International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 06 | June - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

Exception Logger Utility: Contextual Error Tracking for Enhanced Software Application Debugging

Varadraj Patil[1*], Dr. Deepamala N [2*], Dandavati Suhas[3*], Prof. Rashmi R[4*]

[1,2]Computer Science and Engineering, RV College of Engineering, Bengaluru, India

[3,4]Information Science and Engineering, RV College of Engineering, Bengaluru, India

---

Abstract - Software programs that encounter errors are considered to be in an erroneous state. Exception logging is a critical aspect of software development that plays a pivotal role in ensuring the stability and reliability of applications. When unexpected errors occur during program execution, logging these exceptions provides valuable insights into the root causes, facilitates debugging, and aids in developing robust software solutions. To facilitate software sustenance and debugging, developers incorporate logging statements to generate logs and capture system execution details. However, determining the appropriate placement of these logging statements is a challenging task. On one hand, insufficient logging can impede maintenance efforts by omitting crucial system execution data. On the other hand, excessive logging can inundate the logs, obscuring the true issues and significantly degrading performance. Therefore, in the proposed system, an exception logging framework is developed to offer developers the flexibility to log necessary objects when exceptions occur. For instance, one such example would be logging the request object of an API.

Key Words: Exception Logging, API Request Object, Database, Software maintenance, Debugging

I. INTRODUCTION

Software logs are frequently used in software systems to document how the system is executed. The generated logs are used by developers to help with a variety of activities, including debugging, testing, program understanding, system verification, and performance analysis. Software logs are used by developers for a wide range of functions, including performance analysis, program comprehension, testing, and debugging. Despite the significance of logs, earlier research reveals that there is no industry standard for the formulation of logging statements. While logs are normally assessed in tandem, recent research on logs frequently just evaluates the appropriateness of a log as an individual item (for example, one single logging statement). The logging system explained in this paper can be extended to any standalone application.

II. OVERVIEW

To capture system runtime data, developers add logging lines to the source code. The resulting logs are then used to aid in software debugging and maintenance. Logs are printed with complete stack trace. Currently whenever an exception occurs developers need to log at each exception point. As there can be multiple exception handling, it becomes tedious and results in fewer logs. To resolve issues in production DEBUG needs to be enabled to figure out the exact context for which the error occurred and needs to take assistance to reproduce this issue. Enabling DEBUG slows down the service and also we need to wait for the issue to be reproduced. This is a time-consuming process and delays issue resolution. If we can log the full error context which includes the request received and any intermediate objects required for debugging So, the framework would then log the objects added as part of the request scoped in the Exception Handler.

III. RELATED WORKS

A survey on logging practices of exception stack traces in open-source Java projects, the study in the paper [1] sheds light on current developer practices and provides valuable recommendations for improving the logging process. The survey emphasises the need for careful consideration when logging stack traces and advocates for the development of comprehensive guidelines at the organizational or global level. Furthermore, the survey encourages logging frameworks to enhance flexibility to meet diverse logging requirements.

A comprehensive understanding of the considerations that developers undertake when evaluating logging benefits and costs has been discussed in paper [2] . By highlighting the ad hoc balancing of the factors, it emphasizes the need for improved logging practices and the development of more flexible logging capabilities. The review serves as a valuable resource for developers, researchers, and logging library designers seeking to optimize the benefits of logging while minimizing its costs in software development.

The challenges developers face when determining logging locations has been discussed in the paper [3] and proposes a deep learning-based approach for finer-grained logging location suggestions. The survey demonstrates the potential of leveraging syntactic information in the source code and provides insights into sharing logging location suggestions across systems. The findings emphasize the need for further research to combine syntactic and semantic information for improved logging location recommendations.

The significance of logging in performance monitoring and modeling, particularly in web-based systems has been discussed in paper[4]. It explores the proposed approach for

automatically suggesting logging statement locations to support performance monitoring. The review emphasizes the effectiveness of the approach in improving statistical performance models and influencing system performance, showcasing its applicability in both open-source and commercial systems. Integrating the approach into the release pipeline allows practitioners to benefit from periodic logging suggestions for continuous system improvement.

Duplicate logs are yet another problem as there are unnecessary logs and debugging becomes difficult, therefore a study was conducted on all logging statements, defined as logging statements having the same static text message[5].

A server log is a file that is created and maintained by a server that consists of a list of activities that are performed by the server. An AI based approach has been proposed in paper [6] which visualises the logs and provides the knowledge-based solution to the developer troubleshooting the root cause of the error in the server log.

At present developers rely on their intuition while performing the logging activities. There aren't any logging guidelines, hence in the paper [7] a study was conducted on Logging-Code-Issue- Introducing changes in six popular large-scale Java-based open source software systems.

A study on existing deep learning frameworks and how the platforms influence the development and deployment of deep learning software has been conducted in the paper [8]. A Commenting suggestion method has been demonstrated in the paper [9] where machine learning techniques are applied to identify the possible locations where commenting can be done.

An automatic log based method has been proposed in the paper [10] which identifies cloud behaviors which are resulted from the failed executions of the cloud operating system for failure diagnosis. In the paper [11] , a system for analyzing web server logs has been proposed which uses Hadoop and MapReduce that figures out the execution time.An automated approach has been proposed in paper [12] which is known as LogCoCo which estimates the code coverage measures using the execution logs which are available and the results indicate that this approach can be used to evaluate and improve the test suites which are existing.

An empirical study has been done on log-related issues in the paper [13] and conveys that files with log-related issues have undergone frequent bug fixes and changes. An automated tool has been developed that detects four types of log-related issues. The focus in the paper [14] is on automated log parsing for large-scale log analysis in modern systems. The paper presents a comprehensive study of four representative log parsing methods, evaluating their accuracy, efficiency, and effectiveness for subsequent log mining tasks. Based on the study's findings, the authors propose a parallel log parsing method called POP. POP utilizes heuristic rules and a hierarchical clustering algorithm, optimized on top of Spark by employing tailored functions for selected Spark operations. Extensive experiments are conducted on synthetic and real-world datasets, demonstrating that POP performs accurately and efficiently on large-scale log data. The paper also emphasizes the release of POP and the four studied log parsers to facilitate future research and promote their reusability in the field.

Through a comprehensive replication study of logging practices in Java projects in the paper [15], this literature survey confirms the prevalence of logging and active maintenance of logging code. It also reveals differences compared to C/C++ based systems, such as longer bug report resolution time and more consistent updates to log printing code.

The importance of logging code quality and the challenges associated with its development and maintenance is discussed in paper [16]. The paper's findings on logging anti-patterns and the introduction of LCAnalyzer provide valuable insights and practical tools for developers to enhance the quality and effectiveness.

Through a large-scale analysis of try/catch blocks in Java projects hosted on GitHub, the study in the paper [17] provides empirical evidence of current exception handling practices. The findings highlight prevalent issues such as local handling, code duplication, and bad practices. The misjudgment of risk and the tension between local and program-wide exception handling further underscore the need for improvements. The study suggests that future tools, capable of suggesting complete handlers and promoting positive handling policies, could address some of these challenges. Code smells may be signs of poor design and implementation decisions, which may have an impact on the capacity to maintain [18], comprehend, and execute software systems. Studies have been suggested to detect code smells in order to lessen their negative effects [19][20]. Developers who copy and paste a piece of code from one place to another may result in duplicate code (also known as code clones) [19]. Such code clones could be an indication of poor quality. Numerous research [21] have been done that concentrate on understanding and identifying code clones.

The paper [22] explores the design and implementation of a configurable error logging framework for web applications. The paper highlights the advantages of this framework, including the ability to remotely configure logging settings, eliminating the need for application developers to specify log settings. The framework provides a graphical user interface (GUI) for administrators to modify logging settings stored in an XML file, enabling flexible configuration. Additional features such as sending log messages to a remote machine for storage, sending log information via email, and integrating with Chainsaw for log information querying are also discussed. The proposed framework stands out by allowing granular configuration of logging settings, contrasting with static logging in conventional MVC frameworks.

The importance of web data classification and categorization in managing the vast amount of information available on the web was discussed in paper [23]. The role of web server log files as a valuable resource for web mining is highlighted, along with an overview of their types and formats. The comparative analysis of log file formats offers guidance for selecting the most appropriate format for efficient web mining processes. Ultimately, this literature review sets the foundation for further research and advancements in the field

of web data mining. The importance of the information library of active nodes within the Workflow Exception Handling System and its contributions to exception tracking, resolution, and efficient workflow design are discussed in paper [24]. The survey highlights the complementary nature of the subsystems for exception warning and exception handling. Overall, the survey provides insights into the functionality and benefits of Workflow Exception Handling System in managing exceptions within hierarchical modeling frameworks, setting the stage for further research and advancements in this domain.

The literature review in the paper [25] concludes by summarizing the key findings and contributions of the practical approach for exception handling design rule checking in software product lines. It emphasizes the significance of employing JUnit test cases and dynamic mock objects for comprehensive testing. The review highlights the benefits of the approach, as demonstrated through the case study on the Mobile Media product line. It underscores the approach's ability to improve system confidence, support maintainability tasks, and effectively uncover bugs in exception handling code.

IV. METHODOLOGY

As intended, to build an Exception logging framework that would provide flexibility to developers to log required objects on Exception, it is important to understand the flow and decide which object is important to be logged.

To further proceed with the project a class with two methods; one to add the logs into a data structure (to store the objects that are to be logged) and the other to print the logs in case of an exception, was created. The framework is built to function in request-scoped i.e. once the program enters a new API the objects stored in the data structure concerning the previous API are cleared, to achieve this dependency injection is used, and in case of an exception all the objects added in are printed concerning context. ExceptionLogger is used to print the logs concerning the context and the stack trace. The objective behind implementing request-scoped is to avoid any issues concerning memory and performance and printing logs in request-scoped would result in lesser and more efficient and important logs which is one of the main problems of traditional logging systems.

System consist of 2 modules which consist of namely AddObjects Module and LogObjects Module.

AddObject module's purpose to add the logs into a data structure (to store the objects that are to be logged). The critical object or the object requested is the input for the module and the output consists of Objects added into the data structure in the requested scope. Once the program enters a new API the objects stored in the data structure concerning the previous API are cleared. Fig 2 depicts the functionality of the AddObject module.

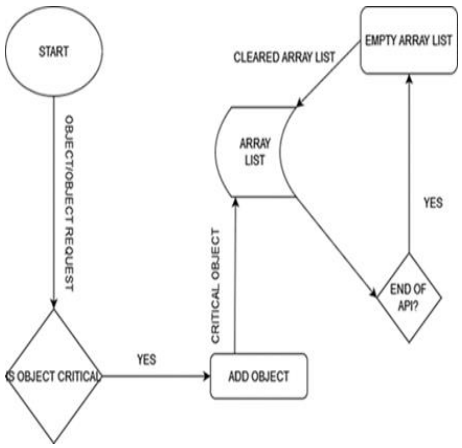

Fig 1: AddObject Module

LogObject Module's purpose is to print the logs in case of an exception. Objects in the data structure are the input to the module. Logs are printed as output in a format that is easily understandable to the developer which has the complete stack trace. In case of an exception, all the objects added in, are printed concerning the context. ExceptionLogger is used to print the logs concerning the context. Fig 2 depicts the functionality of the LogObject module.

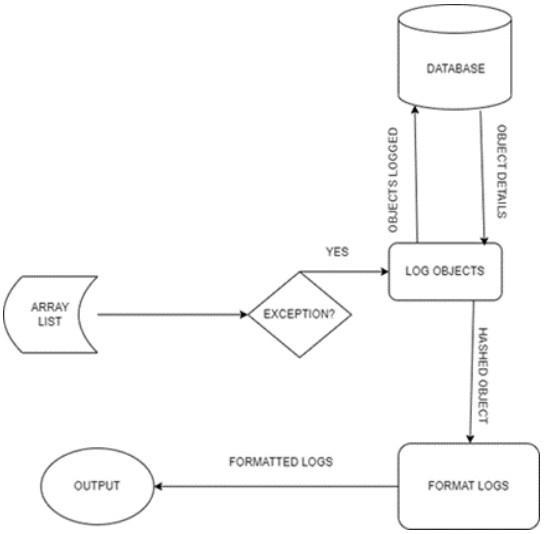

Fig 2: LogObjects Module

V. CONCLUSION

A bug could bring on other faults in the application itself, issues with communication with the server, or issues with database access. To aid in software maintenance and to debug, developers provide logging statements that create logs and record system execution characteristics. But choosing where to put logging statements is a critical but difficult assignment.

This proposed system can help organizations and companies of any size to analyze and understand the logs and debug the error in an effective way. Exception logging frameworks often include features for monitoring and alerting. They can automatically notify developers or system administrators when critical errors occur, ensuring that problems are addressed promptly. This helps in maintaining the overall health and stability of the application.

Exception logging frameworks provide a centralized repository for storing and analyzing exception data. This allows developers to track and manage exceptions across multiple components or instances of an application. It facilitates the identification of recurring issues and helps prioritize bug fixes and improvements. By logging exceptions in a production environment, developers gain insights into the real-world behavior of their application. They can analyze patterns, detect edge cases, and uncover scenarios that were not anticipated during development. This feedback loop contributes to iterative improvements and enhances overall software quality.

In conclusion, an exception logging framework is a valuable tool for capturing, analyzing, and managing exceptions in software applications. It enhances debugging capabilities, facilitates error monitoring and alerting, provides centralized error tracking.

REFERENCES

[1] H. Li, H. Zhang, S. Wang and A. E. Hassan, "Studying the Practices of Logging Exception Stack Traces in Open-Source Software Projects," in IEEE Transactions on Software Engineering, vol. 48, no. 12, pp. 4907-4924, 1 Dec. 2022, doi: 10.1109/TSE.2021.3129688.

[2] H. Li, W. Shang, B. Adams, M. Sayagh and A. E. Hassan, "A Qualitative Study of the Benefits and Costs of Logging From Developers' Perspectives," in IEEE Transactions on Software Engineering, vol. 47, no. 12, pp. 2858-2873, 1 Dec. 2021.

[3] Zhenhao Li, Tse-Hsun (Peter) Chen, Weiyi Shang, "Where ShallWe Log? Studying and Suggesting Logging Locations in Code Blocks", International Conference on Automated Software Engineering (ASE), (2020).

[4] Yao, K., de Pádua, G.B., Shang, W. *et al.* Log4Perf: suggesting and updating logging locations for web-based systems' performance monitoring. *Empir Software Eng* 25, 488–531 (2020).

[5] Zhenhao Li, Tse-Hsun (Peter) Chen, Jinqiu Yang, Weiyi Shang, "DLFinder: Characterizing and Detecting Duplicate Logging Code Smells", International Conference on Software Engineering (ICSE), (2019).

[6] Pratik Padman, Atharva Narlawar, Priya Surana, Roshan Kasliwal, Mayuri Sonwale, "AI-Powered System Providing Knowledge-Based Solution for Errors in Server Logs", International Conference On Computing, Communication, Control And Automation (ICCUBEA), (2019).

[7] Boyuan Chen, Zhen Ming (Jack) Jiang, "Extracting and studying the Logging-Code-Issue- Introducing changes in Java-based large-scale open-source software systems", Empirical Software Engineering 24, 4, (2019), 2285–2322.

[8] Qianyu Guo, Sen Chen, Xiaofei Xie, Lei Ma, Qiang Hu, Hongtao Liu, Yang Liu, Jianjun Zhao, Xiaohong Li, "An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms", IEEE/ACM International Conference on Automated Software Engineering (ASE), (2019).

[9] Yuan Huang, Xinyu Hu, Nan Jia, Xiangping Chen, Yingfei Xiong, Zibin Zheng, "Learning Code Context Information to Predict Comment Locations", IEEE Transactions on Reliability, (2019).

[10] Y. Yuan, W. Shi, B. Liang and B. Qin, "An Approach to Cloud Execution Failure Diagnosis Based on Exception Logs in OpenStack," 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), Milan, Italy, 2019, pp. 124-131, doi: 10.1109/CLOUD.2019.00031.

[11] Pranjali Borgaonkar, Gaurav Kumar, Jyoti Yaduwanshi, "Framework for Analyzing Web Access Logs using Hadoop and MapReduce", International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE), (2018).

[12] Boyuan Chen, Jian Song, Peng Xu, Xing Hu, Zhen Ming (Jack) Jiang, "An Automated Approach to Estimating Code Coverage Measures via Execution Logs", IEEE/ACM International Conference on Automated Software Engineering (ASE), (2018).

[13] Mehran Hassani, Weiyi Shang, Emad Shihab, Nikolaos Tsantalis, "Studying and Detecting Log-Related Issues", Empirical Software Engineering, (2018).

[14] P. He, J. Zhu, S. He, J. Li and M. R. Lyu, "Towards Automated Log Parsing for Large-Scale Log Data Analysis," in IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 6, pp. 931-944, 1 Nov.-Dec. 2018, doi: 10.1109/TDSC.2017.2762673.

[15] Boyuan Chen, Zhen Ming (Jack) Jiang, "Characterizing logging practices in Java-based open-source software projects – a replication study in Apache Software Foundation", Empirical Software Engineering 22, 1, (2017), 330–374.

[16] B. Chen and Z. M. Jiang, "Characterizing and Detecting Anti-Patterns in the Logging Code," 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), Buenos Aires, Argentina, 2017, pp. 71-81, doi: 10.1109/ICSE.2017.15.

[17] M. B. Kery, C. Le Goues and B. A. Myers, "Examining Programmer Practices for Locally Handling Exceptions," 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), Austin, TX, USA, 2016, pp. 484-487.

[18] I. Ahmed, C. Brindescu, U. A. Mannan, C. Jensen, A. Sarma, "An Empirical Examination of the Relationship between Code Smells and Merge Conflicts", ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), (2017).

[19] Q. Fu, J. Zhu, W. Hu, J.-G. Lou, R. Ding, Q. Lin, D. Zhang, T. Xie, "Share on Where do developers log? an empirical study on

logging practices in industry", Companion Proceedings of the 36th International Conference on Software Engineering, (2014).

[20] S. L. Abebe, S. Haiduc, P. Tonella, A. Marcus, "The Effect of Lexicon Bad Smells on Concept Location in Source Code", IEEE 11th International Working Conference on Source Code Analysis and Manipulation, (2011).

[21] N. Busany, S. Maoz, "Behavioral Log Analysis with Statistical Guarantees", IEEE/ACM 38th International Conference on Software Engineering (ICSE), (2016).

[22] V. Krishnamurthy, C. Babu, P. M. Krishnan, C. Aravindan, S. Balamurugan. "ReCon - Aspect oriented remotely reconfigurable error logging framework for web applications", International Conference on Information Communication and Embedded Systems (ICICES), 2013.

[23] P. Sharma, S. Yadav and B. Bohra, "A review study of server log formats for efficient web mining," 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), Greater Noida, India, 2015.

[24] Weili Kou, Peng Gong, Kai Cai and Jing Wang, "Workflow exception handling system application in hierarchical modeling," 2008 12th International Conference on Computer Supported Cooperative Work in Design, Xi'an, China, 2008, pp. 693-698, doi: 10.1109/CSCWD.2008.4537062.

[25] R. J. Sales Júnior and R. Coelho, "Preserving the Exception Handling Design Rules in Software Product Line Context: A Practical Approach," 2011 Fifth Latin-American Symposium on Dependable Computing Workshops, Sao Jose dos Campos, Brazil, 2011.