How Do Developers' Profiles and Experiences Influence their Logging Practices? An Empirical Study of Industrial Practitioners

Guoping Rong†, Shenghui Gu*, Haifeng Shen‡, He Zhang†, Hongyu Kuang†
*†*the State Key Laboratory for Novel Software Technology, Nanjing University*, Nanjing, China.
‡*the HilstLab, Peter Faber Business School, Australian Catholic University*, Sydney, Australia.
*shenghui.gu@smail.nju.edu.cn †{ronggp, hezhang, khy}@nju.edu.cn ‡haifeng.shen@acu.edu.au

2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) | 978-1-6654-5701-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSE48619.2023.00080

***Abstract*—Logs record the behavioral data of running programs and are typically generated by executing log statements. Software developers generally carry out logging practices with clear intentions and associated concerns (*I&Cs*). However, *I&Cs* may not be properly fulfilled in source code as log placement — specifically determination of a log statement's context and content — is often susceptible to an individual's profile and experience. Some industrial studies have been conducted to discern developers' main logging *I&Cs* and the way *I&Cs* are fulfilled. However, the findings are only based on the developers from a single company in each individual study and hence have limited generalizability. More importantly, there lacks a comprehensive and deep understanding of the relationships between developers' profiles and experiences and their logging practices from a wider perspective. To fill this significant gap, we conducted an empirical study using mixed methods comprising questionnaire surveys, semi-structured interviews, and code analyses with practitioners from a wide range of companies across a variety of industrial domains. Results reveal that while developers share common logging *I&Cs* and conduct logging practices mainly in the coding stage, their profiles and experiences profoundly influence their logging *I&Cs* and the way the *I&Cs* are fulfilled. These findings pave the way to facilitate the acceptance of important logging *I&Cs* and the adoption of good logging practices by developers.**

***Index Terms*—Logging practice, Intention, Concern, Fulfill**

I. Introduction

As a common type of runtime data, logs record the dynamic behavior of software systems and thus play a vital role in the daily tasks in software development and operations [1]. Logging is widely used in modern software development and maintenance [2] to obtain logs that facilitate quality management practices such as debugging [3]–[5], performance bottleneck analysis [6], [7], and test analysis [8], [9]. In some scenarios, information in logs is believed to be the only available data source in production environments for failure diagnosis [10]–[13]. Apparently, high-quality logs require well-established logging practices, in which a considerable number of challenges need to be tackled, as revealed in several studies [14]–[16].

In fact, logging practices have received long-term attention from both software engineering practitioners and researchers, resulting in many advanced logging approaches, frameworks, and tools [14]. However, the current state-of-the-practice is far from satisfactory [17] as the mainstream logging practices still require developers to manually place log statements into source code by determining its context (*where to log*) and content (*what to log*). The so-called *2-W* questions are often susceptible to an individual's profile (e.g., role) and experience (e.g., whether to follow logging guidelines). As a result, the effectiveness of logging practices and the quality of the generated logs vary widely [13]. To address this issue, some researchers pointed out that logging practices should accumulate and learn from the *best practices* [18], [19]. This raises the necessity to understand how developers think, design, and enact log instrumentation. However, only a few empirical studies were conducted to understand developers' logging practices. In particular, each study only surveyed developers from a single company, thus limiting the generalizability of its findings. More importantly, no work has been done to establish the relationship between developers' profiles and experiences and their logging practices.

In practice, developers carry out logging practices with clear *Intentions*, e.g., to record the values of important variables, and associated *Concerns* as there are side effects in executing log statements [23]–[26], e.g., performance overhead, as too many log statements may lead to performance decline [18] and incautious log statements may also bring in security risks [27]. In log placement, developers need to consider both *Intentions* and *Concerns* (*I&Cs*) to balance between the benefits and costs of carrying out logging practices. Therefore, it is important to understand developers' *I&Cs* and the way they are fulfilled through log placement as without such an understanding, it may be difficult, if not entirely impossible, to comprehend, learn and adopt good logging practices [10].

In the past few years, only a few empirical studies were conducted to discern the main logging *I&Cs* and their fulfillment, which are listed in Table I. For example, developers' coding behaviors may be directly influenced by specific coding conventions for logging practices in different organizations. However, these findings were only based on the developers from a single company in each individual study, which are hence subject to external validity limitations. Furthermore, synthesizing conclusions across these studies does not lead to new findings. More importantly, as developers are different from one another in terms of their profiles and their experiences of conducting logging practices, one should adopt the logging practices suggested by another only if they share similar profiles

1558-1225/23/$31.00 ©2023 IEEE
DOI 10.1109/ICSE48619.2023.00080

TABLE I
EMPIRICAL STUDIES ON LOGGING PRACTICES IN INDUSTRY.

Empirical Study	Origin of Participants	No. of Projects	Research Method	Research Questions	Research Findings
Fu et al. [20]	1	2	• Code analysis • Questionnaire survey	1) Categories of logged code snippets 2) Factors developers consider to determine whether to log or not 3) Automatic determination of where to log	1) There are five categories of logged snippets, including assertion-check logging, return-value-check logging, exception logging, logic-branch logging, and observing-point logging. 2) Main factors considered for logging include exception type, functions, and variables, while reasons of not logging include passing the logging decision to subsequent operations, recoverable exceptions, and non-critical exceptions. 3) Predicting where to log is feasible, but it requires further exploration.
Pecchia et al. [21]	1	1	• Code analysis • Log inspection • Interview	1) Developers' logging intentions 2) Procedures developers used to fulfill logging intentions 3) Impact of industry practices on logging	1) There are three major logging intentions: state dump, execution tracing and event reporting. 2) While logging is strongly developer-dependent, fulfillment of logging in source code shares common patterns despite the adoption of different logging procedures and the lack of common rules. 3) Industry practices impact logging, for example, lack of company-wide logging regulations leads to a variety of log collection mechanisms and formats in the same software product.
Rong et al. [22]	1	3	• Interview • Code analysis	1) Developers' awareness and understanding of logging practices 2) Developers' *I&Cs* when conducting logging practices 3) Extent of *I&Cs*' fulfillment in source code 4) Potential improvement of logging practices	1) Logging practices are commonly adopted in software development by the company. 2) Common intentions include error debugging, system behavior understanding, and performance diagnosis, while common concerns involve I/O, memory, CPU, and storage overheads. 3) A considerable proportion of developers' *I&Cs* are not fulfilled in the source code due to negligence and code modification. 4) Potential improvement opportunities include enhanced supporting tools and logging guidelines.
This work	42	12	• Questionnaire survey • Interview • Code analysis	1) Influence of developers' profiles on their logging *I&Cs* 2) Influence of developers' experiences on the way their logging *I&Cs* are fulfilled	1) Developers' *I&Cs* may change with time, and novice developers have significantly less logging intentions and are much less concerned about the side effects, but they improve very quickly (within one year) and after that grow slowly with time. 2) Developers who consider log placement in early software development stages and who adopt logging guidelines better fulfill their logging *I&Cs*.

and experiences. However, these studies did not attempt to understand how developers' logging practices were influenced by their profiles and experiences.

Therefore, to facilitate the adoption of these findings, it is crucial to understand the relationships between developers' profiles and experiences and their logging practices from the perspectives of *I&Cs* and the way *I&Cs* are fulfilled. This study aims to fill this significant gap through an empirical study using mixed methods [28] comprising questionnaire surveys, semi-structured interviews, and code analyses involving practitioners from a wide range of companies across a variety of industrial domains. It addresses two research questions. One is about how developers' profiles influence their logging *I&Cs*. The other is about how developers' experiences influence the way their logging *I&Cs* are fulfilled. We have made two key findings. One is that developers' *I&Cs* may change with time, and novice developers have significantly less logging intentions and are much less concerned about the side effects, but they improve very quickly (within one year) and after that grow slowly with time. The other is that developers who consider log placement in early software development stages and who adopt logging guidelines better fulfill their logging *I&Cs*.

The main contributions of this study are as follows:

- We conducted a mixed-method empirical study to collect evidence with respondents from a wide range of companies across a variety of industrial domains.
- We triangulated multiple sources of evidence to gain insight into how developers' profiles and experiences influence their logging practice from the perspectives of their logging *I&Cs* and the way *I&Cs* are fulfilled.
- We recommended good logging practices to practitioners and industry organizations. For example, training and adoption of logging guidelines are both effective ways to help novice developer to perform logging practices, while considering *I&Cs* in early development stages other than the coding stage may better fulfill logging *I&Cs*.

The rest of the paper is organized as follows. Section II introduces some related work. Section III describes the research method and process, followed by the results and findings in Section IV. Sections V and VI discuss the findings and the validity risks pertinent to the study, respectively. Section VII concludes the paper with a summary of major findings.

II. RELATED WORK

In this section, we introduce empirical studies that were conducted to understand logging practices in industry. Due to information security, profitability requirements, and the likes, it is much harder to obtain source code and log statements in industrial projects than in open source projects. As a result, the majority of studies investigate open source projects, while only a few studies investigate logging practices in industrial projects. Nevertheless, compared to the scattered developers in open source projects, developers in industrial projects are more likely to be co-located and as such it is possible to understand

their logging practices through ethnographic methods such as interview. The existing empirical studies listed in Table I all center around logging *I&Cs* and their fulfillment in source code through log placement [20]–[22]. However, all of them only surveyed developers from a single company and engaged in the code analysis of 1–3 software development projects.

In particular, Fu et al. [20] explored logging decisions on *where to log* through code analysis of two projects and a questionnaire survey at Microsoft. They identified five categories of logged snippets and discovered that the main factors considered for logging are exception type, functions, and variables. They also discussed the reasons of not logging including passing the logging decision to subsequent operations, recoverable exceptions, and non-critical exceptions. They further proved the feasibility of predicting where to log.

Pecchia et al. [21] analyzed the code and inspected the log entries of an industrial software platform in the transportation domain. They interviewed several developers from different teams involved in the project at Selex ES who were pursuing different log analysis goals in order to understand developers' common logging intentions, the procedures developers used to fulfill the intentions, and the impact of the company's practices on logging. They identified the three common logging intentions of state dump, execution tracing and event reporting. They further discovered that while logging is strongly developer-dependent, fulfillment of logging in source code shares common patterns despite the adoption of different logging procedures and the lack of common rules. They also confirmed that industry practices impact logging, for example, lack of company-wide logging regulations leads to a variety of log collection mechanisms and formats in the same software product.

Rong et al. [22] interviewed developers from a leading IT company to understand their logging intentions and the associated concerns in conducting logging practices as well as the fulfillment of logging in source code and analyzed the code of three software projects to triangulate their claimed logging practices. They identified the common logging intentions of error debugging, system behavior understanding, and performance diagnosis and the common concerns involving I/O, memory, CPU, and storage overheads. They further revealed that a considerable proportion of *I&Cs* were not fulfilled in the source code due to negligence and code modification.

These studies have painted a clear picture of common logging *I&Cs* and their practical fulfillment in source code. Our work distinguishes itself from the above studies in that it is conducted to establish a comprehensive and deep understanding of the relationships between developers' profiles and experiences and their logging practices from the perspectives of *I&Cs* and the way *I&Cs* are fulfilled with evidence from 42 companies across different industrial domains.

III. Research method

An overview of the research method and process is shown in Fig. 1. In this section, we elucidate our study method, including research questions, study design, and study execution.

A. Research questions

To guide our research, we first describe the goal of this study using a Goal-Question-Metric (GQM) [29] style as follows:

To investigate and analyze *current logging practices*

For the purpose of *understanding the relationships between developers' profiles and experiences and their logging practices from the perspectives of I&Cs and the way I&Cs are fulfilled*

In the context of *real-world software projects in a wide range of companies across a variety of industrial domains.*

To achieve the research goal, we define the following research questions:

RQ1: How do developers' profiles influence their logging *I&Cs*?

RQ1 aims to investigate how developers' *I&Cs* that drive their logging practices are influenced by their profiles characterized by their roles and years of working. The findings from RQ1 will establish a clear relationship between developers' profiles and their logging *I&Cs*.

RQ2: How do developers' experiences influence the way their logging *I&Cs* are fulfilled?

RQ2 aims to shed some light on the extent developers' *I&Cs* are fulfilled in source code through log placement and further investigate how the way their *I&Cs* are fulfilled is influenced by their own experiences, especially in terms of the timing of conducting logging practices and the tendency to following logging guidelines/coding conventions. Therefore, we divide RQ2 into three sub-questions:

(1) RQ2.1: How much are logging *I&Cs* satisfied in log statements?

(2) RQ2.2: How do development stages in which logging *I&Cs* are fulfilled influence the degree of satisfaction?

(3) RQ2.3: What role do logging guidelines/coding conventions play in the fulfillment of logging *I&Cs*?

It should be noted that as logging guidelines are very much likely contained in coding conventions (e.g., [30]), we only investigate the influence of guidelines on the fulfillment of logging *I&Cs* involving coding-related activities. It is also noteworthy that a developer's experience is likely pertinent to their profile, which may indirectly influence the way their logging *I&Cs* are fulfilled. The findings from RQ2 will establish a relationship between developers' profiles, experiences and the way their *I&Cs* are fulfilled.

B. Study design

To collect necessary evidence and ensure its validity, we applied a mixed-method study that combined a structured questionnaire survey and a series of semi-structured interviews supplemented by code analysis. Different sources of evidence played different roles in the analysis. Specifically, all the statistical results were only derived from the answers to the questionnaire. The interview and the code analysis provided supplementary evidence to confirm or contradict the observations from the questionnaire.

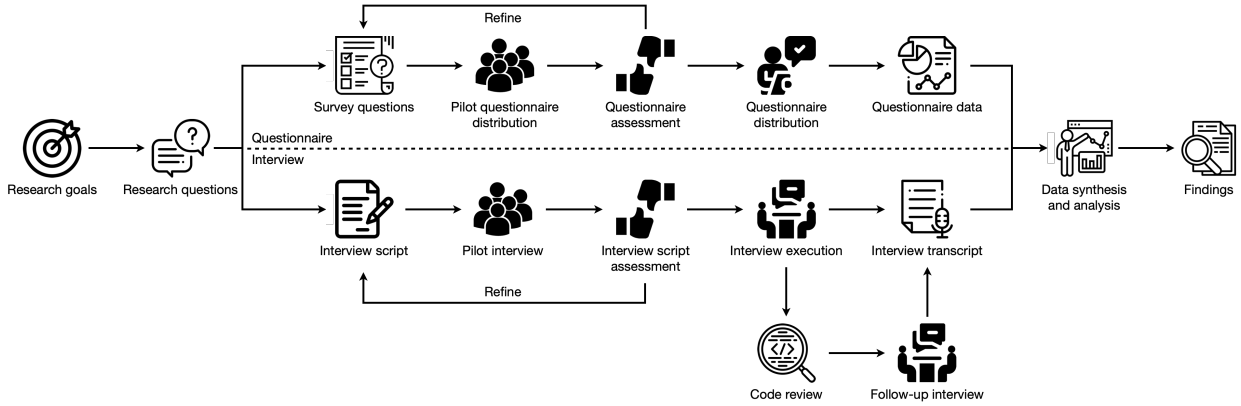

Fig. 1. The research process.

1) Draft preparation: In order to collect the required evidence in a relatively formal way, a questionnaire and an interview script need to be prepared before carrying out the study. We devised the questionnaire and the interview script in iterations to guarantee their effectiveness in helping us obtain the necessary evidence. A series of meetings involving all the authors were held to prepare the first version. The focus of these meetings was to make sure that the topics relevant to the RQs were all covered. We also sought to control the size of the questionnaire by removing potentially redundant questions and merging the questions if the answers to them contained similar information or the answers to one question could be derived from the answers to other questions. More importantly, for the purpose of triangulation, we made sure that any information that could come from both sources of evidence was indeed covered in both questionnaire and interview.

2) Piloting: We conducted several pilot studies whose feedback was used to improve the quality of the designed questionnaire and that of the prepared interview script. For the questionnaire, we applied a non-systematic sampling strategy to carry out 15 pilot studies as the individuals of the entire population were unknown, and it was a small-scale survey [31]. As we deployed the survey on a publicly accessible website, the survey was self-recruited in which the respondents were able to get to know the survey questions and also free to decide whether to participate, and consequently, the individuals of the entire population were unknown. Through this non-systematic sampling process, we contacted and selected convenient persons to act as respondents based on their available time and working experience relevant to logging practices, who were given access to the draft questionnaire and asked to finish the online survey in one week. We specifically sough feedback on the questionnaire, e.g., the difficulty in understanding the questions, the degree of comfort to provide answers, and possible missing parts. All the gathered information was analyzed and discussed by the entire research team to improve the questionnaire. As the last step, we optimized the questionnaire according to a guideline [31] by further re-calibrating the order of questions and the skip-patterns. The final version of the questionnaire consisted of 23 closed-ended and 3 open-ended questions.[1] The first 6 questions were designed to obtain respondents' demographic information such as backgrounds and roles. The subsequent 17 questions were designed to acquire their attitudes towards logging practices, while the remaining 3 open-ended questions were for us to facilitate the subsequent interviews.

As the questionnaire was not suitable for setting up and processing open-ended questions, we conducted a series of interviews to obtain fine-grained data. Another reason was that the respondents may have a cognitive bias when answering the questionnaire, i.e. the actual logging practices may be inconsistent with what respondents answered in the questionnaire. We carried out 5 pilot studies to improve the interview script. However, the major difference between the ways we treated the questionnaire and the interview script is that interviews rely heavily on conversation; therefore, in addition to verifying the content of the questions on the interview script, we focused more on the way the questions were raised, e.g., we prefer asking 'how-to' questions to encourage more conversation. Finally, we formulated totally 21 open-ended questions and designed semi-structured interviews, where questions were planned but not necessarily asked in the same order as they were listed, as semi-structured interviews allow for improvisation and exploration of the issues raised in the conversation.

C. Study execution

In this section, we describe the execution of our study, first the questionnaire survey and then interviews.

1) Questionnaire survey: We deployed the questionnaire survey on Tencent questionnaire, which provides rich features to support questionnaire preparation, release and recovery, and statistical analysis. The questionnaire was then distributed through a variety of channels, including social media, instant messaging, and technical summits. Besides, we also invited experts from several top-tier companies based on the scale of both their employees and the use base of their products. We received a total of 91 valid responses from 42 companies.

[1]The questionnaire and the interview script are accessible at https://figshare.com/s/6b97f36f4ea75994360f.

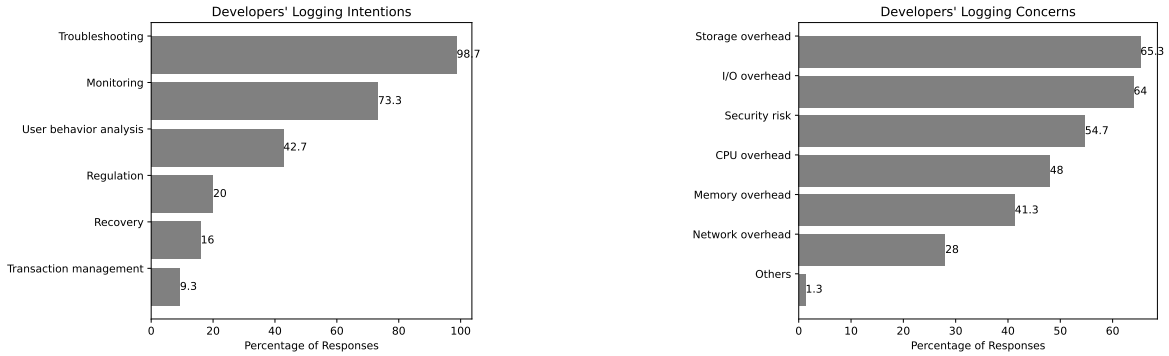

Fig. 2. Developers' common logging intentions and the associated concerns.

2) Semi-structured interviews: We recruited volunteers from the respondents who participated in the questionnaire survey. In total, we interviewed 20 developers from 12 companies, and the average duration of each interview was 34 minutes. To collect the interview data, we documented key points and audio-recorded the interviews throughout as a backup for the subsequent analysis. Two researchers participated in all the interviews, during which one of them asked questions, while the other supplemented questions and also acted as the timekeeper. It is worth noting that we added code analysis at the end of each interview in order to further substantiate the interviewees' actual logging practices in their work as a triangulation mechanism. In the code analysis, we asked the interviewees to provide representative code snippets they had written that contained log statements, which should, but may not, be in line with the answers they provided regarding log placement. By checking the log statements in these code snippets, we were able to assess the actual extent the interviewees' *I&Cs* were correctly fulfilled in source code. When major inconsistencies occurred between the interview answers and results from code analysis, we conducted follow-up interviews to uncover the root causes.

D. Data synthesis and analysis

To answer the research questions, we applied both quantitative and qualitative data synthesis and analysis methods, as both types of data were involved in this study. These methods include descriptive statistics for synthesizing and analyzing the quantitative data involved in the questionnaire survey, thematic analysis for identifying developers' common logging *I&Cs*, and narrative analysis for understanding how interviewees expressed their logging *I&Cs*. Consensus is built among the entire research team in iterations. For example, to perform the coding in thematic analysis, the entire research team worked together to reach consensus by applying a simple voting rule where one voter can *'agree/disagree/be neutral'* with a decision and a decision can only accepted if it has at least one advocate and no opponent. Any disagreement was openly discussed and a new round of voting was then conducted.

IV. Results

This section analyzes the results and presents the findings from the mixed-method study.

A. RQ1: Influence of developers' profiles on their logging I&Cs

We first report the common *I&Cs* identified in this study.

1) Common I&Cs: According to the results of the questionnaire shown in Fig. 2, almost all developers mentioned that their primary logging intentions were for troubleshooting (98.7% of the responses) and monitoring (73.3%), followed by user behavior analysis (42.7%), regulation (20%), recovery (16%), and transaction management (9.3%). Our interviews also confirmed that the primary logging intentions were for troubleshooting and monitoring, which were claimed by 14 (70% of interviewees) and 7 (35%) interviewees respectively and consistent with the findings in previous studies [21], [22].

> *"The logs collected by our project are currently used mainly for problem diagnosis and data monitoring." – Developer h3@Company H*

Figure 2 shows that I/O and storage overheads were deemed the most important concerns, mentioned by 65.3% and 64% of the respondents respectively, followed by security risk (54.7%) and overheads of CPU (48%), memory (41.3%), and network (28%). I/O cost is usually the main bottleneck of a software system compared to other overheads [32], e.g., CPU overhead. Thus, developers pay more attention to the impact of log statements on I/O, which was confirmed by the interview and consistent with the finding in a previous study [22].

> *"Some situations may result in a large amount of logs, which can be very disruptive to the execution of the program. Such output of massive logs should be seriously avoided." – Developer h4@Company H*

Another widely recognized concern was security risk, i.e. log statements may contain sensitive information, which again was highlighted by the majority of the respondents. Although it is not a brand new concern, according to a recent systematic review [17], the fact that more than half of the respondents mentioned this concern highlights the increasing awareness of information security risks when conducting logging practices.

> *"There are strict requirements for the security of log statements, and sensitive information should not be recorded in the logs." – Developer b1@Company B*

Finding 1

While troubleshooting and monitoring, and the overheads of I/O and storage were the primary logging *I&Cs* respectively as reported by previous studies, this mixed-method study confirmed the finding and more importantly identified an additional primary logging concern — security risk.

2) Relationship between developers' profiles and their logging I&Cs: To understand whether logging *I&Cs* are developer-dependent, we conducted a cross analysis between developers' logging *I&Cs* and their profiles in terms of their roles and years of working respectively. The roles are adapted from a survey questionnaire[2] on Microsoft's Log2 project [7]. The years of working are based on our interactions with industrial practitioners who often classify themselves as novice (<1 year), junior (1–3 years), intermediate (3–5 years), experienced (5–10 years), and very experienced (>10 years). With the results in Table II and Table III, we make the following observations.

1) Troubleshooting and monitoring are the primary logging intentions across all developers, regardless of their roles and years of working.
2) Developers with specific roles (e.g., database administrator and user experience designer) may have additional primary logging intentions (e.g., regulation, transaction management, and user behavior analysis).
3) I/O and storage overheads are the primary logging concerns across all developers, regardless of their roles and years of working.
4) There is no clear relationship between developers' logging concerns and their roles, however, the focus of each role's concerns is not consistent.
5) Developers with less than one year of working hardly consider any security risk.
6) Developers with more years of working are more concerned about all types of identified logging side effects.

Finding 2

Developers' profiles influence their logging *I&Cs* in that: (1) developers with specific roles may have additional primary logging intentions, and (2) the obvious dividing line is 1 year of working experience. Those with less than one year of working have significantly less logging intentions and are much less concerned about the side effects than others, implying that a comprehensive awareness of the importance of *I&Cs* in logging practices can be established relatively quickly, but it grows slowly with time.

B. RQ2: Influence of developers' experiences on the way their logging I&Cs are fulfilled

We first get developers' overall perception of the extent to which their logging *I&Cs* are fulfilled through placement of log statements in source code and then analyze how their experiences may impact the fulfillment of their logging *I&Cs*.

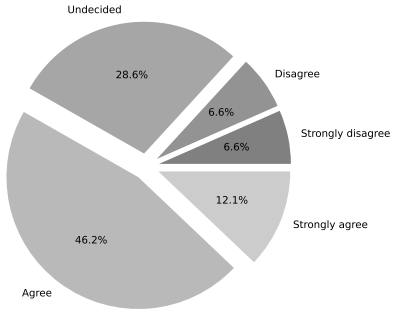

Fig. 3. In practices, does the information contained in the log file generated by the final log statement meet your expectations?

1) RQ2.1: Extent to which log statements are in line with developers' logging I&Cs: After analyzing the answers to the question in Fig. 3, we found that 46.2% of the respondents thought information recorded in log statements was fairly consistent with expectations, 12.1% thought it was fully consistent, and 6.6% did not comply at all. Most respondents were satisfied with the results of their logging practices, which was confirmed by 13 (65%) interviewees. However, 5 (25%) interviewees also mentioned that logging practices that met their expectations underwent multiple rounds of optimization.

> *"Existing logging practices can meet expectations to a large extent, but it usually takes two or three iterations to get it more perfect." – Developer b3@Company B*

However, based on the results of code analysis, it was revealed that for 16 (80%) interviewees, the way the log statements were implemented in the code they had written differed from that they had previously stated. This discovery suggests that the extent to which log statements are in line with developers' logging *I&Cs* is actually lower than what they perceive. Common inconsistencies including *missing log statements* (12 interviewees, 60%), *redundant log statements* (6, 30%), *wrong level/content of log statements* (4, 20%), and *security issues* (2, 10%). For example, after reviewing 1,568 lines of code from one interviewee, 14 inconsistencies were exposed. Based on our follow-up interviews, interviewees acknowledged that the main reason for these inconsistencies was developers' negligence, and, in a few cases, was to meet specific business requirements, which is consistent with the finding in a previous study [22].

> *"There is no log statement instrumented in the 'Catch' block because it was omitted when copying and pasting from other code snippets." – Developer a1@Company A*

2) RQ2.2: Timing of conducting logging practices by developers: According to the survey responses, developers' *I&Cs* were fulfilled in different development stages [33], [34], primarily in the coding stage (81.3%), followed by testing (48%), detailed design (45.3%), operations (40%), system

[2]Also available at https://figshare.com/s/6b97f36f4ea75994360f.

TABLE II
DISTRIBUTION OF LOGGING I&CS OVER DEVELOPERS' ROLES.

		Developer's role								
		R&D‡	**Operator**	**Tester**	**PM**‡	**TS**‡	**DBA**‡	**UXD**‡	**Researcher**	**Others**
Intention	**Troubleshooting**	71 (97.3%)	13 (100.0%)	8 (100.0%)	8 (100.0%)	4 (100.0%)	2 (100.0%)	3 (100.0%)	5 (100.0%)	3 (75.0%)
	Monitoring†	55 (75.3%)	9 (69.2%)	4 (50.0%)	7 (87.5%)	4 (100.0%)	1 (50.0%)	3 (100.0%)	5 (100.0%)	2 (50.0%)
	Regulation†	13 (17.8%)	4 (30.8%)	3 (37.5%)	2 (25.0%)	1 (25.0%)	2 (100.0%)	2 (66.7%)	2 (40.0%)	3 (75.0%)
	Transaction management†	5 (6.8%)	4 (30.8%)	3 (37.5%)	4 (50.0%)	1 (25.0%)	2 (100.0%)	2 (66.7%)	4 (80.0%)	1 (25.0%)
	Recovery†	10 (13.7%)	3 (23.1%)	2 (25.0%)	2 (25.0%)	1 (25.0%)	0 (0.0%)	1 (33.3%)	2 (40.0%)	1 (25.0%)
	User behavior analysis	26 (35.6%)	6 (46.2%)	3 (37.5%)	3 (37.5%)	1 (25.0%)	2 (100.0%)	2 (66.7%)	3 (60.0%)	1 (25.0%)
Concern	**CPU overhead**	38 (52.1%)	6 (46.2%)	5 (62.5%)	4 (50.0%)	2 (50.0%)	0 (0.0%)	2 (66.7%)	3 (60.0%)	4 (100.0%)
	I/O overhead	47 (64.4%)	8 (61.5%)	5 (62.5%)	4 (50.0%)	3 (75.0%)	1 (50.0%)	2 (66.7%)	4 (80.0%)	4 (100.0%)
	Memory overhead	27 (37.0%)	7 (53.8%)	6 (75.0%)	2 (25.0%)	3 (75.0%)	2 (100.0%)	2 (66.7%)	3 (60.0%)	2 (50.0%)
	Network overhead	20 (27.4%)	3 (23.1%)	2 (25.0%)	2 (25.0%)	0 (0.0%)	1 (50.0%)	1 (33.3%)	2 (40.0%)	2 (50.0%)
	Storage overhead	48 (65.8%)	10 (79.6%)	5 (62.5%)	6 (75.0%)	2 (50.0%)	1 (50.0%)	3 (100.0%)	4 (80.0%)	3 (75.0%)
	Security risk	35 (47.9%)	8 (61.5%)	7 (87.5%)	8 (100.0%)	2 (50.0%)	1 (50.0%)	3 (100.0%)	4 (80.0%)	4 (100.0%)

† **Monitoring** refers to the logging of runtime status of a program, such as the time taken for a code fragment to execute; **Regulation** refers to that logging is a work requirement set by the company or project team; **Transaction management** refers to management of transactions by using logs; **Recovery** refers to the use of information recorded in logs to restore a failed service.
‡ **R&D** (Research&Development engineer); **PM** (Project Manager); **TS** (Technical Support); **DBA** (DataBase Administrator); **UXD** (User eXperience Designer).

TABLE III
DISTRIBUTION OF LOGGING I&CS OVER DEVELOPERS' YEARS OF WORKING.

		Developer's working experience				
		Under 1 year	**1 to 3 years**	**3 to 5 years**	**5 to 10 years**	**Over 10 years**
Intention	**Troubleshooting**	6 (100.0%)	12 (92.3%)	19 (100.0%)	31 (93.9%)	20 (100.0%)
	Monitoring†	3 (50.0%)	10 (76.9%)	13 (68.4%)	22 (66.7%)	18 (90.0%)
	Regulation†	2 (33.3%)	3 (23.1%)	5 (26.3%)	4 (12.1%)	5 (25.0%)
	Transaction management†	0 (0.0%)	1 (7.7%)	2 (10.5%)	3 (9.1%)	6 (30.0%)
	Recovery†	0 (0.0%)	3 (23.1%)	4 (21.1%)	4 (12.1%)	3 (15.0%)
	User behavior analysis	1 (16.7%)	4 (30.8%)	4 (21.1%)	15 (45.5%)	8 (40.0%)
Concern	**CPU overhead**	2 (33.3%)	5 (38.5%)	12 (63.2%)	14 (42.4%)	15 (75.0%)
	I/O overhead	5 (83.3%)	6 (46.2%)	14 (73.7%)	19 (57.6%)	15 (75.0%)
	Memory overhead	2 (33.3%)	5 (38.5%)	9 (47.4%)	12 (36.4%)	9 (45.0%)
	Network overhead	1 (16.7%)	3 (23.1%)	5 (26.3%)	10 (30.3%)	6 (30.0%)
	Storage overhead	4 (66.7%)	8 (61.5%)	8 (42.1%)	23 (69.7%)	15 (75.0%)
	Security risk	0 (0.0%)	6 (46.2%)	13 (68.4%)	18 (54.5%)	13 (65.0%)

† The same as Table II.

TABLE IV
DISTRIBUTION OF THE MAIN SDLC STAGES WHERE LOGGING PRACTICES ARE PERFORMED OVER DEVELOPER'S YEARS OF WORKING.

Stage	Developer's working experience				
	Under 1 year	**1 to 3 years**	**3 to 5 years**	**5 to 10 years**	**Over 10 years**
Requirement engineering	2 (33.3%)	1 (7.7%)	3 (15.8%)	10 (30.3%)	12 (60.0%)
System design	0 (0.0%)	2 (15.4%)	5 (26.3%)	13 (39.4%)	12 (60.0%)
Detailed design	2 (33.3%)	3 (23.1%)	9 (47.4%)	16 (48.5%)	12 (60.0%)
Coding	6 (100.0%)	12 (92.3%)	15 (78.9%)	24 (72.7%)	17 (85.0%)
Testing	4 (66.7%)	6 (46.2%)	7 (36.8%)	13 (39.4%)	10 (50.0%)
Operations	2 (33.3%)	4 (30.8%)	5 (26.3%)	12 (36.4%)	12 (60.0%)

design (30.7%), and requirements engineering (26.7%), as show in Fig. 4.

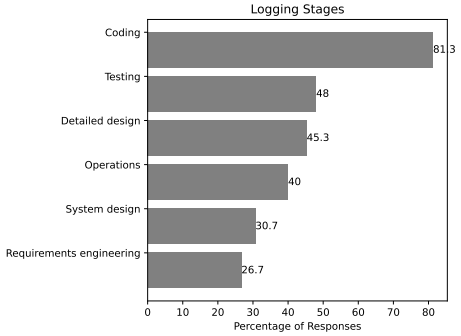

Fig. 4. Development stages where logging *I&Cs* are fulfilled.

Through a cross analysis between the development stages where logging *I&Cs* were fulfilled and developers' years of working, as shown in Table IV, we observed that most developers performed logging practices in the coding stage regardless of their years of working.

> *"I only consider inserting log statements when I am coding. Usually it is a subjective judgment of where the log statements need to be inserted." – Developer c1@Company C*

It was further discovered that more experienced developers tend to consider logging practices in more and earlier stages. For example, developers with more than 10 years of working often spread logging practices over the entire software development life cycle (SDLC).

> *"The way the logs are output and stored is taken into account during project design, such as whether the logs are output at module or class granularity or through traces; whether the logs need to be fed into the monitoring system; and whether the logs are stored locally or centrally." – Developer h3@Company H*

Figure 5 shows the relationship between the extent to which log statements satisfy logging *I&Cs* and the SDLC stage during which developers consider log placement. A higher value (1–5) means that the current logging practice is more capable of satisfying logging *I&Cs*. In general, the earlier the log placement is taken into consideration in the SDLC, the better the logging practices reflect the developers' logging *I&Cs*. The *Mann Whitney U tests* for the differences between 'Requirement engineering' and 'coding' and between 'system design' and 'coding' resulted in both p-values less than 0.05, indicating the null hypothesis (i.e. no significant difference between the two sets of data) can be rejected.

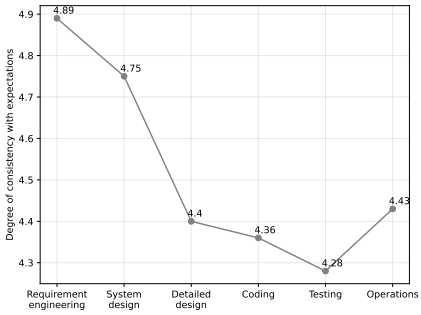

Fig. 5. Influence of the timing of conducting logging practices on the extent of fulfilling *I&Cs*.

3) RQ2.3: Role of logging guidelines/coding conventions: According to the results of the questionnaire, 48.0% of the respondents would follow logging guidelines to a large extent, 26% always, while only 4% would not follow them at all, as shown in Fig. 6.

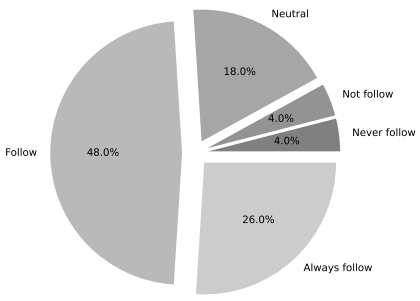

Fig. 6. Developers' tendency to following logging guidelines.

Furthermore, as listed in Table V, the primary guideline developers would follow is for defining log statement severity

(80%), followed by that for the types of code snippets requiring log statements (62%), those for choosing logging variables and appropriate logging severity as well for formatting log statements (all 58%), that for regulating log statements (48%), and that for reducing logging overhead (44%).

TABLE V
LOGGING GUIDELINES.

Reason	# respondents	Percentage
Guideline for defining log statement severity	72	80%
Guideline for the types of code snippets requiring log statements	56	62%
Guideline on choosing logging variables	52	58%
Guideline on choosing appropriate log statement severity	52	58%
Guideline on formatting log statements	52	58%
Guideline on regulating log statements	43	48%
Guideline on reducing logging overhead	40	44%

Developers' tendency to following logging guidelines is inherently influenced by their profiles, especially their years of working, as shown in Fig. 7. We observed from the questionnaire that developers with less than one year of working are more likely to recognize the usefulness of the logging guidelines, however, this attitude tends to decrease first and then increase sharply with increased years of working experience, which was also confirmed by the interviews.

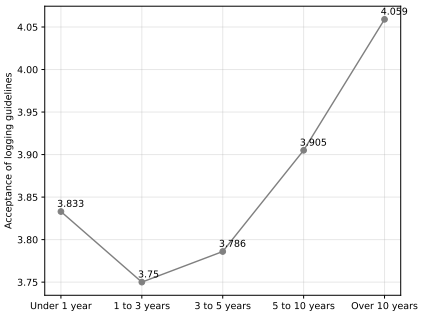

Fig. 7. Influence of developers' years of working on their tendency to following logging guidelines.

On the one hand, novice developers are likely to require some guidance for conducting logging practices. After they have gained some working experience and become reasonably familiar with logging practices, their reliance on guidelines would decrease gradually because a basic logging practice guide is not much useful to developers with some experience, whereas sophisticated questions about logging practices are unlikely to be answered by logging guidelines. On the other hand, developers with more than three years of working re-endorse the value of logging guidelines often because they are likely involved in technical management work and possibly need to lead novice developers. At this point, they begin to recognize the role logging guidelines play in standardizing logging practices. As their years of working increase, the size of the team they manage is likely to grow, and as such, logging normalization becomes more important, and accordingly, the guidelines become more widely recognized.

> *"Logging guidelines are very useful for development and beneficial for troubleshooting errors. At the beginning of my career, I had more confusion about how to instrument log statements and how to use them to troubleshoot issues, and if logging guidelines existed, I could solve these problems without asking for help." – Developer e1@Company E*

Figure 8 illustrates the influence of using logging guidelines on log placement in terms of the extent to which the logged information satisfies the logging *I&Cs*. Evidently, logging practices with the adoption of logging guidelines satisfy the developers' logging *I&Cs* better than those without them, which was also confirmed by 17 (85%) interviewees. The *Mann Whitney U test* on the mean value resulted in a less than 0.05 p-value, indicating that the null hypothesis (i.e. no significant difference between the two sets of data) can be rejected.

> *"The logging guidelines regulate the team's most basic logging practice requirements, which are some of the running behavior information that must be recorded during the development and maintenance of software systems. The logging guidelines allow team members to quickly and accurately carry out the required logging practices, and also reduce a lot of communication and management costs." – Developer b4@Company B*

Fig. 8. Influence of adopting logging guidelines on the fulfillment of *I&Cs*.

Finding 3

Logging practices are mainly conducted in the coding stage and only partially fulfill developers' *I&Cs*, regardless of their profiles and experiences. However, developers' experiences influence the way their *I&Cs* are fulfilled in that: (1) experienced developers with over 10 years of working tend to consider logging practices in early development stages, which leads to higher satisfaction of fulfilling logging *I&Cs*, (2) novice (less than 1 year of working) and experienced (more than 3 years of working) developers are more prone to accepting logging guidelines than intermediate developers, and (3) developers who adopt logging guidelines better fulfill their logging *I&Cs* than those who do not.

V. Discussion

In this section, we first recommend good logging practices and then suggest important future research directions.

A. Recommended good logging practices

Through this empirical study involving participants from 42 companies across a variety of industrial domains, we found that developers share common logging intentions (troubleshooting and monitoring) and concerns (I/O and storage overheads) and conduct logging practices mainly in the coding stage, which only partially fulfill their *I&Cs*, regardless of their roles and working experience. A deeper investigation unveiled that developers' profiles and experiences do influence their logging *I&Cs* and the way their *I&Cs* are fulfilled, Based on Findings 1, 2, and 3, we make the following recommendations.

- Create a habit of considering log placement in early software development stages such as requirements engineering and system design. The earlier the log placement is considered, the more likely it is able to satisfy the developers' expectations of logging practices.
- Adopt available logging guidelines such as those for defining log statement severity and those for the types of code snippets requiring log statements. Developers who adopt logging guidelines better fulfill their logging *I&Cs* than those who do not.
- Provide appropriate training of logging practices to novice developers with less than one year of working experience as they can quickly establish a comprehensive awareness of the importance of logging *I&Cs*.
- Provide relevant training or specific guidelines to raise the awareness of other side effects associated with logging practices in addition to the primary concerns of I/O and storage overheads among junior developers with less than three years of working experience.
- Provide relevant training or specific guidelines to raise the awareness of security issues involved in logging practices among novice developers with less than one year of working experience.
- Provide relevant training or specific guidelines to raise the awareness of important logging intentions in addition to troubleshooting and monitoring among developers with specific roles such as database administrator and user experience designer.

B. Suggested future research directions

The importance of logging guidelines has been widely recognized [2], [18], [23], [25], [26], [35], [36]. Similar requests exist in industry, with several popular blogs discussing the best or worst logging practices, e.g., [37], [38], which could be taken as reference guidelines. Some world-leading software companies have also introduced internal guidelines for logging practices, e.g., [30]. In addition to confirming these known facts, our study sheds new light on some of the more specific mechanisms of logging guidelines. That is, as a vehicle for carrying relevant experience, logging guidelines are able to serve as a materialization of logging *I&Cs* in the early stages of SDLC, guiding and standardizing the implementation of subsequent log statements for novice developers and facilitate management activities for senior developers/managers. Although most of these guidelines are so-called general-purpose guidelines which tend to only deal with basic logging issues and can hardly cope with some more advanced requirements of logging practices, their role in assisting in fulfilling relatively comprehensive logging *I&Cs* is still undeniable. More sophisticated logging practices in addressing advanced *I&Cs* can be transformed into proper expressions and added to these general-purpose guidelines.

TABLE VI
Logging guidelines and supporting tools.

Wishlist	# respondents	Percentage
Automatic logging tools for instrumenting log statements	60	65.9%
Pragmatic logging guidelines	57	62.6%
Logging guidelines/tools for improving the quality of log design	37	40.7%
Automatic logging tools for assessing the validity of log statements	36	39.6%
Features of automatic logging tools		
Instrument log statements semi-automatically or fully automatically	67	73.6%
Record the required information without being affected by changes of business code	58	63.7%
Can be used without domain knowledge	41	45.1%
Optimize the content and location of existing log statements automatically	30	33.0%

Supporting tools should also work with certain rules in line with guidelines [39] or models derived from machine learning techniques [40]. In this sense, tools play a natural role to enact logging guidelines and fulfill logging *I&Cs*. For example, Jia et al. proposed two models to describe logging intentions, and further designed and implemented an automatic log placement tool based on the intention models [41]. Our study revealed that nearly two-thirds of the respondents believe that automatic logging tools and pragmatic logging guidelines can help developers fulfill logging *I&Cs* more easily when conducting logging practices, as shown in Table VI. For example, using logging guidelines/tools for improving the quality of log design and using automatic tools for assessing the validity of log statements accounted for 40.7% and 39.6% of the respondents respectively. Specifically, for automatic logging tools, 73.6% of the respondents wanted them to help developers semi-automatically or automatically log the information they need, 63.7% wanted them to be able to collect information independent of changes in business logic, 45.1% wanted them to be used without requiring domain knowledge, and 33.0% wanted them to automatically optimize the content and location of existing log statements. These remarks to a certain degree point out future research directions and research priorities regarding logging tools.

> *"Since developers tended to write log statements in different styles despite the logging guidelines, it is preferred to automate logging so that developers pay less attention to instrument log statements." –Developer b2@Company B*

VI. Threats to validity

This section discusses several validity risks.

A. Internal validity

Internal validity is the extent to which a study establishes a trustworthy cause-and-effect relationship between a treatment and an outcome. In this study, the main threat to the internal validity arises from the personal bias in the interpretation of the data from questionnaire surveys and interviews. In order to control this validity risk, an iterative strategy was applied in the process of data synthesis and analysis in which the interpretation was performed collaboratively by multiple researchers with cross-checking. Therefore, the threats derived from personal bias can be mitigated. Another noteworthy threat to the internal validity is that we did not take into account the difference between organizations/product lines in the analysis of survey results. The consideration is two-folded. Firstly, for RQ1, the different roles reasonably reflect the potential difference existing in different organizations/product lines. Secondly, for RQ2, according to the research conducted by Pecchia et al. [21], fulfillment of logging intentions in source code shares common patterns despite the adoption of different logging procedures and the lack of common rules. In this sense, we deem this threat factor can be well controlled.

B. External validity

External validity is concerned with the extent to which the results and findings can be generalized. One possible threat is related to the representativeness of the respondents and interviewees. We received 91 valid responses in the questionnaire survey and interviewed 20 volunteer developers in the interviews. Therefore, the results and findings in this study may not reflect all the real status regarding logging practice in industry. To mitigate the impact of this threat, we expanded the scope of the questionnaire survey as much as possible, e.g., distributing the questionnaire using as many channels as possible, and selecting developers from different companies for our study. As the result, we have respondents and interviewees from 42 companies, which to a fair degree minimizes the external validity risks. A closely related threat is the same cultural background of the respondents and interviewees as all the 42 companies are based in the same geographical location. To address this risk, we share the questionnaire and the interview script for other researchers to replicate this study in different geographical locations.

C. Construct validity

Construct validity is concerned with the issues affecting the extent to which the object of study truly represents theory behind the study [42]. In this study, the main treat related to this validity is the suitability of survey questions in the questionnaire and the interview script. In the process of designing the questionnaire questions and the interview questions, we conducted pilot surveys in order to assess and refine the questions in iterations, as described in Section III. By this way, this threat can be largely mitigated.

VII. Conclusion

Logging is an important part of software development, which captures the dynamic behavior of software systems to facilitate other critical quality practices such as troubleshooting. Despite that a lot of research effort has been made on logging practices, what common *I&Cs* developers share, to what degree they are fulfilled in daily development, and what factors contribute to the differences among the developers in logging *I&Cs* and the way they are fulfilled remain unclear.

In this paper, we carried out an empirical study involving participants from a wide range of companies across a variety of industrial domains to characterize the developers' *I&Cs* and further investigate how the *I&Cs* and the way they are fulfilled are influenced by developers' profiles and experiences. We have made several interesting findings through this mix-method study. First, while several common *I&Cs* do exist, they also change with time. For example, *security risk* became a common longing concern nowadays, yet a recent systematic study indicates that this factor has only been covered in a small amount of prior work in research and practice [17]. Besides, it is clear that developers with more working experience tend to have more diverse logging *I&Cs*, indicating that logging *I&Cs* may change along with the increasing working experience to address more sophisticated problems in software development. Last but not least, logging *I&Cs* are only partially fulfilled in industrial projects. How well logging *I&Cs* are fulfilled depends on how early logging practices are conducted and whether developers adopt logging guidelines.

Based on the findings, we have recommended some good logging practices such as providing training of logging practices to novice developers with less than one year of working experience, providing relevant training or specific guidelines to raise the awareness of important side effects associated with logging practices among junior developers and the awareness of important logging intentions among developers with specific roles, creating a habit of considering log placement in early software development stages, and adopting available logging guidelines.

Acknowledgment

This work is jointly supported by the National Key Research and Development Program of China (No.2019YFE0105500) and the Research Council of Norway (No.309494), the Meituan, the Key Research and Development Program of Jiangsu Province (No.BE2021002-2), as well as the National Natural Science Foundation of China (No.62072227, No.62202219). Shenghui Gu is the corresponding author.

References

[1] T. Barik, R. DeLine, S. Drucker, and D. Fisher, "The bones of the system: A case study of logging and telemetry at microsoft," in *Proceedings of the 38th International Conference on Software Engineering Companion*. Association for Computing Machinery (ACM), may 2016, pp. 14–22. [Online]. Available: https://doi.org/10.1145/2889160.2889231

[2] P. He, Z. Chen, S. He, and M. R. Lyu, "Characterizing the natural language descriptions in software logging statements," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. Association for Computing Machinery (ACM), sep 2018, pp. 178–189. [Online]. Available: https://doi.org/10.1145/3238147.3238193

[3] Q. Fu, J.-G. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," in *2009 Ninth IEEE International Conference on Data Mining*. Institute of Electrical and Electronics Engineers (IEEE), dec 2009, pp. 149–158. [Online]. Available: https://doi.org/10.1109/icdm.2009.60

[4] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles (SOSP)*. Association for Computing Machinery (ACM), oct 2009, pp. 117–132. [Online]. Available: https://doi.org/10.1145/1629575.1629587

[5] D. Yuan, H. Mai, W. Xiong, L. Tan, Y. Zhou, and S. Pasupathy, "SherLog: Error diagnosis by connecting clues from run-time logs," in *Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems (ASPLOS)*. Association for Computing Machinery (ACM), mar 2010, pp. 143–154. [Online]. Available: https://doi.org/10.1145/1736020.1736038

[6] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen, "Performance debugging for distributed systems of black boxes," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 74–89, dec 2003. [Online]. Available: https://doi.org/10.1145/1165389.945454

[7] R. Ding, H. Zhou, J.-G. Lou, H. Zhang, Q. Lin, Q. Fu, D. Zhang, and T. Xie, "Log2: A cost-aware logging mechanism for performance diagnosis," in *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. Santa Clara, CA: USENIX Association, jul 2015, pp. 139–150. [Online]. Available: https://www.usenix.org/conference/atc15/technical-session/presentation/ding

[8] B. Chen, J. Song, P. Xu, X. Hu, and Z. M. J. Jiang, "An automated approach to estimating code coverage measures via execution logs," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. Association for Computing Machinery (ACM), sep 2018, pp. 305–316. [Online]. Available: https://doi.org/10.1145/3238147.3238214

[9] F. Horváth, T. Gergely, Á. Beszédes, D. Tengeri, G. Balogh, and T. Gyimóthy, "Code coverage differences of java bytecode and source code instrumentation tools," *Software Quality Journal*, vol. 27, no. 1, pp. 79–123, dec 2017. [Online]. Available: https://doi.org/10.1007/s11219-017-9389-z

[10] 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," *IEEE Transactions on Software Engineering*, vol. 47, no. 12, pp. 2858–2873, dec 2021. [Online]. Available: https://doi.org/10.1109/tse.2020.2970422

[11] D. Yuan, S. Park, and Y. Zhou, "Characterizing logging practices in open-source software," in *2012 34th International Conference on Software Engineering (ICSE)*. Institute of Electrical and Electronics Engineers (IEEE), jun 2012, pp. 102–112. [Online]. Available: https://doi.org/10.1109/icse.2012.6227202

[12] X. Zhao, K. Rodrigues, Y. Luo, M. Stumm, D. Yuan, and Y. Zhou, "Log20: Fully automated optimal placement of log printing statements under specified overhead threshold," in *Proceedings of the 26th Symposium on Operating Systems Principles*. Association for Computing Machinery (ACM), oct 2017, pp. 565–581. [Online]. Available: https://doi.org/10.1145/3132747.3132778

[13] G. Rong, S. Gu, H. Zhang, D. Shao, and WanggenLiu, "How is logging practice implemented in open source software projects? a preliminary exploration," in *2018 25th Australasian Software Engineering Conference (ASWEC)*. Institute of Electrical and Electronics Engineers (IEEE), nov 2018, pp. 171–180. [Online]. Available: https://doi.org/10.1109/aswec.2018.00031

[14] B. Chen and Z. M. J. Jiang, "A survey of software log instrumentation," *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–34, may 2022. [Online]. Available: http://dx.doi.org/10.1145/3448976

[15] J. Cândido, M. Aniche, and A. van Deursen, "Log-based software monitoring: A systematic mapping study," *PeerJ Computer Science*, vol. 7, p. e489, may 2021. [Online]. Available: https://doi.org/10.7717/peerj-cs.489

[16] G. Rong, Q. Zhang, X. Liu, and S. Gu, "A systematic review of logging practice in software engineering," in *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*. Institute of Electrical and Electronics Engineers (IEEE), dec 2017, pp. 534–539. [Online]. Available: http://dx.doi.org/10.1109/APSEC.2017.61

[17] S. Gu, G. Rong, H. Zhang, and H. Shen, "Logging practices in software engineering: A systematic mapping study," *IEEE Transactions on Software Engineering*, pp. 1–1, 2022. [Online]. Available: https://doi.org/10.1109/tse.2022.3166924

[18] J. Zhu, P. He, Q. Fu, H. Zhang, M. R. Lyu, and D. Zhang, "Learning to log: Helping developers make informed logging decisions," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Institute of Electrical and Electronics Engineers (IEEE), may 2015, pp. 415–425. [Online]. Available: https://doi.org/10.1109/icse.2015.60

[19] B. Chen and Z. M. J. Jiang, "Studying the use of Java logging utilities in the wild," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. Association for Computing Machinery (ACM), jun 2020, pp. 397–408. [Online]. Available: https://doi.org/10.1145/3377811.3380408

[20] 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*. Association for Computing Machinery (ACM), may 2014, pp. 24–33. [Online]. Available: https://doi.org/10.1145/2591062.2591175

[21] A. Pecchia, M. Cinque, G. Carrozza, and D. Cotroneo, "Industry practices and event logging: Assessment of a critical software development process," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)*. Institute of Electrical and Electronics Engineers (IEEE), may 2015, pp. 169–178. [Online]. Available: https://doi.org/10.1109/icse.2015.145

[22] G. Rong, Y. Xu, S. Gu, H. Zhang, and D. Shao, "Can you capture information as you intend to? a case study on logging practice in industry," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. Institute of Electrical and Electronics Engineers (IEEE), sep 2020, pp. 12–22. [Online]. Available: https://doi.org/10.1109/icsme46990.2020.00012

[23] B. Chen and Z. M. Jiang, "Characterizing and detecting anti-patterns in the logging code," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. Institute of Electrical and Electronics Engineers (IEEE), may 2017, pp. 71–81. [Online]. Available: https://doi.org/10.1109/icse.2017.15

[24] H. Li, W. Shang, and A. E. Hassan, "Which log level should developers choose for a new logging statement?" *Empirical Software Engineering*, vol. 22, no. 4, pp. 1684–1716, oct 2016. [Online]. Available: https://doi.org/10.1007/s10664-016-9456-2

[25] Z. Liu, X. Xia, D. Lo, Z. Xing, A. E. Hassan, and S. Li, "Which variables should i log?" *IEEE Transactions on Software Engineering*, pp. 1–1, sep 2019. [Online]. Available: https://doi.org/10.1109/tse.2019.2941943

[26] X. Liu, T. Jia, Y. Li, H. Yu, Y. Yue, and C. Hou, "Automatically generating descriptive texts in logging statements: How far are we?" in *Programming Languages and Systems*. Springer International Publishing, nov 2020, pp. 251–269. [Online]. Available: https://doi.org/10.1007/978-3-030-64437-6_13

[27] R. Zhou, M. Hamdaqa, H. Cai, and A. Hamou-Lhadj, "MobiLogLeak: A preliminary study on data leakage caused by poor logging practices," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. Institute of Electrical and Electronics Engineers (IEEE), feb 2020, pp. 577–581. [Online]. Available: https://doi.org/10.1109/saner48275.2020.9054831

[28] J. W. Creswell and J. D. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*. SAGE Publications, 2017.

[29] V. R. Basili, "Goal question metric paradigm," *Encyclopedia of software engineering*, vol. 1, pp. 528–532, 1994.

[30] Alibaba. (2017) Alibaba Java coding guidelines. [Online]. Available: https://alibaba.github.io/Alibaba-Java-Coding-Guidelines/

[31] T. Punter, M. Ciolkowski, B. Freimut, and I. John, "Conducting on-line surveys in software engineering," in *Proceedings of International Symposium on Empirical Software Engineering (ISESE)*. Institute of Electrical and Electronics Engineers (IEEE), oct 2003, pp. 80–88. [Online]. Available: https://doi.org/10.1109/isese.2003.1237967

[32] L. S. Keller, "Operating systems," in *Encyclopedia of Physical Science and Technology*. Elsevier, 2003, pp. 169–191. [Online]. Available: https://doi.org/10.1016/b0-12-227410-5/00851-6

[33] W. W. Royce, "Managing the development of large software systems: Concepts and techniques," in *Proceedings of the 9th international conference on Software Engineering*, mar 1987, pp. 328–338.

[34] N. B. Ruparelia, "Software development lifecycle models," *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 3, pp. 8–13, may 2010. [Online]. Available: https://doi.org/10.1145/1764810.1764814

[35] H. Li, T.-H. P. Chen, W. Shang, and A. E. Hassan, "Studying software logging using topic models," *Empirical Software Engineering*, vol. 23, no. 5, pp. 2655–2694, jan 2018. [Online]. Available: https://doi.org/10.1007/s10664-018-9595-8

[36] H. Anu, J. Chen, W. Shi, J. Hou, B. Liang, and B. Qin, "An approach to recommendation of verbosity log levels based on logging intention," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. Institute of Electrical and Electronics Engineers (IEEE), sep 2019, pp. 125–134. [Online]. Available: https://doi.org/10.1109/icsme.2019.00022

[37] J. Skowronski. (2017, jan) 30 best practices for logging at scale. [Online]. Available: https://www.loggly.com/blog/30-best-practices-logging-scale/

[38] L. Tal. (2017, jan) 9 logging best practices based on hands-on experience. [Online]. Available: https://www.loomsystems.com/blog/single-post/2017/01/26/9-logging-best-practices-based-on-hands-on-experience

[39] M. Cinque, D. Cotroneo, and A. Pecchia, "Event logs for the analysis of software failures: A rule-based approach," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 806–821, jun 2013. [Online]. Available: https://doi.org/10.1109/tse.2012.67

[40] Z. Li, T.-H. P. Chen, and W. Shang, "Where shall we log? studying and suggesting logging locations in code blocks," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. Association for Computing Machinery (ACM), dec 2020, pp. 361–372. [Online]. Available: https://doi.org/10.1145/3324884.3416636

[41] Z. Jia, S. Li, X. Liu, X. Liao, and Y. Liu, "SMARTLOG: Place error log statement by deep understanding of log intention," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. Institute of Electrical and Electronics Engineers (IEEE), mar 2018, pp. 61–71. [Online]. Available: https://doi.org/10.1109/saner.2018.8330197

[42] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer Science & Business Media, jun 2012.