ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA241032

Nissist: An Incident Mitigation Copilot Based on Troubleshooting Guides

Kaikai An[a,*], Fangkai Yang[b,], Junting Lu[a,*], Liqun Li[b], Zhixing Ren[b], Hao Huang[b], Lu Wang[b], Pu Zhao[b], Yu Kang[b], Hua Ding[b], Qingwei Lin[b], Saravan Rajmohan[b], Dongmei Zhang[b] and Qi Zhang[b]**

[a]Peking University
[b]Microsoft

Abstract. Effective incident management is pivotal for the smooth operation of Microsoft cloud services. In order to expedite incident mitigation, service teams gather troubleshooting knowledge into Troubleshooting Guides (TSGs) accessible to On-Call Engineers (OCEs). While automated pipelines are enabled to resolve the most frequent and easy incidents, there still exist complex incidents that require OCEs' intervention. In addition, TSGs are often unstructured and incomplete, which requires manual interpretation by OCEs, leading to on-call fatigue and decreased productivity, especially among new-hire OCEs. In this work, we propose *Nissist* which leverages unstructured TSGs and incident mitigation history to provide proactive incident mitigation suggestions, reducing human intervention. Leveraging Large Language Models (LLM), Nissist extracts knowledge from unstructured TSGs and incident mitigation history, forming a comprehensive knowledge base. Its multi-agent system design enhances proficiency in precisely discerning OCE intents, retrieving relevant information, and delivering systematic plans consecutively. Through our user experiments, we demonstrate that Nissist significantly reduce Time to Mitigate (TTM) in incident mitigation, alleviating operational burdens on OCEs and improving service reliability. Our webpage is available at https://aka.ms/nissist.

1 Introduction

In the rapidly evolving landscape of cloud operation, incident management stands as a pivotal challenge for enterprise-level cloud service providers [13, 3, 17] such as Microsoft, Google, and Amazon. The profound impact of incidents, exemplified by notable events such as the Amazon outage [15], underscores the need for a robust incident management system. Incidents can range from minor operational interruptions to severe system failures, with potential consequences including financial loss, operational disruption, reputational harm, and legal complications. Swiftly identifying, troubleshooting, and resolving system incidents is essential for maintaining service reliability and operational continuity [12, 11]. While automated pipelines can handle low-severity incidents due to their simplicity and commonality, high-severity incidents require immediate and hands-on intervention by On-Call Engineers (OCEs), beyond the capabilities of automated systems. Service teams address this challenge by documenting frequent troubleshooting steps in Troubleshooting Guides (TSGs), empowering OCEs to efficiently resolve incidents [8, 6].

To investigate the effect of TSGs on incident mitigation, we analyze around 1000 high-severity incidents in the recent twelve months that demand immediate intervention from OCEs. Consistent with findings from prior studies [8, 18, 9], which demonstrate the efficacy of TSGs in incident mitigation. We found that incidents paired with TSGs exhibit a 60% shorter average time-to-mitigate (TTM) compared to those without TSGs, emphasizing the pivotal role played by TSGs. This trend is consistent across various companies, as evidenced by research [14, 10], even among those employing different forms of TSGs. However, despite their utility, as highlighted by [18, 2], the unstructured format, varying quantity, and propensity for internal use purpose of TSGs, impede their optimal utilization. Particularly, such unstructured TSGs pose challenges for new hires and contribute to the complexity of the incident mitigation process, especially in scenarios requiring coordination across multiple teams. In addition, some TSGs are outdated, lacking the most recent knowledge of incident mitigation. The incident mitigation history provided by OCEs in the internal incident management platform serves as another valuable resource for extracting incident mitigation knowledge.

Recent works have focused on leveraging TSGs to facilitate incident mitigation process. [18] fine-tunes models to extract knowledge from TSGs, while [9, 2] identify relevant TSGs in root cause analysis. However, the prevalent unstructured nature of existing TSGs limits the effectiveness of fine-tuning procedures, and the complexity of high-severity incidents still require human interventions. In this work, we propose *Nissist*, aiming to reduce OCE workload and assist incident mitigation processes. Firstly, we establish a set of rigorous TSG criteria to convert unstructured TSGs into structured, high-quality formats leveraging Large Language Models (LLMs), while also providing guidelines for OCEs when documenting new TSGs. Subsequently, we propose a novel structure of knowledge base comprising discrete executable nodes extracted from TSGs and incident mitigation history. Moreover, we introduce an advanced multi-agent system [24] designed to proficiently interpret queries, retrieve relevant knowledge nodes, and formulate actionable plans in a semi-automated manner. By interacting with Nissist, the incident mitigation trajectories are optimized, allowing OCEs to focus on challenging mitigation steps not covered by TSGs and mitigation history, thus significantly reducing direct human intervention.

* This work is done during the internship at Microsoft.
** Corresponding author

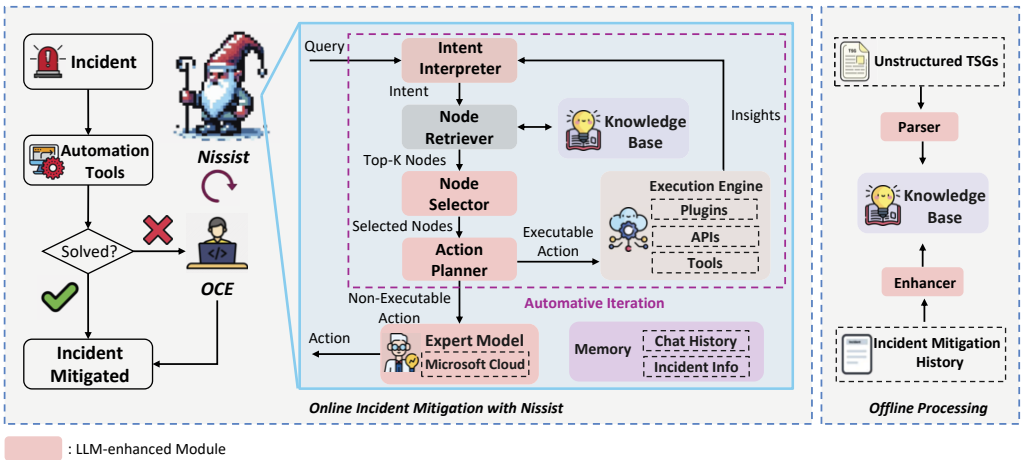

Figure 1: The Semi-Automated Incident Mitigation Framework with Nissist. When incidents exceed automation capabilities, OCEs engage in iterative interactions with Nissist. Nissist interprets OCE intents, retrieves knowledge from the knowledge base, and formulates actions. Executable action is conducted with the execution engine, generating insights for the next round node retrieval and action planning in an automative iteration manner (purple dashed box). Actions that cannot be carried out by the execution engines are then delegated to OCEs for manual execution. The knowledge base is built offline with LLM-extracted knowledge from unstructured TSGs and mitigation history.

2 System Overview

As illustrated in Figure 1, OCEs engage in iterative interactions with Nissist when automation tools are insufficient to address an incident. Initially, Nissist constructs *Knowledge Base* offline by parsing knowledge from unstructured TSGs and enhances it with knowledge from incident mitigation history not covered in TSGs. Subsequently, it iteratively processes OCE queries. Nissist is designed to mitigate incidents in a fully automated manner. However, due to the complex nature of incidents, not all actions suggested by Nissist can be automatically executed due to the lack of related execution functions in the execution engine. Thus, the non-executable actions are delegated to OCEs for manual execution, while those executable actions are passed to the execution engine, generating insights to trigger the next step of mitigation, automating the mitigation iteration (purple dashed box in Figure 1). Each module, powered by LLMs, serves as an agent responsible for specific tasks, including interpreting intents, selecting the most relevant knowledge (nodes from knowledge base), suggesting actions, etc. These modules concurrently communicate with each other to efficiently mitigate incidents.

2.1 Constructing the Knowledge Base

The primary source of knowledge is derived from unstructured TSGs, which typically encompass information on investigating and mitigating incidents. However, their unstructured nature poses challenges for traditional data retrieval methods, as the appropriate action may not always exhibit semantic or lexical similarity to the query due to unstructured nature. Additionally, incident mitigation often requires a sequential steps of actions. Chunking TSGs in retrieval methods can result in the disruption of this sequence, particularly when certain steps are spread across multiple TSGs and collectively represent the entire flow of steps. To address this, we develop quality criteria and use LLMs to reformat original TSGs into structured ones, including background, terminology, FAQ (frequently asked questions), flow, and appendix. In particular, "flow" represents sequential steps of actions. We then construct a *knowledge base* comprising knowledge *nodes*. Each node is formated in JSON consisting of type, intent, action, and linker. "Intent" describes the purpose of the node, and it is used as the indexing context [4] which is later used to retrieve the entire node. "Linker" connects the outcomes of taking current actions to the intents of next step nodes.

Parsing the structured TSGs with LLMs yields this knowledge base that facilitates easy retrieval of relevant nodes for actionable plans. Additionally, the node-level granularity enables discovery of new connections among TSGs, including cross-team mitigation flows not presented in any raw TSGs (refer to Cross-TSG case in Section 3). Microsoft maintains an incident management platform where OCEs can document and collaborate during incident mitigation. Beyond unstructured TSGs, incident mitigation history serves as an additional source of knowledge. To complement any outdated or missing steps in TSGs, we have designed an enhancer that captures the latest solutions from these discussions on mitigation history.

2.2 Muti-Agent System

Intent Interpreter. This module is crucial for understanding OCE's intent and determining whether Nissist's intervention is necessary during the conversation. It guides OCEs to incident troubleshooting topics and helps refine and clarify their input. If needed, it seeks confirmation from OCEs for intent clarification.

Node Retriever & Selector. The node retriever module retrieves relevant nodes from the knowledge base concerning the clarified intent. Unlike traditional retrievers [5], which index documents or chunks, we use the clarified intent as the query to retrieve top-k nodes by comparing with indexed "Intent" of each node. To enhance fault tolerance, the node selector selects the most relevant ones from retrieved top-k nodes. This is crucial as semantic discrepancies may exist despite specific matched keywords. By sourcing information from multiple nodes, Nissist enriches the knowledge context for subsequent actions. If no relevant node is found, it indicates current incident exceeds Nissist's scope and informs OCEs for intervention.

Action Planner. This module serves as the central and critical component within Nissist, recommending appropriate actions based on the selected nodes and memory. Unlike SOTA LLM planners that automate reasoning, actions, and observations in an interleaved manner [26, 21, 7], our domain finds this planning style unsuitable. Fully automated plugins or tools cannot handle all incidents due to their complexity, risking wrong plugin invocation or omission of execution steps by the planning module. Hence, a semi-automated miti-

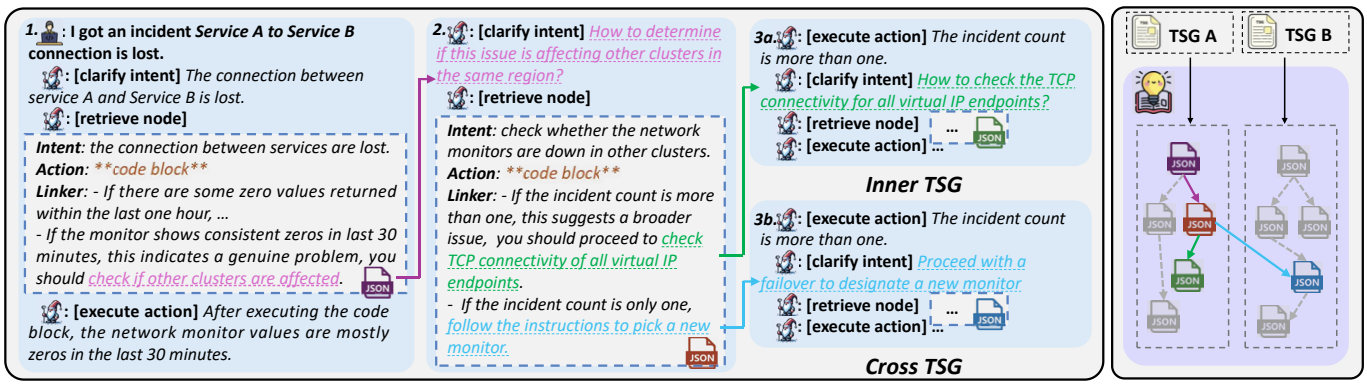

Figure 2: A use case demonstrates that Nissist mitigates the connection lost incident between *Service A* and *Service B*. For simplicity, only the first three iterations are presented. 3*a* & 3*b* show two different mitigate paths due to two different execution results. In particular, 3*b* indicates that Nissist can leverage knowledge cross TSG (the blue-colored node is extracted from another TSG).

gation process occasionally requiring human intervention is necessary for the security purpose. Action planner generates steps based on incident complexity and plugin availability. For incidents can be covered in TSGs, exhaustive step-by-step and manual planning is unnecessary. Instead, Nissist suggests sequential steps automatable by execution engine, thereby bypassing the need for OCE input. The execution engine should be able to execute the action and analyze the execution outcomes in order to give insights. The execution engine could be available plugins, APIs or LLM-based code generator [16].

Post Processor. In open-domain planning scenarios, self-reflection of planned strategies against actual observations with LLMs is practical and efficient [19, 25]. However, its utility is somewhat limited in incident mitigation due to the potential for hallucinations resulting from the lack of domain-specific knowledge in pre-trained LLMs [23, 22], such as GPT-4 [1]. To address this limitation, we integrate a pre-trained LLaMA2 model [20] as the expert model that has undergone supervised fine-tuning (SFT) on a corpus of Microsoft Cloud documentation [23]. The expert model enhances the post-processing procedure by providing informed corrections rooted in cloud domain expertise.

3 User Experiments and Case Study

To validate the advantages of Nissist, we conducted a human evaluation involving twenty OCEs[1] and five[2] incidents. These incidents are categorized as simple or hard based on their mitigation history. Note that simple incidents here are not easy ones that can be handled with automated mitigation tools. Each OCE is tasked with mitigating all five incidents. To ensure a fair comparison, only one mitigation approach is assigned to each incident, either with Nissist or manual mitigation. We distribute the incidents to ensure each incident gets an equal number of Nissist mitigation and manual mitigation. We demonstrate the effectiveness of Nissist and report several metrics including Success Rate (SR): whether the incident can be mitigated without human intervention; Human Intervention (IR): steps need human intervention; Turns: numebr of mitigation turns; TTM reduction (TTM ↓) compared with manual mitigation.

Table 1 shows a significant improvement in TTM reduction compared to manual mitigation which either requires substantial mitigation experience or involve navigating through unstructured TSGs. Specifically, Nissist achieves a TTM reduction of 98.93% for simple incidents and 94.85% for hard ones. Hard incidents requires additional turns for mitigation due to their complex nature. Nissist shows a 77.19% full automation SR for simple incidents and 52.63% SR for hard ones, demonstrating a notable reduction of manual efforts. With some minor human intervention (11.28% for simple incidents and 15.79% for hard ones), incidents can be effectively mitigated.

Table 1: User experiment results on metrics.

Category	SR	IR	Turns	TTM ↓
Simple	77.19%	11.28%	2.56	98.93%
Hard	52.63%	15.79%	5.74	94.85%

Figure 2 illustrates a use case demonstrating industrial practices with Nissist. Given an OCE query "*Service A to Service B connection is lost*", Nissist interprets the query, identifies the intent, and uses it to retrieve and select the most relevant node. The action planner then fills the given parameters, such as service information, into the code block, *i.e.*, a Kusto query with parameter placeholders in this use case. After the action is passed to the execution engine, the outcome indicates "*the network monitor values are mostly zeros in the last 30 mins*". Nissist correlates this outcome with the "Linker" in the retrieved node, where this outcome indicates "*a genuine problem and should check if other clusters are affected*". Then Nissist generates a new intent "*How to determine if this issue is affecting other clusters*" for the next round of interaction automatically. This interaction continues until the incident is mitigated or requires human intervention. Additionally, Nissist digests all TSGs into knowledge base, making it possible to discover connections between nodes located in different TSGs. For example, 3*b* in Figure 2 demonstrates another execution result which requires knowledge from a different TSG. Previously, it requires OCEs to take great efforts searching for the knowledge in other TSGs, which often did not list such knowledge in their titles.

4 Conclusion

We address the incident mitigation challenges in Microsoft by optimizing TSG usage and reducing human effort. We introduce Nissist, which leverages LLMs to digest TSGs and incident mitigation history into a knowledge database. We establish a multi-agent system with semi-automation to precisely detect OCE intents, retrieve relevant nodes, and provide stepwise actions. Our experiments show reduced TTM and significantly alleviated OCE workload.

[1] Varying across new-hire and experienced OCEs.
[2] Controlling time to each experiment around 30 mins to 1 hour per OCE ensures consistency, engagement, and ethical treatment.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen, et al. Empowering practical root cause analysis by large language models for cloud incidents. *arXiv preprint arXiv:2305.15778*, 2023.

[3] Z. Chen, Y. Kang, L. Li, X. Zhang, H. Zhang, H. Xu, Y. Zhou, L. Yang, J. Sun, Z. Xu, et al. Towards intelligent incident management: why we need it and how we make it. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1487–1497, 2020.

[4] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou. The faiss library. 2024.

[5] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[6] S. Ghosh, M. Shetty, C. Bansal, and S. Nath. How to fight production incidents? an empirical study on a large-scale cloud service. In *Proceedings of the 13th Symposium on Cloud Computing*, pages 126–141, 2022.

[7] L. Guan, K. Valmeekam, S. Sreedharan, and S. Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36:79081–79094, 2023.

[8] J. Jiang, W. Lu, J. Chen, Q. Lin, P. Zhao, Y. Kang, H. Zhang, Y. Xiong, F. Gao, Z. Xu, et al. How to mitigate the incident? an effective troubleshooting guide recommendation technique for online service systems. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1410–1420, 2020.

[9] Y. Jiang, C. Zhang, S. He, Z. Yang, M. Ma, S. Qin, Y. Kang, Y. Dang, S. Rajmohan, Q. Lin, et al. Xpert: Empowering incident management with query recommendations via large language models. *arXiv preprint arXiv:2312.11988*, 2023.

[10] J. Li, A. Sun, and Z. Xing. Learning to answer programming questions with software documentation through social context embedding. *Information Sciences*, 448:36–52, 2018.

[11] L. Li, X. Zhang, X. Zhao, H. Zhang, Y. Kang, P. Zhao, B. Qiao, S. He, P. Lee, J. Sun, et al. Fighting the fog of war: Automated incident detection for cloud systems. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 131–146, 2021.

[12] Y. Li, X. Zhang, S. He, Z. Chen, Y. Kang, J. Liu, L. Li, Y. Dang, F. Gao, Z. Xu, et al. An intelligent framework for timely, accurate, and comprehensive cloud incident detection. *ACM SIGOPS Operating Systems Review*, 56(1):1–7, 2022.

[13] J. Liu, S. He, Z. Chen, L. Li, Y. Kang, X. Zhang, P. He, H. Zhang, Q. Lin, Z. Xu, et al. Incident-aware duplicate ticket aggregation for cloud systems. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 2299–2311. IEEE, 2023.

[14] R. Lotufo, Z. Malik, and K. Czarnecki. Modelling the 'hurried'bug report reading process to summarize bug reports. *Empirical Software Engineering*, 20:516–548, 2015.

[15] C. O'Donovan. Amazon's cloud computing service recovers from outage. Accessed 23 June 2023. https://www.washingtonpost.com/technology/2023/06/13/amazon-aws-down-outage/, 2023.

[16] B. Qiao, L. Li, X. Zhang, S. He, Y. Kang, C. Zhang, F. Yang, H. Dong, J. Zhang, L. Wang, et al. Taskweaver: A code-first agent framework. *arXiv preprint arXiv:2311.17541*, 2023.

[17] S. R. Sarkar, R. Mahindru, R. A. Hosn, N. Vogl, and H. V. Ramasamy. Automated incident management for a {Platform-as-a-Service} cloud. In *Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE 11)*, 2011.

[18] M. Shetty, C. Bansal, S. P. Upadhyayula, A. Radhakrishna, and A. Gupta. Autotsg: Learning and synthesis for incident troubleshooting. ESEC/FSE 2022, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450394130. doi: 10.1145/3540250.3558958. URL https://doi.org/10.1145/3540250.3558958.

[19] N. Shinn, B. Labash, and A. Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.

[20] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[21] K. Valmeekam, M. Marquez, S. Sreedharan, and S. Kambhampati. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005, 2023.

[22] C. Wang, X. Liu, Y. Yue, X. Tang, T. Zhang, C. Jiayang, Y. Yao, W. Gao, X. Hu, Z. Qi, et al. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *arXiv preprint arXiv:2310.07521*, 2023.

[23] Z. Wang, F. Yang, P. Zhao, L. Wang, J. Zhang, M. Garg, Q. Lin, and D. Zhang. Empower large language model to perform better on industrial domain-specific question answering. *arXiv preprint arXiv:2305.11541*, 2023.

[24] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. 2023.

[25] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[26] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.