A Systematic Review of Logging Practice in Software Engineering

Guoping Rong
Software Institute
Nanjing University
Nanjing,Jiangsu,P.R.China
ronggp@nju.edu.cn

Qiuping Zhang
The department of
computer science and technology
Nanjing University
Nanjing,Jiangsu,P.R.China
131220111@smail.nju.edu.cn

Xinbei Liu, Shenghiu Gu
Software Institute
Nanjing University
Nanjing,Jiangsu,P.R.China

***Abstract*—*Background:* Logging practice is a critical activity in software development, which aims to offer significant information to understand the runtime behavior of software systems and support better software maintenance. There have been many relevant studies dedicated to logging practice in software engineering recently, yet it lacks a systematic understanding to the adoption state of logging practice in industry and research progress in academia. *Objective:* This study aims to synthesize relevant studies on the logging practice and portray a big picture of logging practice in software engineering so as to understand current adoption status and identify research opportunities. *Method:* We carried out a systematic review on the relevant studies on logging practice in software engineering. *Results:* Our study identified 41 primary studies relevant to logging practice. Typical findings are: (1) Logging practice attracts broad interests among researchers in many concrete research areas. (2) Logging practice occurred in many development types, among which the development of fault tolerance systems is the most adopted type. (3) Many challenges exist in current logging practice in software engineering, e.g., tradeoff between logging overhead and analysis cost, where and what to log, balance between enough logging and system performance, etc. *Conclusion:* Results show that logging practice plays a vital role in various applications for diverse purposes. However, there are many challenges and problems to be solved. Therefore, various novel techniques are necessary to guide developers conducting logging practice and improve the performance and efficiency of logging practice.**

***Keywords*—Logging Practice, Systematic Literature Review, Software Engineering**

I. Introduction

As a technique to collect the runtime information of system behavior, logging practice is more and more important in modern software development and maintenance. By inserting logging statements in the source code, key information about the behavior of software systems could be captured and recorded in logs. These logs play a vital role in debugging, failure handling and system recovery, system analysis, and so on [1], which help a lot in the operation and maintenance of software systems [2].

Logs are very powerful and efficient for locating and resolving the problems when the software systems are becoming more and more complicated. Since logging practice is the starting point to generate logs, many researchers are dedicating to develop unified logging specifications or tools to guide other developers carrying out logging or even support automatic logging. Besides, studies focusing on better methods of logging practice are also abundant. Nevertheless, we still lack necessary understanding towards the adoption status and research progress of logging practice in software engineering. To this end, we carried out a systematic review on the logging practice in software engineering by following the guideline proposed by Kitchenham etc. [3] so as to establish a holistic view.

As a result, we identified 41 relevant primary studies on this topic. Based on the evidence we collected, we systematically illustrate the occasions that logging practice is needed, the challenges, the current state of the art and the future research directions of logging practice in software engineering.

The rest of this paper is organized as follows: Section II describes the background and related work of logging practice in software engineering. Section III illustrates the steps of our research and the results and findings are discussed in Section IV. We discuss the threats to validity at this stage in Section V and finally conclude this paper in Section VI.

II. Background & Related Work

There have been several frameworks providing supports for logging, such as Apache Logging Services, Windows Event Log, IBM Common Event Infrastructure. These frameworks offer APIs for creating, collecting and parsing log entries, but leaving the implement of logging (i.e. to insert logging statements in source code) for developers.

Logging is a significant activity in many fields, such as software debugging [4], failure recovery [5], system analysis [6], etc. Therefore, a large number of studies pay close attention to specifying how to log appropriately or how to improve logging [7] [8] [9] [10].

By logging, we can collect a great deal of key information on the runtime behavior of software systems for postmortem analysis. For example, Oliner et al. [11] presented an overview of logging analysis, including the application areas of log analysis and the methods to analyze logs. Security related events can be extracted from logs for analysis too, which are very valuable in terms of system security [6].

978-1-5386-3681-7/17 $31.00 © 2017 IEEE
DOI 10.1109/APSEC.2017.61

Message logging and secure logging are two major types of logging which are used for specific purposes. Message logging [12] is a significant activity in fault-tolerant systems, which is able to help system recover from process crash. Each process periodically logs the messages it received. When a process failure occurs, it can replay the messages in the logs in the sequence of the process received, and thus restore to its original state before the failure. However, there are still some issues [13] [14] regarding the performance of message logging protocols, such as the size of messages that should be logged, the time for recovering, the failure-free performance.

Since logs potentially store a lot of information of software systems, some of the information may be very sensitive, therefore it is important to keep the logging system from intrusion to avoid fateful consequences. Secure logging [15] [16] is a mechanism intended to guarantee log data correct for future use.

III. Research Design

According to the guidelines proposed by Kitchenham et al. [3], we carried out our systematic review on logging practice.

A. Research Questions

To address the objective of identifying the adoption status and the research progress of logging practice, we designed five research questions as follows.

RQ1: *What kinds of software development require logging practice?*

RQ1 aims to identify the application context of logging practice so as to guide practitioners if them involve in software development with similar context. For example, high performance computing systems require logging to ensure performance in particular. Besides, we also noticed that software development involving multi-core, distributed systems, etc. also requires logging practices. By collecting the types of software development regarding logging practices, we intent to identify situations that logging practice is potentially needed.

RQ2: *How do developers conduct logging practice at present?*

RQ2 is mainly answered in two aspects, how developers utilize log tools as well as where and what they code in log statements. Findings derived in this research question might provide useful guidance for practitioners to carry out logging practice.

RQ3: *What issues and challenges need to be addressed in logging practice?*

RQ3 intends to recognize the issues and challenges regarding the practical adoption of logging practice in industry. For instance, fully-automatic tools need to be designed to decide the location and content of log statement with high accuracy for the sake of reducing developers' workload. Unfortunately, it is not easy at present. By summarizing recognized issues and challenges, developers may make better decisions when logging practice is needed in their development. Meanwhile, researchers may also find clues to carry out relevant studies.

RQ4: *What is the state-of-the-art of the research on logging practice?*

RQ4 attempts to portray a landscape on the researches of logging practice up till now, including the research topic, their findings, their future work, etc. RQ5 shows researchers the way that which directions are worth further investigation.

B. Research scope

To constrain the scope of this study, two noteworthy points should be elaborated here. First of all, although research on log analysis constitutes a large portion of log-related studies and there are numerous studies on this hot topic, we did not include this topic as our research concern. Log analysis aims at utilizing existing log files to detect bugs, optimize performance, etc. However, our study focuses on developer's perspective, i.e. to investigate logging practice which is the development practice that software developers insert statements into source code to collect the runtime information for log analysis in particular. As for the other, our SLR excluded the topics related to user behavior logging, e.g., logging the expression of users via camera or logging the users' browsing trials on some websites (e.g., e-commerce sites). While these types of logging usually works for business purpose, we only intend to focus on software engineering.

C. Search Process

Seven researchers are involved in this study. To guarantee work quality during publication screening and data extraction, at least two researchers took care of one paper at the same time. Besides, one experienced researcher cross-checked the work randomly. In addition, regular meetings were held among the researchers in order to deal with the issues and divergence occurred during the study.

1) Manual search: The manual search was directed towards the top conferences and journals in software engineering including: International Conference on Software Engineering (ICSE), IEEE Transactions on Software Engineering (TSE), Empirical Software Engineering (EMSE), IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE Working Conference on Mining Software Repositories (MSR) and Journal of Systems and Software (JSS)

The publication time span was constrained from 2000 to 2016 for the reason that the number of papers related to log seems to be quite small before 2000 based on our basic concept of study population derived from preliminary automatic search. Finally, we got 42 papers on log in total and then obtained 13 papers focusing on logging practice after screening. These resulted papers portrayed a landscape for us to make a better understanding of the research status of logging practice and laid the foundation for the subsequent stages.

2) Proposing search string: After several rounds of tuning, the resulting search string is then given below:

((logging OR log) AND (practice OR tool OR technique) AND (development OR code OR programming)) in title or keyword or abstract

TABLE I
SELECTION CRITERIA

Inclusion Criteria	
I1.	Publications that investigate the methodology for logging practice.
I2.	Publications that investigate the tools/frameworks/systems which support logging practice.
I3.	Publications that propose a standard for logging practice.
I4.	Publications that are peer-reviewed (conference paper, journal article).
I5.	Publications that are primary studies on logging practice.
Exclusion Criteria	
E1.	Publications that investigate log analysis.
E2.	Publications that investigate the usage of logs.
E3.	Publications that investigate the technologies on logging user behaviors.
E4.	Publications that are not written in English.
E5.	Additionally, short papers, demo or industry publications are excluded.

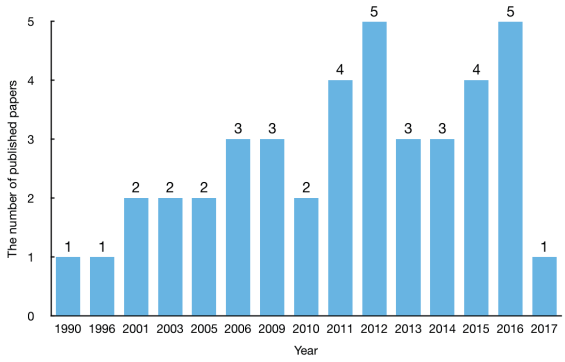

Fig. 1. The distribution of the papers' published years

Note that each word in this search string needs to transformed into different term variants so as to be applied in diverse search engines , e.g., *practices* is one of the variant of *practice*.

3) Automatic search: The automatic search was performed in five popular electronic libraries, i.e. ACM Digital Library, IEEE Xplore, ScienceDirect, Wiley Online Library, and Springer Link, according to suggestions derived in most existing SLR studies [17]. The automatic search results in 19641 articles, which to a certain degree reflects the attention researchers paid to this research topic.

D. Study Selection

Basically, the study selection process is divided into two steps, i.e. the *Quickly Scanning* and *Entirely Reading* respectively.

1) Quickly Scanning: During this step, screening of titles and abstracts for the potential studies was performed according to the inclusion & exclusion criteria detailed in Table I. As for those papers unable to be decided just on titles and abstracts, we postponed the decision to the following steps. As the result, we included 71 publications at this stage.

2) Entirely Reading: Given the initially identified publications, we unfolded an entirely reading for the sake of deciding the uncertain ones and eliminating the irrelevant ones. As a result, a total of 41 publications were identified for final analysis.

E. Data Extraction

In order to collect evidence to answer the research questions, a data extraction schema was applied to collect relevant data from the studies we identified previously, as listed in Table II.

IV. RESULTS

A. Overall Status

Our SLR finally identified 41 papers[1] related to the logging practice for the final analysis. The 41 papers are composed of 26 conference papers and 15 journal articles. Fig. 1 presents the distribution of the published years of the papers. The majority of the papers were published after 2001, but there are two outliers published in 1990 and 1996 respectively. It can be seen that the logging practice has always been a research focus in software engineering in recent years. One noteworthy fact is that since we carried out this study early this year, there is only one study in 2017.

[1]http://tinyurl.com/y7es7hpd

B. Types of Software Development Needing Logging Practice (RQ1)

Given the collected evidence, it is obvious that logging practice has been involved in several main types of software development. As shown in Fig. 2, logging practice has been generally applied in systems where *fault tolerance* is one of the most vital requirements ([S22], [S31], [S24], [S6], [S26], [S9], [S32], [S4], [S8], [S5], [S13]). In *fault tolerance* systems, logging usually plays a vital role to support rollback recovery. Systems can recovery from a failure using necessary messages recored in logs. Besides, it intrigued us that six papers among the eleven which mention fault tolerance could also be categorized as *high performance computing* system ([S22], [S31], [S24], [S6], [S32], [S13]). Developers apply logging practice to ensure high performance and reliability, so as to reduce the workload of maintenance.

Right following *fault tolerance* systems, logging for *secure* purpose is another major software development type ([S34], [S23], [S33], [S3], [S2], [S37], [S1]). During the data extraction process, we noticed that secure logging usually appeared along with audit log as a rule except one paper ([S1]). Audit logs ordinarily include some sensitive information which may be modified illegally by attackers.

In addition, *open source* projects ([S7], [S39], [S17], [S21], [S20]) and some *complex* systems ([S10], [S40]) also utilize logging for preferable bug tracking and maintenance. As for *distributed* systems ([S16], [S29], [S9], [S32], [S18]), logging is used to deal with the predicament in distributed circumstance particularly, e.g., obtaining a correct behavior sequence. Furthermore, in *concurrent* environment ([S19], [S35]) and *multicore* systems ([S19], [S15]), logging acts a pivotal part in making it possible to trace the execution.

TABLE II
THE DATA EXTRACTION SCHEMA

Attribute	Description	To answer RQ
Author	The author(s) of the publication.	Metadata
Title	The title of the publication.	
Year	The published year of the publication.	
Venue	The publisher of the publication.	
Characteristics related to logging practice	Macroscopically, the context of projects which conduct logging practice, e.g., type of project, scale of project, etc.; Microscopically, the distinctions between code with logging statements and those without.	RQ1
Research objective	Logging practices applied in industry, mainly in two perspectives. Manual coding: focusing on the strategies or specifications of logging in code; Automated tools: focusing on the improvement or optimization of code via the log statements generated by automated tools.	RQ2
Issues and challenges	Issues and challenges mentioned in the paper when it comes to conducting logging practice.	RQ3
Research question	The research question(s) of the selected paper.	RQ4
Research method	The method adopted in the selected paper.	
Conclusion	The conclusion(s) of the selected paper.	
Future work	The direction of the investigation of logging practice mentioned in the paper.	

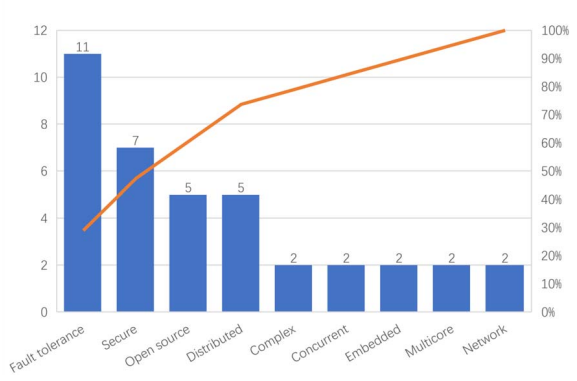

Fig. 2. Development context mentioned in papers

It should be pointed out that the types of software development may overlap here. For example, an *open source* project might also be a *secure* system. On one hand, papers we retrieved in this study normally did not contain detailed information on the systems. One the other hand, however, the development types we discussed above have presented useful information already, which fits our original research objective at this stage.

C. How To Do Logging (RQ2)

Research question 2 investigated the techniques used for logging practices. The collected evidence shows various types of logging practices and techniques used in practice.

One common focus is message logging, which is investigated in 9 papers ([S22], [S31], [S24], [S6], [S26], [S9], [S32], [S5] and [S13]). Message logging is composed of a family of algorithms and used as an important rollback recovery technique, which contributes to fault tolerance in practice. Most message logging are designed and implemented at library level, such as MPI (Message Passing Interface) library, and used to alternate Checkpoint/Restart as a common approach for fault tolerance. Protocol is unquestionably the main concern in message logging. Existing optimistic message logging protocols saves dependency information at regular intervals and replays them when needed. So overhead and memory pressure might be relatively high, and scalability might be limited as well.

Another common focus is secure logging, which is investigated in 7 papers ([S34], [S23], [S33], [S3], [S2], [S37] and [S1]). Logging practices focusing on secure logging usually compose logging systems, which aims at improving the robustness on their own and serves for system accountability, which is why the integrity of the log entries is, beyond all doubt, a cardinal requirement for secure logging systems. In order to improve the logging systems themselves, they are built above authentic standards, procedures, protocols or techniques, such as *Syslog* [S34], *tamper-proof logging protocols* [S33] and *cryptographic log protection* [S23], [S37].

Industrial projects are another type of main concern of logging practices. General approaches to instrument projects with log messages include ad-hoc logging, general-purpose logging libraries and specialized logging libraries [S7]. To be specific, logging in industry relies on various models and frameworks, such as *Apache log4cxx* [S36], [S7], [S39], [S28], [S41], [S21], [S20], *Apache Commons Logging* [S20], *SLF4J* [S20], *Logback* [S36], *POCO C++ Libraries* [S36], *LogicPoet* [S36], *syslog* [S39] and *cyclic debugging* [S16]. Meanwhile, there are several log levels like TRACE, DEBUG, INFO, WRAN, ERROR, and FATAL, which can be set by developers to deal with different development tasks within various contexts[S7], [S39], [S40], [S17], [S21], [S14], [S20].

D. Challenges of Logging Practice (RQ3)

It can be observed that many studies ([S10], [S38], [S39] and [S28]) point out that, in practice, there are frequently subjective and arbitrary logging practices because the implementation of logging mostly depends on the knowledge and expertise of the developers even with logging libraries. In other words, logging practices lack formal specifications and systematic design [S28], [S41], [S35]. As a result, the

logs can be incompleteness and inaccurate. Owing to the bad performance of logging practices, more effort would be spent on adjusting logging statements [S20]. Therefore, it is important for software developers to know where to log and what to log. However, as studies [S38] [S39] implied, failures are hard to predict so that it is hard to determine where to log.

Logging is a helpful activity, but how to log appropriately and effectively is a critical problem. Logging frameworks are required to be flexible for developers to dynamically insert or delete logging statements, as well as be specific to obtain structured and high-qualified logs [S36], [S12]. In reality, it is very likely for developers to log either too little or too much in software development [S3], [S41], [S21], [S14], [S20]. Logging too little may result in insufficient runtime information for postmortem analysis to diagnose and understand the behavior of software systems, whereas logging too much not only takes developers' time to write and maintain the statements but also causes runtime overhead. What's worse, excessive logs make it much difficult to discover the useful information for postmortem analysis.

It is an important issue of logging practice that how to make the tradeoff between logging overhead and analysis cost [S4], [S11]. Logging provides necessary runtime information for postmortem analysis. Therefore, adequate logging is important for understanding the problems comprehensively, but it is also equally important to reduce the cost for log analysis. In this sense, accurate logging that generate proper size of logs which contain enough information for analysis is an important research topic.

E. State of Art of Logging Practice (RQ4)

In general, we can categorize the 41 studies into two types. The first type mainly focuses on providing guidance for developers to conduct logging practice. The second type focuses on improvement to current practice and method to perform logging practices. To be specific, both types also contain relevant studies to devise automatic tools.

a) Guide Logging Practice: Several studies focus on guiding software developers to perform logging practice during coding. For example, the study conducted by Fu et al. [S14] systematically investigated where to log from industrial systems. Yuan et al. [S39] and Chen et al. [S7] conducted two quantitative characteristic studies that focused on the log modification and Pecchia et al. [S28] studied event logging practices by assessing an industrial software development process. These studies analyzed the logging practice or logs in industrial software development to form useful suggestions, which could be taken as guidelines for logging practice. In study [S21], the reasons of log changes are investigated and then just-in-time suggestions for log changes during coding can be automatically provided. Besides of this study, there are several studies [S41][S30][S20][S17] that developed automatic tools to support logging practice.There is another important concern in logging practice which is the tradeoff between logging and analysis [S11][S18][S25]. These studies investigated how to log with minimal overhead and meanwhile guarantee enough data for efficient postmortem analysis. Another research concern belonging to this type of study is the storage of logs. For example, several studies [S2][S1][S33] take an insight into the secure storage of the logs.

b) Improve Logging Practice: Another category of the relevant studies is about how to improve the logging practice in terms of functionality, efficiency and scalability ([S36][S4][S27][S15][S29]). Besides, there are also several studies focusing on enhancing logging practice so as to improve the capability for system dependability evaluation [S10], replay [S27], failure diagnosis [S40][S38], etc. Message logging is designed for recovery from failure in fault-tolerance systems([S26][S6][S13][S24][S5][S31][S22][S9]). The size of message logs and the scalability are two big issues. For example, the study [S26] presented a technique to reduce the messages logged by uncoordinated checkpointing algorithms, while [S6] presented a hybrid approach that combines coordinated and non coordinated checkpointing to reduce the overhead, and meanwhile, the studies [S5][S31] aimed at good scalability. There are five studies that presented their designed tools to improve the logging. The studies [S16] and [S35] presented their tools to verify system behaviors by analyzing the date logged at runtime, while [S19] and [S8] described their logging and replaying tools with as less as possible cost. The study [S12] illustrated a dynamic logging facility for networked embedded systems to achieve better flexibility, efficiency, and high synchronization accuracy.

c) Research Direction in Future: Researchers already involved in practice logging usually possess a better position to recognize important topics which need further investigation. With this consideration, we summarized the future research work identified by these researchers briefly to provide clues for future research direction. Generally, we can classify these future work into two major categories, i.e. the specific improvement and the general directions for future research. For studies in the first category, several authors discussed future work to improve their current work with specific purpose. For example, the study [S21] investigated the reasons of log statement changes in source code so as to summarize and provide suggestions to developers to make log changes. Researchers suggested that feedback collected from actual developers should be considered to achieve better understanding and improvement of the method. In study [S11], the authors identified the branches that need to be logged to reduce the total number of paths for trace analysis, and therefore to address the balance between instrumentation overhead and debugging time. The extension of this approach for multi-thread applications forms the topic for future work. Tasiran et al. [S35] illustrated how to verify system behaviors by analyzing runtime information and the future work should be that applying their techniques to industrial software. For future research directions with general purpose, we identified six research directions in 5 studies. 1) The first future work should be more powerful logging tools to support logging automatically[S14][S10], which can not only reduce developers' efforts to write logging statements but also standardize the logs so as

to facilitate log analysis. 2) Besides, logging on demand should be investigated to dynamically generate the necessary logs so as to reduce redundant logs [S14]. 3) In addition, end-to-end logging is another direction to help picture the behaviors of the system [S14]. 4) Moreover, it can be significantly useful to give logs a tag when generation and thus logs can be categorized for better postmortem analysis[S14]. 5) According to [S41], current logging is all static and what to log is determined before runtime. Therefore runtime logging can be further investigated for efficient logging practice. 6) Lastly, balance between logging and analysis [S11][S18] is always a significant focus for further study.

V. Threats to Validity

We intend to provide a comprehensive and systematic review on logging practice in software engineering. However, there are some threats to validity that need further discussion.

First, although we tuned the search string in this study several rounds, it is still difficult to achieve good balance between the huge number of articles and the coverage of articles retrieved in manual search. Besides, various search engines treat search string in different ways, leading slight adjustment to the search string for specific search engines. As a consequence, it is possible that several relevant papers may have been neglected.

Second, the evidence in the papers may be insufficient to comprehensively picture the whole adoption of logging practice in industry. We noticed that there are many materials on various technical websites, providing suggestions to perform logging practices in software projects. To guarantee the quality of studies, we did not include these evidence at this stage.

VI. Conclusion

Logging practice plays a vital role in modern software development and maintenance. This paper presents a systematic review on existing studies with the topic of logging practice in software engineering. Based on 41 identified papers, we gain an insight of the industry adoption and the research progress of logging practice in software engineering. The contribution of our work can be highlighted as follows:

First, we carried out a systematic review with the topic of logging practice in software engineering to portray a big picture of the adoption status and research progress. To the best of our knowledge, this might be the first systematic review on logging practice.

Second, our investigation implies that although the importance of logging practice has been recognized by many practitioners and researchers, there is still a lack of suitable guidance to help developers conduct efficient logging practice. As the result, most developers carry out logging practice mainly depending on personal experience.

Third, a balance (or tradeoff) between extensive logs and lost-cost analysis is a critical challenge regarding logging practice. Moreover, this issue may also impact system performance due to the fact that most logging statements require extra computer resources.

Acknowledgment

The authors would like to thank Dexian Yu, Hui Wang and Xin Kang for their help on determine the research questions and improving the protocol. We also thank them for collaboration on screening and identifying relevant papers.

References

[1] A. Pecchia, M. Cinque, G. Carrozza, and D. Cotroneo, "Industry practices and event logging: Assessment of a critical software development process," in *Proceedings of the 37th IEEE International Conference on Software Engineering (ICSE '15)*. IEEE, 16 May 2015, pp. 169–178.

[2] M. D. Syer, Z. M. Jiang, M. Nagappan, A. E. Hassan, M. Nasser, and P. Flora, "Leveraging performance counters and execution logs to diagnose memory-related performance issues," in *Proceedings of the 29th IEEE International Conference on Software Maintenance (ICSM '13)*. IEEE, 22 September 2013, pp. 110–119.

[3] B. A. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," *Technical Report EBSE 2007-001. Keele University and Durham University Joint Report*, 9 July 2007.

[4] D. Yuan, H. Mai, W. Xiong, L. Tan, Y. Zhou, and S. Pasupathy, "SherLog: Error diagnosis by connecting clues from run-time logs," *ACM SIGARCH Computer Architecture News*, vol. 38, no. 1, pp. 143–154, 13 March 2010.

[5] E. Meneses and L. V. Kalé, "Camel: Collective-aware message logging," *The Journal of Supercomputing*, vol. 71, no. 7, pp. 2516–2538, 1 July 2015.

[6] J. P. Leite, "Analysis of log files as a security aid," in *Proceedings of the 6th Iberian Conference on Information Systems and Technologies (CISTI '11)*. IEEE, 15 June 2011, pp. 1–6.

[7] Q. Fu, J. Zhu, W. Hu, J.-G. Lou, R. Ding, Q. Lin, D. Zhang, and T. Xie, "Where do developers log? an empirical study on logging practices in industry," in *Companion Proceedings of the 36th International Conference on Software Engineering (ICSE Companion '14)*. New York, NY, USA: ACM, 7 June 2014, pp. 24–33.

[8] G. Lee, J. Lin, C. Liu, A. Lorek, and D. Ryaboy, "The unified logging infrastructure for data analytics at Twitter," *Proceedings of the Vldb Endowment*, vol. 5, no. 12, pp. 1771–1780, 1 August 2012.

[9] D. Yuan, J. Zheng, S. Park, Y. Zhou, and S. Savage, "Improving software diagnosability via log enhancement," *ACM Transactions on Computer Systems (TOCS)*, vol. 30, no. 1, pp. 4:1–4:28, 1 February 2012.

[10] S. Lal, N. Sardana, and A. Sureka, "LogOptPlus: Learning to optimize logging in catch and if programming constructs," in *Proceedings of the 40th Annual Computer Software and Applications Conference (COMPSAC '16)*. IEEE Computer Society, 10 June 2016, pp. 215–220.

[11] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Communications of the ACM*, vol. 55, no. 2, pp. 55–61, February 2012.

[12] L. Alvisi and K. Marzullo, "Message logging: Pessimistic, optimistic, causal, and optimal," *IEEE Transactions on Software Engineering*, vol. 24, no. 2, pp. 149–159, February 1998.

[13] A. P. Sistla and J. L. Welch, "Efficient distributed recovery using message logging," in *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*. ACM, 1989, pp. 223–238.

[14] E. N. Elnozahy and W. Zwaenepoel, "On the use and implementation of message logging," in *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on*. IEEE, 1994, pp. 298–307.

[15] R. Accorsi, "Towards a secure logging mechanism for dynamic systems," *Proceedings of It Security Symposium*, 2005.

[16] S. Sackmann, J. Strker, and R. Accorsi, "Personalization in privacy-aware highly dynamic systems," *Communications of the Acm*, vol. 49, no. 9, pp. 32–38, 2006.

[17] H. Zhang, M. A. Babar, and P. Tell, "Identifying relevant studies in software engineering," *Information and Software Technology*, vol. 53, no. 6, pp. 625–637, 30 June 2011.