and team information enrichment provides a synergistic effect that leads to superior triage performance.

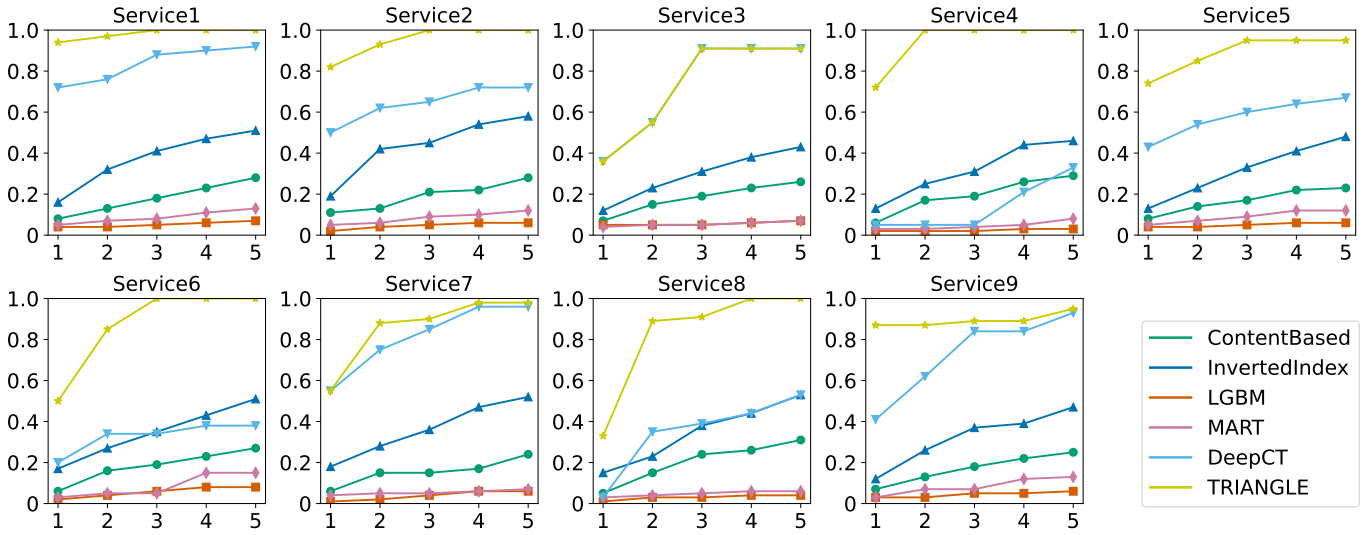

Fig. 6: Effectiveness comparison among TRIANGLE, ContentBased, InvertedIndex, LGBM, MART and DeepCT, for each studied cloud services (the x-axis represents the number of triage hops and the y-axis presents the accuracy of incident triage).

TABLE IV: Bug triage accuracy of TRIANGLE and baseline methods on the MSR 2013 Bug Dataset.

Model	Content Based	Inverted Index	LGBM	MART	DeepCT	TRIANGLE
Acc (%)	38.1	26.9	54.8	53.8	52.1	63.2

F. RQ4: General Capabilities

Although TRIANGLE has been optimized specifically for incident triage tasks, we conducted comprehensive experiments to evaluate its generalizability using the MSR 2013 Bug Dataset [31]. This dataset provides an ideal testing ground due to its extensive collection of bug reports with detailed technical descriptions, comprehensive metadata, and assignment histories similar to incident management workflows.

We selected this dataset for three primary reasons: First, its public availability supports reproducibility. Second, it structurally resembles incident triage data, as both tasks require examining descriptive text, contextual metadata, and historical assignment records. Third, the MSR 2013 Bug Dataset is widely recognized within the bug triage research community [3].

Since the dataset lacks the "team function document" necessary for TRIANGLE, we adopted a semi-manual approach: for each of 20 randomly selected developers, we summarized key assignment phrases based on existing metadata and historical bug resolution records. This approach balanced manual effort with adequate coverage for evaluation across 200 assembled bug cases.

As shown in Table IV, TRIANGLE achieved 63.2% accuracy compared to DeepCT's 52.1% under identical experimental conditions. While both methods showed decreased performance compared to incident triage—likely because bug reports contain less rich textual information and require more domain-specific software knowledge—TRIANGLE maintained significant superiority. We attribute this performance to our method's ability to incorporate historical assignment information and domain-specific knowledge from key phrase summarization, validating TRIANGLE's generalizability to similar technical classification tasks.

V. DISCUSSION

A. Lessons learned

To enhance the accuracy of incident triage, we have identified several key lessons after deploying TRIANGLE to production systems.

First, the completeness and accuracy of historical data are critical. TRIANGLE relies heavily on historical incident triage records as well as each team's functional documentation to perform automated triage. A common failure mode we observed stemmed from missing or insufficient historical signals—such as lacking indicative keywords—that are essential for associating incidents with the correct team. In other cases, the team documentation itself was vague or ambiguous, making it difficult for the system to derive meaningful mappings.

Second, the presence of reasoning-based textual content in historical incident records significantly improves triage performance. For instance, in RQ3, we found that teams with higher triage accuracy often had incident logs that included explicit reasoning (e.g., "after seeing this log, we determined it falls outside our team's scope"). Such information allows the model to better understand the decision-making process and improves its ability to generalize to new incidents.

To address these challenges, we experimented with several strategies to support underperforming teams. These included standardizing documentation formats and terminology to improve textual consistency, and allowing teams to integrate custom systems to provide richer contextual information.

These efforts collectively helped TRIANGLE gain widespread recognition from product teams during its deployment in real-world scenarios.

B. Threat to Validity

Internal validity threats mainly stem from the implementation of our method TRIANGLE and the comparison methods. To mitigate this threat, two authors thoroughly review the code. Specifically, we implement these methods based on a mature industry framework.

External validity threats primarily concern the subjects used. In our study, we employed data from several large-scale cloud service systems. Although these data are derived from real industry applications, the subjects may not fully represent service systems in other companies. In future work, we will apply TRIANGLE to a broader range of service systems.

Construct validity threats primarily lie in the choice of parameters and metrics used. To mitigate the threat from parameters, we employ grid search to optimize the parameters in both TRIANGLE and the comparison methods. To address the threat from metrics, we utilize the most commonly used accuracy and time cost metrics in our study. In future work, we plan to incorporate additional metrics, such as false positive rate and recall, to more comprehensively evaluate the effectiveness and efficiency of TRIANGLE.

VI. RELATED WORK

Incident triage. Recent advancements in incident triage have utilized deep learning to enhance accuracy and efficiency [32]. DeepTriage [7] uses various machine learning models to automate triage, improving accuracy by learning from historical data. The most similar work is DeepCT [3], which performs continuous incident triage using Convolutional Neural Networks (CNNs) to encode domain-specific text and Gated Recurrent Units (GRUs) to extract temporal relationships, complemented by attention mechanisms to reduce noise. Its effectiveness depends on extensive human discussions, which cannot be fully automated. In contrast, our multi-agent based solution can collect troubleshooting information and manage negotiation processes like a human.

Bug triage. Research on bug triage for traditional software is extensive , focusing mainly on two approaches: learning-based and information-retrieval-based methods. Learning-based approaches treat bug triage as a supervised classification problem, using techniques such as ensemble learning [8], and deep learning with Convolutional Neural Networks (CNNs) [9, 10] to classify bugs. Information-retrieval methods focus on leveraging expertise and historical data, with approaches like Latent Dirichlet Allocation (LDA) [11] to match developers to bugs, topic-modeling [12] to map bug report terms to topics, and historical bug-fix analysis [13] to link developers, code components, and bugs. However, incident triage presents a more complex challenge in industry practice because of the intricate nature of cloud systems.

LLM for cloud systems. In recent years, the integration of Large Language Models (LLMs) into cloud systems has gained significant traction, reflecting a broader trend toward enhancing automation and efficiency in cloud operations. Research and practical implementations have demonstrated how LLMs can be leveraged for various tasks, including incident detection [6, 33], assessment [34, 35], and diagnosis [1, 2, 36–39]. For example, RCAgent [40] enhances LLM-generated root cause reports with a Self-Consistency mechanism and domain-specific knowledge integration. ReAct [41] applies LLMs to root cause analysis in cloud management, showing high performance and accuracy with real-world data. DB-GPT [42] merges LLMs with traditional databases to improve natural language query responses, featuring a retrieval-augmented generation system and adaptive learning. To the best of our knowledge, no existing multi-agent solutions have been proposed specifically for incident triage. TRIANGLE is the first end-to-end multi-agent based incident triage approach. Nonetheless, it is quite natural to leverage LLMs to emulate human capabilities in performing triage tasks.

VII. CONCLUSION

Effective and accurate incident triage is crucial for maintaining service quality and reducing time to engagement and mitigation in large-scale cloud service systems. In this paper, we present TRIANGLE, an end-to-end incident triage system designed using a Multi-Agent framework. We introduce a novel semantic distillation mechanism that leverages the powerful semantic understanding capabilities of LLMs to tackle the issue of incident semantic heterogeneity, significantly enhancing triage accuracy. Additionally, we develop a multi-role agent framework equipped with an effective negotiation mechanism, allowing the system to dynamically manage multi-team domain knowledge and simulate the workflow of human engineers. Moreover, TRIANGLE includes an automated team information enrichment mechanism, enabling end-to-end triage without incurring additional human labor costs, even in scenarios requiring incident reassignment. Extensive experiments on real-world incident triage data from a production system serving tens of millions of users demonstrate the strong performance and practical utility of TRIANGLE. The system has improved triage accuracy up to 97% while reducing Time to Engage (TTE) by up to 91%. The deployment of TRIANGLE in a production system serving tens of millions of users in a leading global technology company has shown its effectiveness and reliability in real world environments. We believe that our approach can provide valuable insights and serve as a foundation for future research and development in automated incident triage systems for large-scale cloud services.

References

[1] T. Ahmed, S. Ghosh, C. Bansal, T. Zimmermann, X. Zhang, and S. Rajmohan, "Recommending root-cause and mitigation steps for cloud incidents using large language models," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1737–1749.

[2] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen, J. Zeng, S. Ghosh, X. Zhang, C. Zhang, Q. Lin, S. Rajmohan, D. Zhang, and T. Xu, "Automatic root cause analysis via large language models for cloud incidents," in *Proceedings of the Nineteenth European Conference on Computer Systems, EuroSys 2024, Athens, Greece, April 22-25, 2024*. ACM, 2024, pp. 674–688.

[3] J. Chen, X. He, Q. Lin, H. Zhang, D. Hao, F. Gao, Z. Xu, Y. Dang, and D. Zhang, "Continuous incident triage for large-scale online service systems," in *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*. IEEE, 2019, pp. 364–375.

[4] Z. Yu, C. Pei, S. Zhang, X. Wen, J. Li, G. Xie, and D. Pei, "Autokad: Empowering KPI anomaly detection with label-free deployment," in *34th IEEE International Symposium on Software Reliability Engineering, ISSRE 2023, Florence, Italy, October 9-12, 2023*. IEEE, 2023, pp. 13–23.

[5] Z. Yu, C. Pei, X. Wang, M. Ma, C. Bansal, S. Rajmohan, Q. Lin, D. Zhang, X. Wen, J. Li, G. Xie, and D. Pei, "Pre-trained KPI anomaly detection model through disentangled transformer," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, R. Baeza-Yates and F. Bonchi, Eds. ACM, 2024, pp. 6190–6201.

[6] Z. Yu, M. Ma, C. Zhang, S. Qin, Y. Kang, C. Bansal, S. Rajmohan, Y. Dang, C. Pei, D. Pei, Q. Lin, and D. Zhang, "Monitorassistant: Simplifying cloud service monitoring via large language models," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering, FSE 2024, Porto de Galinhas, Brazil, July 15-19, 2024*, M. d'Amorim, Ed. ACM, 2024, pp. 38–49.

[7] P. Pham, V. Jain, L. Dauterman, J. Ormont, and N. Jain, "Deeptriage: Automated transfer assistance for incidents in cloud services," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA: Association for Computing Machinery, 2020.

[8] L. Jonsson, M. Borg, D. Broman, K. Sandahl, S. Eldh, and P. Runeson, "Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts," *Empirical Software Engineering*, vol. 21, pp. 1533–1578, 2016.

[9] C. d. Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proceedings of COLING*, vol. 25, 2014, p. 69.

[10] S.-R. Lee, M.-J. Heo, C.-G. Lee, M. Kim, and G. Jeong, "Applying deep learning based automatic bug triager to industrial projects," in *Proceedings of the 2017 11th Joint Meeting on foundations of software engineering*, 2017, pp. 926–931.

[11] H. Naguib, N. Narayan, B. Brügge, and D. Helal, "Bug report assignee recommendation using activity profiles," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 22–30.

[12] X. Xia, D. Lo, Y. Ding, J. M. Al-Kofahi, T. N. Nguyen, and X. Wang, "Improving automated bug triaging with specialized topic model," *IEEE Transactions on Software Engineering*, vol. 43, no. 3, pp. 272–297, 2016.

[13] H. Hu, H. Zhang, J. Xuan, and W. Sun, "Effective bug triage based on historical bug-fix information," in *2014 IEEE 25th international symposium on software reliability engineering*. IEEE, 2014, pp. 122–132.

[14] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang, "Large language model based multi-agents: A survey of progress and challenges," *arXiv preprint arXiv:2402.01680*, 2024.

[15] J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan, and O. Press, "Swe-agent: Agent-computer interfaces enable automated software engineering," *Advances in Neural Information Processing Systems*, vol. 37, pp. 50 528–50 652, 2024.

[16] X. Meng, Z. Ma, P. Gao, and C. Peng, "An empirical study on llm-based agents for automated bug fixing," *arXiv preprint arXiv:2411.10213*, 2024.

[17] Y. Wang, W. Zhong, Y. Huang, E. Shi, M. Yang, J. Chen, H. Li, Y. Ma, Q. Wang, and Z. Zheng, "Agents in software engineering: Survey, landscape, and vision," *arXiv preprint arXiv:2409.09030*, 2024.

[18] J. He, C. Treude, and D. Lo, "Llm-based multi-agent systems for software engineering: Literature review, vision and the road ahead," *ACM Transactions on Software Engineering and Methodology*, 2024.

[19] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou *et al.*, "Metagpt: Meta programming for multi-agent collaborative framework," *arXiv preprint arXiv:2308.00352*, vol. 3, no. 4, p. 6, 2023.

[20] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, and M. Sun, "Communicative agents for software development," *arXiv preprint arXiv:2307.07924*, vol. 6, no. 3, 2023.

[21] G. Chen, S. Dong, Y. Shu, G. Zhang, J. Sesay, B. F. Karlsson, J. Fu, and Y. Shi, "Autoagents: A framework for automatic agent generation," *arXiv preprint arXiv:2309.17288*, 2023.

[22] N. Nascimento, P. Alencar, and D. Cowan, "Self-adaptive large language model (llm)-based multiagent systems," in *2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-*

C). IEEE, 2023, pp. 104–109.

[23] G. Li, H. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "Camel: Communicative agents for" mind" exploration of large language model society," *Advances in Neural Information Processing Systems*, vol. 36, pp. 51 991–52 008, 2023.

[24] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "Autogen: Enabling next-gen llm applications via multi-agent conversation framework," *arXiv preprint arXiv:2308.08155*, vol. 3, no. 4, 2023.

[25] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th annual acm symposium on user interface software and technology*, 2023, pp. 1–22.

[26] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving factuality and reasoning in language models through multiagent debate," in *Forty-first International Conference on Machine Learning*, 2023.

[27] H. Yan, S. Ding, and T. Suel, "Inverted index compression and query processing with optimized document ordering," in *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, J. Quemada, G. León, Y. S. Maarek, and W. Nejdl, Eds. ACM, 2009, pp. 401–410.

[28] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 3146–3154.

[29] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[30] Z. Ahmed, S. Amizadeh, M. Bilenko, R. Carr, W. Chin, Y. Dekel, X. Dupré, V. Eksarevskiy, S. Filipi, T. Finley, A. Goswami, M. Hoover, S. Inglis, M. Interlandi, N. Kazmi, G. Krivosheev, P. Luferenko, I. Matantsev, S. Matusevych, S. Moradi, G. Nazirov, J. Ormont, G. Oshri, A. Pagnoni, J. Parmar, P. Roy, M. Z. Siddiqui, M. Weimer, S. Zahirazami, and Y. Zhu, "Machine learning at microsoft with ML.NET," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019, pp. 2448–2458.

[31] A. Lamkanfi, J. Perez, and S. Demeyer, "The eclipse and mozilla defect tracking dataset: a genuine dataset for mining bug information," in *MSR '13: Proceedings of the 10th Working Conference on Mining Software Repositories, May 18--19, 2013. San Francisco, California, USA*, 2013.

[32] J. Chen, X. He, Q. Lin, Y. Xu, H. Zhang, D. Hao, F. Gao, Z. Xu, Y. Dang, and D. Zhang, "An empirical investigation of incident triage for online service systems," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 111–120.

[33] J. Liu, C. Zhang, J. Qian, M. Ma, S. Qin, C. Bansal, Q. Lin, S. Rajmohan, and D. Zhang, "Large language models can deliver accurate and interpretable time series anomaly detection," *arXiv preprint arXiv:2405.15370*, 2024.

[34] P. Jin, S. Zhang, M. Ma, H. Li, Y. Kang, L. Li, Y. Liu, B. Qiao, C. Zhang, P. Zhao, S. He, F. Sarro, Y. Dang, S. Rajmohan, Q. Lin, and D. Zhang, "Assess and summarize: Improve outage understanding with large language models," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, December 3-9, 2023*. ACM, 2023, pp. 1657–1668.

[35] X. Zhou, B. Xu, K. Kim, D. Han, H. H. Nguyen, T. Le-Cong, J. He, B. Le, and D. Lo, "Leveraging large language model for automatic patch correctness assessment," *IEEE Transactions on Software Engineering*, 2024.

[36] Y. Jiang, C. Zhang, S. He, Z. Yang, M. Ma, S. Qin, Y. Kang, Y. Dang, S. Rajmohan, Q. Lin, and D. Zhang, "Xpert: Empowering incident management with query recommendations via large language models," in *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024*. ACM, 2024, pp. 92:1–92:13.

[37] J. Huang, J. Liu, Z. Chen, Z. Jiang, Y. Li, J. Gu, C. Feng, Z. Yang, Y. Yang, and M. R. Lyu, "Faultprofit: Hierarchical fault profiling of incident tickets in large-scale cloud systems," in *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, 2024, pp. 392–404.

[38] W. Zhang, H. Guo, J. Yang, Z. Tian, Y. Zhang, C. Yan, Z. Li, T. Li, X. Shi, L. Zheng *et al.*, "mabc: multi-agent blockchain-inspired collaboration for root cause analysis in micro-services architecture," *arXiv preprint arXiv:2404.12135*, 2024.

[39] C. Pei, Z. Wang, F. Liu, Z. Li, Y. Liu, X. He, R. Kang, T. Zhang, J. Chen, J. Li *et al.*, "Flow-of-action: Sop enhanced llm-based multi-agent system for root cause analysis," in *Companion Proceedings of the ACM on Web Conference 2025*, 2025, pp. 422–431.

[40] Z. Wang, Z. Liu, Y. Zhang, A. Zhong, L. Fan, L. Wu, and Q. Wen, "Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models," *arXiv preprint arXiv:2310.16340*, 2023.

[41] D. Roy, X. Zhang, R. Bhave, C. Bansal, P. Las-Casas, R. Fonseca, and S. Rajmohan, "Exploring llm-based agents for root cause analysis," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, 2024, pp. 208–219.

[42] S. Xue, C. Jiang, W. Shi, F. Cheng, K. Chen, H. Yang,

Z. Zhang, J. He, H. Zhang, G. Wei *et al.*, "Db-gpt: Empowering database interactions with private large language models," *arXiv preprint arXiv:2312.17449*, 2023.