Received: Added at production | Revised: Added at production | Accepted: Added at production
DOI: xxx/xxxx

An Exploratory Semantic Analysis of Logging Questions

Harshit Gujral*[1] | Sangeeta Lal[2] | Heng Li[3]

[1]Department of Computer Science Engineering and Information Technology, Jaypee Institute of Information Technology, Noida, India. Email: harshitgujral12@gmail.com
[2]Lecture Data Science, School of Computing and Mathematics, Keele University, Keele, United Kingdom. Email: sangeeta.69b1@gmail.com
[3]Department of Computer and Software Engineering, Polytechnique Montréal, Montréal, Canada. Email: heng.li@polymtl.ca

Correspondence
*Corresponding author.

Abstract

Logging is an integral part of software development. Software practitioners often face issues in software logging, and they post these issues on Q&A websites to take suggestions from the experts. In this study, we perform a three-level empirical analysis of logging questions posted on six popular technical Q&A websites, namely Stack Overflow (SO), Serverfault (SF), Superuser (SU), Database Administrators (DB), Software Engineering (SE), and Android Enthusiasts (AE). The findings show that logging issues are prevalent across various domains, e.g., database, networks, and mobile computing, and software practitioners from different domains face different logging issues. The semantic analysis of logging questions using Latent Dirichlet Allocation (LDA) reveals trends of several existing and new logging topics, such as logging conversion pattern, android device logging, and database logging. In addition, we observe specific logging topics for each website: DB (Log shipping, Log file growing/shrinking), SU (Event Log, Syslog configuration), SF (Log analysis, Syslog configuration), AE (App Install, Usage tracking), SE (Client server logging, Exception logging), and SO (Log file creation/deletion, Android Emulator Logging, Logger class of log4j). We obtain an increasing trend of logging topics on the SO, SU, and DB websites, whereas a decreasing trend of logging topics on the SF website.

KEYWORDS:
Empirical Software Engineering; Logging; Technical Q&A websites; Latent Dirichlet Allocation; Mining Software Repositories; Software Maintenance

1 | INTRODUCTION

Logging is a crucial software development practice that is used to record essential runtime information about the software. This runtime information is later used by software developers and practitioners for various activities[1 2]. Logs are often the only information available to the software developers for system diagnosis, which makes them significant[3]. At present, several tools and libraries are available in the market for logging. However, similar to other software technologies, logging tools and libraries are also changing rapidly making it challenging for the software practitioners to keep pace with them. It is found that even the experienced software practitioners find difficulties with logging[1 2]. In the literature, several studies concern buildings tools to aid software developers in logging through *proactive logging*[4 5], *learning to log*[2 1 6 7 8], *log level prediction*[9], and *log statement stability prediction*[10]. In particular, these studies either use static analysis approaches to add logging statements in the source code or use machine learning based approaches to build models using existing data, followed by using these models to provide logging suggestions to developers[11].

The aforementioned studies provide an insightful path towards improving the current logging practices and techniques. However, the major limitation of these approaches is that they consider the needs of software developers only. Generally, logging is used by many different stakeholders and software practitioners, e.g., database administrators, network administrators, and general users. There exists a possibility that software developers are not entirely aware of the logging requirements across different software engineering practitioners. Without considering these logging requirements/issues of different software practitioners, the current logging automation or improvement studies may not provide reasonable logging solutions to software developers. Therefore, in this study, we aim to identify various logging issues faced by different software practitioners.

To this end, we perform an exploratory study of logging questions posted on technical Q&A websites. We aim to explore and identify the logging issues faced by various software practitioners and stakeholders. Technical Q&A websites, e.g., Stack Overflow (SO), provide an open platform for software practitioners to ask and discuss the technical issues. These websites feature worldwide experts, and these experts answer the questions of users and share their knowledge. Gradually, these websites have become knowledge repositories that consist of valuable information about the trends of software tools and technologies. Data from these websites can be mined to extract valuable insights to benefit software practitioners. The software engineering research community has already recognized the potential of mining these repositories and has used these data in various applications[12 13 14]. However, logging is still a relatively less explored area with respect to Q&A websites. We believe that mining logging questions asked on these websites can reveal significant insights.

These Q&A websites comprise substantial information about logging; however, analyzing the content of logging questions from these websites is technically challenging owing to three main reasons. First, the volume of the dataset; for example, the SO website consists of approximately 14 million questions, forming a dataset of approximately 55 GB. Mining these considerable amounts of questions is not only time-consuming but also a technically challenge[12]. Second, identifying logging questions from a pool of all questions is not a trivial task because these Q&A websites are open to diverse questions from software practitioners. To rephrase, practitioners can ask questions on any topic if it fits under the domain of the respective Q&A website. Thus, identification and extraction of logging-related questions from all these questions is a challenging task. Third, the content of the posts on these questions is in natural language (i.e., unstructured); hence, traditional data mining techniques may not work on them[15 12]. Thus, we apply advanced data mining techniques to extract useful information from these websites.

In this work, we overcome the aforementioned challenges and perform an exploratory study of logging questions at 3 different levels (refer to Table 3 and Figure S1). At the first level, we identify the frequency and popularity of logging questions across different websites. At the second level, we perform an analysis of popular words and tags of logging questions with respect to different programming languages. At the third level, we perform a semantic analysis of logging questions using topic models. We select six popular Q&A websites, namely Stack Overflow (SO)[16], Serverfault (SF)[17], Superuser (SU)[18], Database Administrators (DB)[19], Software Engineering (SE)[20], and Android Enthusiasts (AE)[21], in our study. These websites largely represent either a distinguished user base or a generic user base, e.g., SO. According to the Stack Exchange network, the questions on these websites adheres to the criteria below:

1. AE: It concerns users of the Android OS, covering questions about (1) using Android device, (2) using a particular app, or (3) addressing an error or other issue in using your Android device[1].

2. DB: It concerns users of traditional SQL RDBMS and NoSQL alternatives, covering questions about (1) database administration, (2) advanced querying, (3) data modelling and database-design, (4) advanced programming in built-in server-side languages, and (5) data warehousing and business intelligence[2].

3. SF: It concerns users managing IT systems in a business environment, covering questions about (1) managing servers, storage or networks, (2) tools used to administer, monitor, or automate, and (3) deployment to and management of third-party provided IT platforms[3].

4. SE: It concerns users working within the systems development life cycle, covering questions about (1) software development methods and practices, (2) requirements, architecture, and design, (3) quality assurance and testing, and (4) configuration management, build, release, and deployment[4].

5. SO: It concerns professional and enthusiast programmers, among others, it covers questions about (1) a specific programming problem, (2) a software algorithm, (3) software tools commonly used by programmers[5].

6. SU: It concerns computer enthusiasts and power users, covering questions about (1) computer hardware, (2) computer software, and (3) personal and home computer networking[6].

It is evident that SO represents a generic knowledge domain, whereas AE, DB, SE, SF, and SU represent a specific knowledge domain. Thus, the analysis of Stack Exchange highlights general logging issues. In contrast, analyses of the rest of the websites highlight logging issues specific

[1]https://android.stackexchange.com/help/on-topic
[2]https://dba.stackexchange.com/help/on-topic
[3]https://serverfault.com/help/on-topic
[4]https://softwareengineering.stackexchange.com/help/on-topic
[5]https://stackoverflow.com/help/on-topic
[6]https://superuser.com/help/on-topic

to the concerned user base of these websites. To this end, moderators, as well as users, on these websites strive to keep content relevant to the concerned website by either transferring or removing the irrelevant content. Notably, since several of these domains are interdependent, an overlap of content among these websites might exist.

Subsequently, we extract approximately 82K logging questions from these six websites and conduct an in-depth empirical analysis. In particular, we answer the following six research questions:

RQ1: What domains are affected by logging? What is the response of the community towards logging questions in each domain?
Logging issues occur on all the six technical Q&A websites (domains) considered in this study; they also receive responses from the community experts. This indicates that logging is an important issue that needs to be considered with respect to various domains.

RQ2: What are the differentiating terms present in the logging questions of each website (domain)?
Word cloud analysis of logging questions indicates the presence of different types of issues in different domains. For example, *Android* developers tend to resolve issues like *crashes, malware, and system errors* with the use of logging. *Database administrators* employ a very specific *shipping-log* technique to ensure prompt database backup and recovery. In *server environment*, *syslog and its frameworks* play a crucial role in the centralized processing of log files in a network. Software engineers on *SE website* focus on the development of *state-of-the-art* logging techniques to improve programming practices and performance. The *SO website* emphasizes a variety of user queries related to *Java, Python, and ELK Stack*. The *SU website* majorly focuses on *event logs* that are particularly designed for troubleshooting issues in the Windows operating system.

RQ3: What are the differentiating terms present in the logging questions of each programming languages?
We have observed that each programming language has its niche in the logging environment, with obvious overlaps. Logging questions of the C programming language are most related to networking and server environments, whereas logging questions of C++ are usually related to multi-threaded and multiprocessing environment. C# is predominantly used to implement logging in a .Net environment (especially in ASP .NET MVC frameworks). Java provides a wide range of logging libraries for Java-based projects and software. JavaScript is used for logging Web Applications with efficient logging libraries like Winston that not only provide a logging framework but also an analytics framework (using Loggly). With its popular library Graylog, Python finds application in log management and analytics.

RQ4: What are the most popular logging topics that software practitioners talk about across the Q&A websites?
This analysis revealed several interesting new and existing logging topics, for example, *logging conversion pattern, android device logging, database logging, Linux logging, logging file creation/deletion/append, and logging level.*

RQ5: Do different Q&A websites (domains) have different logging topics?
There are very few common topics across these websites, such as Android emulator logging and Syslog configuration. The majority of the logging topics are specific to each website. Following is a list of the top topics obtained from each website: DB (*Log shipping, Log file growing/shrinking*), SU (*Event Log, Syslog configuration*), SF (*Log analysis, Syslog configuration*), AE (*App Install, Usage tracking*), SE (*Client server logging, Exception logging*), SO (*Log file creation/deletion, Android Emulator Logging, Logger class of log4j*).

RQ6: How does the interest in logging techniques change over time?
We obtain an increasing trend of logging topics on SO, SU, and DB websites. For example, the topics *framework specific logging problems, log file creation/deletion, log shipping, event log, and android emulator* logging show an increasing trend. In contrast, we obtain a decreasing trend of logging topics on the SF website. For example, the topic *Apache web server logging* shows a decreasing trend.

This study can be beneficial to the community in several ways, as we describe in more detail throughout this paper. First, it shows that different domains or different software practitioners have different logging needs. Hence, future logging studies should put effort into integrating or considering the views of different software practitioners. Second, the research communities, companies, and software developers can use our results to identify troublesome logging areas, tools, API, etc. Companies can learn from our findings to tune their tools, update their documentation, and improve their logging libraries. In addition, companies and corporates can identify logging areas where more training sessions are required to update the current logging skills of software developers with respect to different domains. Third, it shows a path towards new research direction of integrating Q&A websites and other software repositories for identifying major logging issues as well as needs. **The work presented in this paper is a significant extension of our previously published work**[22]. Specifically, the main contributions of this study, along with the difference between this and previously published work, is detailed as follows:
In this work, we perform a three-level analysis of logging questions on six Q&A websites. We answer six research questions by conducting an in-depth empirical analysis of approximately 82K logging questions extracted from these websites. To the best of our knowledge, at present, there is

only one work published on this topic, which is our previous paper *(A Three Dimensional Empirical Study of Logging Questions From Six Popular Q&A Websites*[22]*). We used similar datasets for both of the studies. However, the previous study focused mainly on conducting quantitative or statistical analysis of logging questions, whereas in this study, we primarily focus on semantic analysis of logging questions by analyzing the content (i.e., words and topics) of the logging questions.* In particular, the major additions or unique research contributions made by this study are as follows:

1. Extension: Overall analysis of accepted answers, view counts, and answer counts of logging questions in RQ1 (refer to Section 4.1). In the previous study, we also perform analysis of logging questions across websites; however, our focus was on the yearly trends of these metrics. In contrast, in this paper, our focus is on analyzing the overall values of these metrics with respect to each website, as an overall statistical view is necessary for an overall semantic analysis.

2. Extension: Analysis of most frequent words present in logging questions of different domains and programming languages in RQ2 and RQ3(refer to Section 5.1 and 5.2). In the previous study, we also perform analysis of logging questions across different programming languages; however, we only analyzed the terms present in their tags. In this study, we analyzed the question content as well as the terms present in their tags. Such question content can provide better information about the issues (other than the tags) communicated in the logging questions. The analysis is also extended to gain insight into logging practices across programming languages.

3. New: Use of the Topic Presence (TP) metric for identifying topics in RQ4 and RQ5 (refer to Section 6.1 and 6.2). We identify prominent logging topics on the entire corpora as well as across each website. RQ4 and RQ5 form the core of our semantic analyses.

4. New: Use of the Impact metric for analyzing the trend of logging topics in RQ6 (refer to Section 6.3). After the identification of topics, it is crucial to identify the underline patterns and the evolution of the topics. The analysis of increasing and decreasing trends is essential to understand the overall evolution of logging questions on these websites.

5. New: In-depth manual analysis of the topics obtained from using the Topic Presence and the Impact metrics. The manual analysis includes assigning labels to the unstructured topics by analyzing several logging questions for each topic across these websites.

The remainder of the paper is organized as follows. In Section 2, we describe the closely related studies in the context of our study. In Section 3, we describe the case study setup. In addition, we describe the various research levels and research questions addressed in this work. The motivation, approach, and results of the empirical analysis across three research layers are described in Sections 4–6. In Section 6, we discuss the implications of our work to various software practitioners and the research community. In Section 7, we provide threats to the validity of our findings. Subsequently, we discuss threats to validity in Section 8. Finally, in Section 9, we present the conclusion of the study and discuss future research directions.

2 | RELATED WORK

In this section, we review the closely related work to our research. We divide related works into four lines of research: (1) Empirical studies on logging, (2) Automated Logging, (3) Empirical analysis of Q&A websites, and (4) Uses of LDA topic modeling in software engineering research.

2.1 | Empirical studies on logging

Prior work on logging tried to identify logging issues by conducting (1) surveys of software developers, or (2) empirical analyses of issue reports, source code repository, etc. Currently, there exists no in-depth study that focuses on analyzing the logging issues from the Q&A website. Prior empirical studies analyzed logging with respect to several dimensions[23 5 2 24 25 26 9 27 28]. Yuan et al.[23] analyze open-source software projects and identify types of modifications on which software developers spend most of their time. Shang et al.[24] perform an empirical study of logging modifications and report that debugging, feature change, inaccurate logging level, and redundant logging, are the four major reasons of logging statements modification. Kabinna et al.[26] analyze Java software projects to identify major reasons and after-effects of logging library migration. Li et al.[9] analyze log levels of logging statements to identify the distribution of log levels in various projects. Li et al.[27], in another study, summarize 20 main reasons for changes in logging statements. These reasons belong to four main categories: *changing context code*, *improving logging*, *dependency-driven changes*, and *fixing logging issues*. Yuan et al.[5] identify the most common error types from C\C++ bug reports. Lal et al.[28] analyze logged and non-logged catch-blocks. They report several distinguishing characteristics between logged and non-logged catch-blocks. Li et al.[11] performed a qualitative study that combines a survey of 66 developers and a case study of 223 logging issue reports to identify a comprehensive picture of benefits and costs of logging from developers' perspectives. These studies are useful in identifying several interesting issues about logging. Our work is complementary to these works and can further help in improving the understanding of logging issues.

2.2 | Automated Logging

Several studies focus on automating various aspects of logging. Yuan et al.[5] propose *LogEnhancer* tool to enhance the content of log statements by adding causally-related information to improve failure diagnosis. Lal et al.[6 7 29 30], Fu et al.[2], and Zhu et al.[1] work on predicting logged and non-logged code snippets using machine learning. Li et al.[9] leverages an ordinal regression model to predict the verbosity level of logging statements. These studies are quite interesting and helpful to software developers. **Our work is complementary to these studies**. It can be **beneficial in improving the feature engineering** step of these studies. Additionally, our work can **provide a fresh perspective towards logging automation for other stakeholders** related to software development, such as network administrations, database administrators, mobile application developers, users, etc.

2.3 | Empirical analysis of Q&A websites

Several prior studies show that the data present on Q&A websites can be used to gain information insights related to software development issues. The data present on the Q&A websites has been mined for various purposes in the past. Pinto et al.[13] analyze questions and answers related to energy consumption. They study the main issues that cause software energy consumption related concerns. In addition, the authors analyze potential solutions proposed by the software developers to resolve these issues. Authors in[31] analyze water-related questions posted over environment-related Q&A websites to explore solutions to global water scarcity. Mario et al.[14] mine mobile related questions from SO. They identify the three most common mobile issues: compatibility, crash reports, and database connection. Beyer et al.[32] analyzed android development related issues. They found that software developers commonly face issues related to API uses. Yang et al.[33] analyze questions consisting of security-related text in their title or body. They found that security-related questions belong to five main categories: web security, mobile security, cryptography, software security, and system security. Malik et al.[34] analyze questions related to energy consumption in android devices. They report that inappropriate implementation, sensor, and radio utilization are the main causes of energy consumption issues in android devices. Nagy et al.[35] mine SQL related questions. They analyze some of the error patterns and found that they occurred because of incorrect usage of SQL. In this work, we analyze the logging questions from the Q&A websites to identify logging issues. **Therefore, these studies serve as a motivation for this work.**

2.4 | Uses of LDA topic modeling in software engineering research

LDA[36] is a popular topic modeling technique that is beneficial for discovering popular topics present in a given collection of documents. **In the literature, several studies have applied the LDA technique to infer topics present in various repositories[37 12 38 39]**. Tian et al.[38] used LDA to identify the programming languages used to develop software. They build a corpus of documents using the identifier and comments present in the source code and classify software in different programming languages, such as Java, Python, C, C++. Thomas et al.[37] use LDA for analyzing software evolution. Their study reveals that the topic modeling approach is beneficial in identifying changes in fragments of code. Their approach can discover several interesting topics, e.g., 'affin-transformation' and 'undoable edit'. Pagano et al.[39] analyze blogs of committers and non-committers. They used LDA topic modeling for this analysis. They found a huge difference in the commit behavior of commiters and non-commiters. In addition, they report that committers often write blogs related to source code, whereas non-committers write blogs related to 'conferences', 'events', etc. Barua et al.[12] used LDA on SO posts and identified popular topics discussed among the software developer community. Their results show that 'mobile application' related questions are gaining popularity. They focused on identifying all the popular topics occurring on SO rather than focusing on a specific domain. Prior studies found that LDA topics are easy to understand by software developers[40 41] and are useful in providing high level view of the system[42]. These studies show the successful application of LDA and **serve as a motivation for the work done in this paper.** In this work, we identify high level logging topics present on various Q&A websites. Moreover, some prior logging studies have already applied LDA. Li et al.[43] and Lal et al.[28] use LDA to analyze the source code of popular software projects to identify the relationship between topics present in a code snippet and their probability of getting logged. In contrast to these studies, we emphasize identifying logging topics present on Q&A websites.

3 | EMPIRICAL STUDY SETUP

In this section, we describe the studied system and method followed by this study.

3.1 | Studied System

We perform this study on six popular Q&A websites: Stack Overflow (SO)[16], Serverfault (SF)[17], Superuser(SU)[18], Database Administrators (DB)[19], Software Engineering (SE)[20], and Android Enthusiasts (AE)[21]. All these websites belong to StackExchange network[7]. We selected these websites as they have a large user base and are popular in their respective domains, each having years of history. Additionally, we wanted to explore logging questions posted on websites belonging to different domains. For example, SO is a Q&A website created for professional and enthusiast programmers. SF invites questions on system and network administration. SU website is created for computer enthusiasts and power users. DB is a website for database professionals who wish to improve their database skills and learn from others in the community. SE website is created for professionals, academics, and students working within the systems development life cycle. AE is a Q&A website for enthusiasts and power users of the Android operating system. All these websites are ≈8-10 years old and had ≈0.1-8.2 million users at the time of this study.

3.2 | Dataset extraction

In this subsection, we describe the steps that we used to create our experimental dataset (refer to Figure 1). The data of all the StackExchange websites is publically available under cc-by-sa 4.0 licence [8]. We downloaded the dataset of all the six websites and pre-processed it. We first had to identify all the logging questions from these websites. However, manual identification of logging questions from the pool of all the questions is a non-trivial task. Hence, we designed the following two-step methodology to filter all the logging questions:

Step 1: We extract all the logging questions that consisted of **log** tag. However, we observed that this approach resulted in false positives. For example, questions with tags, such as *login*, *logins*, also appeared during the search. Therefore, we manually removed all such tags from the list of logging tags. Additionally, we noticed that not all the logging questions could be extracted using this procedure. Hence, we moved to step 2 for the extraction of the remaining logging questions.

Step 2: We identify the top 6 programming languages (using TIOBE index[44]) and their commonly used logging libraries. We manually identify all the logging tags related to these logging libraries. Examples of extracted tags are: *SLF4J*, *Winston*, and *Morgan*.

Using the above two steps, we identified 169 tags and extracted all the questions consisting of these tags. Appendix A presents the list of all these tags. **Data and source-code used for this research are publicly available for further research at GitHub repository newtein/EmpiricalAnalysesLogging**.

3.3 | Dataset Statistics

We followed the process mentioned in step 1 and step 2 and extract a total of 169 tags (Appendix A provides the details of the tags). We extract all the questions consisting of any of these tags. In addition to regular questions, our corpus consists of closed and duplicate questions. Inclusion of these questions would provide a wholesome perspective to the analysis. Table 1 presents the details of the dataset extracted. It is evident from Table 1 that a large number of questions are asked on logging (≈82K). In addition, it shows that the percentage of logging questions is much more on DB and SF as compared to other websites.

3.4 | LDA pre-processing steps

Topics across each website are identified with the aid of the LDA algorithm[36]. Researchers have proposed several topic modeling techniques, but we have selected LDA because it is suitable for identifying topics in natural language text documents[12 36]. For each website, the initial corpus consists of documents constructed from logging-related questions. These documents represent the title and description of each question. As shown in Figure 2, the following preprocessing steps were applied to these documents: First, source code snippets from the questions were removed by eliminating content amidst '<code>' and '<code>' tags. Second, all the embedded HTML codes and tags were removed by replacing tags, such as '&lt; to '<', followed by eliminating them using regular expressions. Finally, some miscellaneous tags were identified following a manual inspection, such as '&quot;', &#xA; etc. Subsequently, we removed them along with any special characters. The preprocessed corpus is imported into the MALLET (MAchine Learning for LanguagE Toolkit) by keeping sequence and removing stopwords. Keeping sequence is a MALLET configuration

[7]https://stackexchange.com/sites
[8]https://creativecommons.org/licenses/by-sa/4.0/

that preserves the document as a sequence of word features, rather than a vector of word feature counts. Remove stop words option is responsible for ignoring a standard list of common adverbs, conjunctions, pronouns, and prepositions.

3.4.1 | LDA Hyper-Parameter Tuning

Binkley et al.[45] discuss tuning hyper-parameters of the LDA algorithm in detail, especially in Software Engineering. The authors found out that the number of topics, alpha, and beta should be determined according to the problem. Specifically, the authors study that increasing alpha yields more topics per document. This potentially facilitates straightforward topics with a lesser number of dominant words. After conducting experiments, we choose 50 as the value for alpha. In addition, authors found out that the default value of beta, i.e., 0.01, and larger value of alpha encourage lesser words in the topic and more topics in a document. We choose a reasonably high number of iterations to enable converging of the model. However, this is not much theoretical evidence related to optimize-burn-in and optimize-interval. In particular, the authors of this study explains that existing methods do not support the investigation of the optimize-interval well. Authors show that smaller burn-in is probably acceptable; however, 2K seems to be a conservative choice. After conducting experiments, we found that 1000 and 100 for the choice of optimize-burn-in and optimize-interval, respectively, generate satisfactory results. A brief description of LDA hyper-parameters, along with their configurations are detailed in Table 2. Post-execution of the MALLET, doc-topics matrix is used for calculation of topic presence across each topic. Modeled topics are then sorted in reverse ascending order, followed by labeling the first five explainable topics.

3.4.2 | LDA Topic Label Assignment

LDA results in unstructured topics that require multiple rounds of manual inspection for labeling. Labels are intuitively assigned by two of the authors of this paper by grouping the terms belonging to a similar domain. In addition, these labels are further validated by both the authors by analyzing logging questions in our corpus related to the topics.

3.5 | Research Layers and Research Questions

Table 3 shows three main research levels (RLs) and respective research questions (RQs) considered in this work. Following is a brief description of each RL and respective RQs:

RL 1: Popularity and frequency analyses of logging questions: In the RL 1, we answer questions related to the statistical properties of logging questions. We analyze the number of logging questions with accepted answers & count of answers for each logging question. In addition, we analyze the time taken to resolve logging questions. This RL is helpful in identifying whether logging questions get attention on all the Q&A websites/domains or not.

RL 2: Popular words and tags analyses with respect to different websites (domains) and programming languages: In RL 2, we analyze what the most important terms that occurs in the title, body and tags of logging questions w.r.t each programming language. We perform this analysis domain wise as well as programming language wise. The insight derived from this RL can be beneficial in identifying the important logging concerns of software practitioners with respect to different programming language they are working.

RL 3: Semantic analysis of logging questions: In this RL, we analyze the content of logging questions using LDA topic modeling, followed by identifying prominent topics. We perform three types of analysis in this RL. In RQ 4, we combine logging questions from BAR GRs and identify the popular logging topics using the topic presence metric (refer to section 6.1 for more details). In RQ 5, we identify the top five logging topics from each website with the highest Topic Presence. In RQ 6, we analyze the trend of various logging topics obtained in RQ 5. The insights derived from this RL would be beneficial to software practitioners and the research community in current as well as future logging studies.

4 | POPULARITY AND FREQUENCY ANALYSIS

4.1 | RQ1: What domains are affected by logging? What is the response of the community towards logging questions in each domain?

Motivation: The primary motivation of this RQ, is to identify the domains/websites, which involves the usage of logging. Additionally, we want to identify (1) the kind of response that logging questions receive from the community, and (2) whether logging issues receive views and answers.

These questions would receive views and answers only when other users face similar issues, and the community members are interested in resolving them. To answer this RQ, we analyze the statistical properties of logging questions with respect to three dimensions. First, we analyze the percentage of successful logging questions among the total logging questions. A successful logging question indicates that the logging question has an accepted answer. A high percentage of successful logging questions indicate that the community is interested in resolving these questions. Second, we analyze the number of answers posted for each logging question. A significant number of answers on logging questions indicate that logging questions invite a considerable amount of discussion. Third, we analyze the number of views of logging questions. A large number of views can indicate the interest of users in logging questions.

Approach: To answer this question, we extract all the logging questions, number of answers posted for each logging questions, and number of views that each logging question has received. Next, to analyze successful logging questions, we create one graph consisting of bars for each website, illustrating the percentage of successful logging questions. Subsequently, we create a boxplot to analyze the number of answers received for each logging question. Finally, we create another boxplot to analyze the number of views gathered by each logging question.

Results: The three metrics chosen to describe the statistical properties of the logging questions are the number of successful questions, the number of answers per question, and the number of views accumulated by each question.

1. **Successful Logging Questions:** Figure 3a represents bar chart of successful to total questions. It can be inferred that the ratio of the number of successful logging questions to the total number of questions for AE, DB, SO, SE, SF, and SU are 0.27, 0.49, 0.49, 0.57, 0.51 and 0.42, respectively. The closer this ratio reaches unity; it indicates more successful questions. It is observed from the analysis that nearly half of the questions asked on these websites get an accepted answer with an exception to AE. AE has the lowest successful to total question ratio, i.e., 0.27. This finding has an important meaning for the android community. It implies that approximately 70% of questions asked by users on AE did not receive a satisfactory response. On the other hand, approximately 60% of questions on SE get a satisfactory response. This implies satisfaction and the growing interest of the community in the resolution of software engineering-related queries.

2. **Answers per Question**: Figure 3b represents a boxplot of the number of answers received by each question. The median answers received by each question on AE, DB, SO, SF, and SU are 1, whereas SE has median answers equal to 2. The Inter Quartile Range (IQR) for AE, DB, SO, SE, SF, and SU are 1.5, 1, 1, 2, 1, and 1, respectively. IQR delineates the spread of data between 1st and 3rd Quartile. The maximum number of answers received by AE, DB, SO, SE, SF, and SU are 3, 3, 3, 6, 3, and 3, respectively, along with a plethora of outliers. Results indicate that questions on SE invite a great deal of discussion in comparison with other websites. Additionally, the central tendency of data of logging-related questions suggests that each question receives a median of 1 answer with an exception to SE.

3. **Views per Question:** Figure 3c represents a boxplot of the number of views received by each question. The median views gathered by each question on AE, DB, SO, SE, SF, and SU are 574, 368, 349, 444, 565, and 538, respectively. This shows the interest of the audience (both technical and non-technical) towards the concerned platform regarding logging-related questions. AE, followed by SF, has the most median views, whereas SO has the least. IQR for AE, DB, SO, SE, SF, and SU are 2315.5, 1351.5, 1152, 1273, 1705.5, and 2243.5, respectively. The minimum views received by AE, DB, SO, SE, SF, and SU are 9, 11, 3, 28, 3, and 6, respectively. The maximum views received by AE, DB, SO, SE, SF, and SU are 5629, 3474, 2976, 3332, 4426, and 5697, respectively, along with several outliers. These outliers are removed for the sake of ease in visualization. Surprisingly, the biggest website of all, i.e., SO has the least IQR and least extreme maximum and minimum when compared to other websites. This might infer to a gradually emerging uniformity among the views per question. **An interesting observation from this RQ concerns AE. Although questions on AE receives a broad range of views with extreme maximums and minimums, the community on AE is not completely satisfied with the answers (i.e., no accepted answers) on approximately 70% of the questions.**

Summary of RQ1: The results of RQ1 show that not only logging issues occur on all the six websites or domains (considered in this study), they also receive responses from the community experts. This indicates that logging is an important issue that needs to be considered from various dimensions. The current research on automated logging mostly focuses on providing recommendations to software developers using features from source code or issue reports. Our study provides a fresh perspective towards improving the current logging automation studies by connecting them to Q&A websites. For example, the feature engineering step of the machine learning algorithms in these studies can be tuned for each domain by utilizing the information present on the Q&A websites.

5 | POPULAR WORDS AND TAGS ANALYSIS

5.1 | RQ2: What are the differentiating terms present in the logging questions of each website (domain)?

Motivation: In this RQ, we analyze the content of the logging questions from different websites (domains) by examining the important words/terms present in the title, body, and tags of these logging questions. The insights derived from this RQ can be beneficial in identifying the broad or high-level issues faced by software practitioners from different domains with respect to logging. We analyze both the tags and content of the logging questions, since the analysis of tags provides the high-level problems that the software practitioners face, whereas the terms present in the title & body provide the lower level details of the problems faced by users. For example, there can be many users facing problems with the 'Log4j' (tag) library, but there can be a few developers who are facing issues related to 'imports' in Log4j, whereas others can be facing issues related to 'performance' in Log4j. The terms present in the title & body can help identify these lower-level details about the logging issue.

Approach: We extract the title, body, and tags of these logging questions across six websites. Subsequently, we apply preprocessing techniques, such as stemming and stop word removal, on the question title and body. Notably, we do not apply any preprocessing techniques on the question tags since they were anticipated to be more helpful in their natural form. To this end, we form two corpora, one using the words present in the title & body of the questions, and other, using tags of the question. Finally, we create word clouds from these corpora. Word cloud is a text visualization technique. It takes a corpus as an input and creates a collage of words present in the corpus. The size of words present in the collage is proportional to the frequency of that word in the corpus, i.e., the large size of the word indicates the higher frequency.

Results: As shown in Figure 4 and Table 4, the discussion of the results and insights are as follows:

1. **AE:** Users on AE frequently discuss connectivity-related features of android devices, such as Bluetooth, WiFi, Call, and USB-debugging (See tags: 'bluetooth', 'usb debugging', 'wifi' and 'call'). This can be attributed to the fact that the application of logging becomes crucial when interacting with third-party devices in the environment. In addition, users mention some of the issues pertaining to their devices. As depicted by Fig. 4a and S2a, these include insufficient memory, crashes, malware, system errors, and freeze (See tags: 'insufficient memory', 'crashes', 'malware', 'system error', and 'freeze').

   The most frequently mentioned android version is Jelly bean, followed by Lollipop, Kitkat, and Marshmallow. Some of the offered solutions by the community involves troubleshooting and resource monitoring. Logging plays a key role in this process, and Logcat is the frequently mentioned logging library.

   The number of questions on AE concerns the curiosity of users to view and troubleshoot their problems using log files. Some of these questions are *Problems accessing message logs on Jelly Bean with aLogcat* and *How can I view and examine the Android log?*. Community suggests that the preferred way is to download the SDK and use adb logcat. Additionally, to view full system logs, one requires the root access [9] [10].

2. **DB:** The content of tags and questions as featured in Fig. 4b and S2b focuses on database backup and recovery (See tags: 'backup transaction', 'mysql replication', 'replication binlog', 'disaster recovery', 'mirroring', etc.)

   Logging plays a crucial role in the database backup and recovery process. Key insights in this process are derived through the implementation of one or more types of logging techniques. Apart from generic error logs, this process potentially utilizes transaction logs, server logs, and shipping logs (See tags: 'transaction log', 'server log', 'error log', 'shipping log', etc.). A particular technique to this domain is log shipping. It is the process of automating the backup of transaction log files on a primary database server, and then restoring them onto a standby server (See tags: 'shipping sql', 'shipping log', 'server transaction', etc.).

   At first glance, *'sever r2'* and *'r2 transaction'* seem to be some of the obscure mentions by the users on DB. In contrast, these refer to user's concerns towards SQL Server 2008 R2. It is a major build version of SQL Server (10.50.xxxx). *'sql-server-2008-r2'* tag features more than 3.7K questions on DB. Some of them concern the default location of the log file and the rebuilding of transaction logs [11] [12].

3. **SF:** SF features questions about server, networking, and related infrastructure administration. As anticipated, users on SF mainly discuss the implementation of logging techniques over a network that usually includes servers and databases (See tags: 'server log', 'syslog', 'windows server', 'sql server', 'ubuntu logging', etc.). The operating system is an important part of a server. Notably, users on SF mentions the Windows Server more frequently than Ubuntu Server.

[9] https://android.stackexchange.com/q/27586
[10] https://android.stackexchange.com/q/14430
[11] https://dba.stackexchange.com/q/77510
[12] https://dba.stackexchange.com/q/133559

Notably, Figures 4c and S2c provide frequent mentions of logging in Elasticsearch, apache, ubuntu, Nginx, and postfix (See tags: 'elasticsearch logstash', 'apache logging', 'ubuntu logging' and 'postfix logging'). This implies the popularity of tools, such as Elasticsearch and Nginx, in the server environment. Elasticsearch is a distributed multitenant-capable full-text search engine, whereas Nginx is a web server that can also be used as a reverse proxy, load balancer, mail proxy, and HTTP cache. (See tag: 'web server'). In one of the successful questions on SF, a user asks the community the following question *best way to debug Nginx config file?*. The community provides the user with a very efficient answer that involves turning on *rewrite logs*, followed by setting the debug level in the error log directive [13].

Syslog, i.e., System Logging Protocol, is the standardized logging practice in the network. It is primarily used to collect various device logs from several machines in a central location for monitoring and review. Over the years, several enhancements of Syslog has been introduced. For example: rsyslog and syslog ng (See tags: 'syslog ng', 'syslog', 'ryslog', 'rsyslog linux', etc).

4. **SE:** It is observed from Figures 4d and S2d that users on SE primarily focus on the implementation practices of source-code logging. This includes but not limited to improving logging design patterns, architecture, and performance (See tags: 'design patterns', 'programming practices', 'architecture', 'performance', etc.). Logging is an integral part of Software Engineering; hence, software developers are always curious about state-of-the-art logging programming practices and conventions (See tags: 'documentation', 'conventions', etc.). These ranges from conventional object-oriented style to emerging functional style (See tags: 'object-oriented', 'functional', etc.). Some of the relevant questions include *Design pattern for wrapping logging around execution* and *Rules and advice for logging?*. In both cases, users seek community support for understanding design patterns and conventions for hassle-free and efficient implementation of source code logging [14] [15].

   As anticipated, Java being the most popular language for software development is the most tagged programming language, followed by Python, on SE (See tags: Java and Python). Furthermore, the mention of spring suggests frequent implementation-related concerns regarding logging in web applications (See tags: 'spring' and 'web application'). For example, a user asks *Is it still a good practice to log parameters and returns?*. This user further explains his/her idea of implementing a Java Spring Boot Application and asks the community about relevant logging practices in this regard [16].

   The concept of internationalization of logs is debatable among the SE community (See tag: 'internationalization'. Internationalization (i18n), i.e., an emerging practice among software engineers, is the process of software design so that it can be adapted to various languages and regions without engineering changes). Users on SE are observed to discuss the pros and cons of the internationalization of logs. On the one hand, some users support the idea of logs to be readable by users instead of developers or technicians, whereas others contradict it by pointing out that most of the users are not active maintainers of the software. Thus, it seems futile for them to understand the various implementation details of the software [17].

5. **SO:** SO is a specially oriented website of Stack Exchange network that hosts a multitude of concerns of programmers coming from a wide variety of domains. Analysis of word-cloud from Figures 4e and S2e provided us a deeper insight into logging practices. Logging practices in Java are observed to be most mentioned by the users, followed by Python (See tags: 'java logging' and 'python logging'). In addition, Apache Log4J, i.e., a widely popular logging library in Java, is frequently discussed.

   Logstash was released in February 2016 (word-cloud illustrates the data until 2017). Logstash is mentioned with two popular frameworks, such as Elasticsearch and Kibana (See tags: 'elasticsearch logstash', 'logstash kibana', and 'logstash grok'). On the one hand, visibility on SO can be beneficial for the developers; on the other hand, it indicates a large volume of user's concerns. Elasticsearch, Logstash, and Kibana (ELK) Stack is widely popular. Grok filter of Logstash makes it a unique logging module. It is used to parse unstructured data into something structured and queryable. It sits on top of regular expression and uses text patterns to match lines in log files.

   Microsoft's Enterprise Library provides APIs to facilitate proven practices in core areas of programming, including data access, logging, exception handling, and others (See tags: 'enterprise library'). This library is no longer under development. We have observed the dilemma of multiple users over the continued use of this library. Some of them claim it to be an extremely over-engineered solution for logging. Few instances of such questions are as follows: *Are you using the Microsoft Enterprise Library?*, *Is Enterprise Library still being updated?* and *Continue with Microsoft Enterprise Library?* [18] [19] [20].

[13]https://serverfault.com/q/333048
[14]https://softwareengineering.stackexchange.com/q/298714
[15]https://softwareengineering.stackexchange.com/q/165650
[16]https://softwareengineering.stackexchange.com/q/286641
[17]https://softwareengineering.stackexchange.com/q/316539
[18]https://stackoverflow.com/q/990669
[19]https://stackoverflow.com/q/46736539
[20]https://stackoverflow.com/q/1069892

6. **SU:** Most questions on SU concerns Event Viewer - a native software to the Windows operating system. It extends the functionality of viewing event logs to administrators and users on a local or remote machine (See tags: 'event viewer', 'windows event', 'event log', etc.).

   Figures 4f and S2f frequently mention event logging. Event logging is a term commonly used in Windows environment. It is a comprehensive record of system, security, and notifications stored by the Event Viewer. It claims to not only diagnose system problems but also predict future issues. These logs can be of high volume; thus, filtering relevant records is essential. A user on SU asks *How can I use Event Viewer to confirm login times filtered by User?*. To rephrase, this user wants to view the logs depicting information regarding his/her 'physical' logins. The community delves into technical details and endows the user with a variety of event nomenclature to filter relevant records.

   Similar to SF, the SU also records queries related to implementing logging in a network and servers (See tags: 'networking', 'windows server', 'apache server', etc.). SU is observed to be more focused on Windows event logging, whereas SF is focused on varied logging frameworks.

RQ 2 Summary: Each website provides us with a different perspective towards the development in the source-code logging. All of these websites consist of a wide variety of knowledge regarding logging that would guide users towards the respective platform. Broadly, users on AE, DB, SF, SO, and SU emphasize the discussion related to the implementation of source-code logging in a specific context, whereas users on SE discuss the development of logging. Following is the summarization of the content featured at each of these websites. Android developers tend to log for resolution of issues like crashes, malware, and system errors. Database administrators employ a very specific shipping-log technique to ensure prompt database backup and recovery. Syslog and its frameworks play a crucial role in the centralized processing of log files in a server environment. Software engineers on SE work on the development of state-of-the-art logging techniques, which improve programming practices and improving performance. SO emphasizes a variety of user queries related to Java, Python, and ELK Stack. SU focuses upon event logs that are particularly designed for troubleshooting issues in the Windows operating system. Notably, the difference in the logging practices of different domains suggest that future studies should address the specific logging needs of different practitioners.

5.2 | RQ3: What are the differentiating terms present in the logging questions of each programming languages?

Motivation: In this RQ, we analyze the important terms present in the title, body, and tags logging questions belonging to each programming language. We believe that the insights derived from this RQ help identify the important logging issue faced by software developers across each programming language. We use a similar high-level and low-level approach as used in RQ2. We focus on identifying popular words that are present in logging questions of each programming language.

Approach: To answer this RQ, we select the top 6 programming languages. The selection criteria of these six programming languages are mentioned in Section 3.2. We select logging questions belonging to these programming languages in two steps. First, we identify popular logging libraries belonging to these six categories (refer Table5). We assign the question to the programming language to which the library belongs. Second, we extract all the questions that consists of any of the programming 'language tag' (for example, C', 'C++', 'C#', 'Java', 'Python', 'JavaScript') and any 'logging tag'. In this case, the question is assigned to the programming language whose tag is present. Next, we extract the title-body and tags of these logging questions, which serve as our two corpora, respectively. The remainder of the steps is similar to RQ2.

Results: The content of each question has been examined with the aid of high-level and low-level approaches, as shown in Figure 5 and Table 6. The results corresponding to each programming language are delineated below.

1. **C:** It is observed that network and system logging is chiefly implemented through C language. Figures 5a and S3a mention Syslog and its variants, such as rsyslog, syslog4j, syslogd, and syslog-ng.

   Syslog stands for *"System Logging Protocol"*, it is the default logging mechanism for devices like routers, switches, firewalls, wifi access points, Unix/Linux Servers. It is a standard under RFC 5424 for sending and receiving notifications.

   In addition, Figure 5a features Postfix. Postfix, a mail transfer agent, is an engine for email servers on Linux, and logging is crucial to it. A user on SO asks the community to alter the log collection target of Postfix from the default syslog to another container. Although an unsuccessful question, the community advises the user to try using rsyslog. Rsyslog is an application that is a syslog daemon. Gradually, it has developed into a general-purpose logging tool that can read, parse and buffer logs coupled with the utility to send them to multiple locations [21].

2. **C++:** As illustrated in Fig. 5b, tags of C++ related logging questions can be categorized into user's issues related to logging related libraries and their implementation in various environments.

   Log4cpp, Log4cplus, boost-log, glog, log4cxx, and Patheios are some of the most frequent C++ logging libraries mentioned. Some of the notable implementations are, Apache log4cxx and Glog. Apache log4cxx is a logging framework specially designed for C++ using a pattern similar to popular Apache log4j; thus, we can find mention of log4j as well. Glog is the C++ implementation of Google logging modules.

   In addition to the mention of logging libraries, we have observed tags such as macros logging, Qt logging, multi-threading logging, and Windows API (winapi). These suggest the application of logging in various environments of C++. For example, Qt is the C++ widget toolkit for creating graphical user interfaces; Windows API i.e., written in C++, is the core set of application programming interfaces available in the Microsoft Windows operating systems. In addition, these questions mention the usage of inherent C++ based features, e.g., operator overloading.

   Figure S3b suggests that users are particularly finding issue with boost log. Boost is a set of libraries for the C++ language that provides support for a multitude of tasks, including logging. Users are finding difficulty in linking their application with a boost log, especially in a multi-threaded environment. Related error words are 'mt posix', 'mt v2s', and 'mt nt5' (Figure S3b). Table 7 shows some of the issues that faced by software practitioners while linking boost log library[22].

   All the questions were asked on the SO website and received hundreds of views. All the users faced these issues on the Linux machine. The thing to note here is that for windows user there is tool name **Boost.log Compile Error (Repair Tool)** [23] that helps users in resolving boost log linking errors but there is no such tool for the Linux users (to the best of our knowledge). The research community can use this insight and can plan to develop such repair tools for Linux users as well.

   Furthermore, there is frequent mention of log4cplus appender (log4cplus::Appender). Appender is the class of log4plus that can be extended for implementing customized strategies for printing log statements [24].

3. **C#:** Users working with C# primarily raise issues that concern with log4net in addition to NLog, as illustrated in Fig. 5c. Apache Log4net is a widely used logging library in the .Net environment, whereas NLog is an open-source alternative to Log4Net. Figure S3c illustrates that users have issues majorly with three aspects of log4net: its overall configuration, appender class, and integration with ASP .NET MVC (Model View Controller) framework.

   Users find the configuration in Log4net especially difficult because it can only be configured through a config file (e.g., app.config file) or a dedicated log4net.config (See Tags: 'log4net config', 'conflig file', etc). In the example below, a user asks on Stack Overflow: *Can you configure log4net in code instead of using a config file?*. This is a successful question that has received multiple helpful answers from the community. In addition to that, NLog can be configured through both XML and C#. This makes configuration through NLog easier [25].

   Apache Logging Services categorized Apache Lag4Net as a dormant project from 1st April 2020 onwards. Dormant status implies complete inactivity in the development of the project. This step is expected to increase the interest of the community towards NLog.

4. **Java:** As depicted in Fig. 5d, Java-related logging questions frequently concern three logging tools along with their implementations and configurations. These tools are Java Util Logging (java.util.logging), Log4j, and Logback. First, Java Util Logging provides the classes and interfaces for core logging facilities. Second, Apache Log4j is a popular logging package written in Java, and it is widely used. Third, Logback has been a recent one that claims to be a replacement for its predecessor i.e., Log4j.

   It can be observed from Fig. 5d that users mention SLF4J along with Log4j and Logback. It can be attributed to the fact that the Simple Logging Facade for Java or SLF4J serves as a simple facade or abstraction for various logging frameworks, including java.util.logging, logback, log4j. Users also mention 'maven log4j' that suggests difficulty in the installation/configuration of log4j. Maven is a build automation tool for Java projects and helps in downloading packages and their dependencies.

   Multiple issues regarding Log4j can be identified from Fig. S3d. These queries are regarding various classes of Log4j namely appender, logger, core, patternlayout (aids in customizing output according to a pattern) and properties (log4j configuration file) [26].

[21] https://stackoverflow.com/q/32118734
[22] https://stackoverflow.com/q/23137637
[23] http://winbytes.org/help/boost-log/boost-log-compile-error.html
[24] https://stackoverflow.com/q/12464866
[25] https://stackoverflow.com/q/16336917
[26] https://stackoverflow.com/q/51624211

Furthermore, it appears that users are finding issues with the compilation of java environment, especially while logging. This is concluded from the usage of words, such as 'jar compile' and 'Catalina' (Catalina is Tomcat's servlet container.) in Fig. S3d.

5. **JavaScript:** Figures 5e and S3e find multiple mentions of Winston i.e., a JavaScript logging library (See tags: 'winston javascript', 'express winston', 'winston node', 'logging winston', etc.). Winston provides a complete logging solution to JavaScript as well as its frameworks, such as Express JS, Angular JS, and Node JS. Additionally, it provides users to access inbuilt loggly transport (See tags: 'winston loggly'). SolarWinds Loggly is a cloud-based log management and analytics service provider. Thus, it provides users with logging as well as for analytics utilities. Figure 5e also features syntax of Winston use to implement logging, for e.g., 'new winston', 'require winston', 'winston logger', 'winston transport', etc.

   Additionally, Bunyan is another logging framework (See Tags: 'Bunyan javascript'). However, its working is moderately different from Winston, making its priority to provide quite structured logs. As a result, a log record from Bunyan is one line of JSON.stringify output.

6. **Python:** Figure 5f illustrates the high-level content of logging questions related to Python that largely discusses three things. First, Web engine based logging related to Django and Flask. Second, analysis of log files using Python with the help of log management and analytics tools such as Graylog and Graylog2. Third, the implementation of logging in multithreaded and multiprocessing applications. Additionally, topics with notable frequency include open source and popular logging libraries/utilities, such as syslog, rsyslog, and Logstash, along with logging at google cloud and elastic search.

   From the low-level perspective shown in Fig. S3f, the logging questions largely comprises of Python-based implementation details (e.g., init.py, try, except, etc.). They mention logging files, their configuration, and formatting along with issues in importing libraries. Moreover, it mentions getLogger that is the method to instantiate native logging library of python (e.g., logging.getLogger(name)).

RQ 3 Summary: We observed that each programming language has its niche in the logging environment, with obvious overlaps. In the context of logging, logging questions of C is primarily about networking and server environments, whereas logging questions of C++ finds a wide range of mentions in a multithreaded and multiprocessing environment. C# is predominantly used to implement logging in a .Net environment (especially in ASP .NET MVC frameworks). Java provides a wide range of logging libraries for Java-based projects and software. JavaScript is used for logging Web Applications with efficient logging libraries like Winston that provide not only logging framework but also analytics framework (using loggly). With its popular library graylog, Python finds application in log management and analytics. While existing logging studies use choose subject projects of a single programming language (e.g., C/C++[23], Java[25,43]), the difference in logging across programming languages should be paid attention to in future studies. In addition, to entirely understand the context, exploring the semantic analysis based on programming languages remains the future work.

6 | SEMANTIC ANALYSIS

6.1 | RQ 4: What are the most popular logging topics that software practitioners are talking about across the Q&A websites?

Motivation : In this RQ, we aim to identify the top ten logging concerns of software practitioners across the six websites. This can be beneficial to software practitioners in multiple ways. First, website moderators can report these common logging concerns to experts or companies. Subsequently, these experts or companies can utilize this knowledge and develop new technologies to address them. Second, the unresolved logging questions can be forwarded to logging experts on other websites.

Approach: We use LDA topic modeling technique to answer this RQ. We considered the title and description of a post for topic modeling. We combined the logging questions from the six websites and created a single corpus, followed by executing LDA on this corpus. LDA takes the number of topics, k, as input. However, there is no best value known for k that is suitable for all datasets. Thus, we use the number of topics (k) between 100 to 500, depending on the size of the website. Let there be Z_1, Z_2, ..., Z_k, k topics in the system. The membership value of the topic (Z_k) in the document (d_i) is defined as $\theta(d_i, Z_k)$. Notably, a document can have multiple topics with different membership values. For this analysis, we set a threshold (δ) as the membership cutoff, which is equal to 0.10. Further, we compute *Topic Presence (TP)*1, which yields the proportion of the posts consisting of a given topic. For example, if metric TP is equal to 0.05 for topic Z_k, it implies that 5% of all the posts consist of topic Z_k. Next, we compute the topic presence for all of the topics and report the top ten topics with the highest topic presence.

$$Topic\ Presence\ (Z_k) = \frac{1}{|N|} \sum_{d_i \in N} P_{ik} \tag{1}$$

In Equation 1, N denotes the set of all the logging posts in the corpus. Notably, the title and description of a question are combined and then considered as a document. The symbol d_i represents the ith document, and Z_k represents the kth topic. P_{ik} identifies whether topic k is present in document d_i. P_{ik} is defined in Equation 2.

$$P_{ik} = \begin{cases} \theta(d_i, Z_k), & if\ \theta(d_i, Z_k) \geq \delta \\ 0, \quad Otherwise \end{cases} \tag{2}$$

Results: As shown in Table 8, the insights obtained from the analysis are as follows:

Topic 1: Logging Applications: The appearance of this topic can be attributed to the fact that the software developers often ask questions related to the applications of logging (e.g., log processing applications). Table 9 delineates a snippet of such a question from the SF-194037[27] website. In this question, a software practitioner is looking for good visualization techniques in the Unix system to analyze large log files.

Topic 2: Logging Conversion Pattern: Logging libraries like Log4j have three main parts: *Logger*, *Appender*, and *Layout*[46]. In particular, *Logger* is used for logging the messages, and *Appender* is used for logging print the messages to destinations, such as console, file, etc. *Layout* specifies the format of logged information. Notably, the *conversion pattern* specifies the formatting of log messages using literals, conversion characters, and format modifiers. To this end, this topic shows the concern of software practitioners with respect to printing, formatting, and setting of log messages. For example, Table 9 shows a snippet of such a question from the SO-11245993[28] website where a user is facing issues with the conversion pattern output.

Topic 3: Automated Logging: Topic 3 consists of some generic words, namely 'simple', 'capture', 'set' 'automatically', and 'standard'. These words are not specific; thus, it is not clear to which particular topic they belong. Analysis of some questions under this topic indicates that users have asked questions about automated/built-in technologies for logging. Example 3 (SF-193100[29]) in Table 9 shows a question snippet belonging to this topic, where a user has asked a question about the *built-in auditing/debug/logging feature in Windows.*

Topic 4: Android Device Logging: The fourth topic is related to the logging of Android devices. It is interesting to note that queries about **Galaxy/Samsung** devices are prominent in this topic. In addition, the words, such as **Eclipse** and **emulator**, are highlighted. These words are frequently used by software practitioners for developing mobile applications. Table 9 shows an interesting example (AE-142694[30]), where a user is interested in knowing why his mobile phone is making a strange sound. This confirms that logging is not necessarily used by the software practitioners involved in application development, but it can also be used by regular users for tracing errors in devices, which further highlights the importance of logging.

Topic 5 (Log Files: Creation/Append) & Topic 6: (Log Files: Multiple Log Files) : We notice that topics 5 and 6 are related to the log file creation. These topics are similar because several terms are the same in both topics, such as 'file', 'files', 'create', 'write', and 'written'. Nevertheless, in-depth analysis from questions and terms belonging to these topics revealed two interesting insights. First, the words belonging to topics 5 and 6 are similar; however, these words are unique. For example, topic 5 consists of unique terms: *append*, *appended*, *overwrite*, *empty*, *etc.*. Similarly, topic 6 consists of unique terms: *multiple*, *generate*, *specific*, *location*, *etc.*. Second, we notice that the top questions belonging to topics 5 and 6 are from

[27]https://serverfault.com/questions/194037/good-data-visual-package-for-unix
[28]https://stackoverflow.com/questions/11245993/grails-log4j-conversion-pattern-throwable
[29]https://serverfault.com/questions/193100/log-ldap-access-of-the-active-directory
[30]https://android.stackexchange.com/questions/142694/strange-sound-from-an-unknown-application

the SO and SF websites, respectively. Specifically, when we consider the top 20 questions from each topic, then in topic 5, 85% of questions are from SO, whereas in topic 6, 70% of questions are from SF. Analysis of some questions in topic 5 shows that questions in this topic are mainly related to log append issues. Example 5 Table 9 shows sample questions (SO-10738953[31]), where a user is facing a problem with the log file that is printing data multiple times. In most questions belonging to topic 6, the error was related to multiple log files. For example, finding all the log files with specific patterns and issues about the time zone in multiple log files. Example 6 (SF-192134[32]) in Table 9 depicts sample questions in which the user is interested in finding all the log files that are bigger than 100 Mb.

Topic 7: Searching Logging Solution: Topic 7 consists of the words, such as 'google', 'search', 'stackoverflow', 'link', and 'documentation', indicating that the user is asking for help related to their logging issues. This topic is not about any specific logging issue; however, it shows general challenges of users about logging. In Table 9, example 7 shows such a question (SO-10437544[33]), where the user is requesting for some examples about 'Java remote logging through HTTPS'. In essence, this topic sheds light on the motivation of this work and indicates that users face several challenges in logging. Hence, there is a pressing need to improve the current logging tools/technologies/practices.

Topic 8:Logging in Database Topic 8 consists of the words, such as 'database','transaction', 'sql', 'dbcc', 'ldf', and 'recovery', which indicates logging issues about databases. Logging is quite useful in database applications for maintaining the transaction and recovery. In Table 9, example 8 (DB-119694[34]) shows such a question where a user is asking about the functioning of the SQL server when the transaction log of the database is full. In addition, this topic highlights some interesting terms like '**DBCC**' and **'LDF'**. DBCC denotes DBCC statements that act as Database Console Commands for the SQL Server[35] and are used to check the consistency of the SQL server database. LDF is responsible for holding the transaction log of the database for backup purposes.

Topic 9: Linux Logging Topic 9 shows the words, such as 'syslog', 'server', 'messages', 'rsyslog', 'syslogd', and 'UDP'. Syslog is a protocol that provides a way for network devices to communicate with the server using the UDP protocol, and Rsyslog is an extension of syslog and offers new features, such as RELP and buffered operation. Syslog is the most common logger for Linux and Unix systems. The syslogd daemon handles messages from the server and programs while providing a unified way of handling log messages. This topic is clearly about logging in the Linux and Unix machines. Example 9 (SF-278369[36]) in Table 9 provides an example where the user is interested in sending the UNIX log messages to the Windows event viewer. This example provides an interesting direction to companies working in the logging sector to extend their current technology and provide logging techniques for sharing log messages across different platforms.

Topic 10: Logging Levels : Topic 10 consists of the words, such as 'level', 'debug', 'error', 'warn', and 'fatal'. This topic is clearly indicating questions related to log levels. Our analysis of logging questions depicts that software practitioners often get confused with different logging levels. Recently, some researchers have proposed using machine learning approaches to automatically identify the correct log level in the source code[9]. Software companies may consider improving the current logging technologies by using these machine learning approaches. In addition, we identified certain logging questions where the software practitioners had asked questions related to the logging level merely out of curiosity. Example 10 (SF-28268[37]) in Table 9 shows such an interesting example where a user is asking about the history of 'severity' logging level.

RQ 4 Summary: The topic presence metric reveals several interesting existing and new logging topics. For instance, the software engineering research community is already working on *automated logging* and *logging level* issues, which is revealed by this analysis. In addition, this analysis revealed several new logging topics; for example, *android logging*, *database logging*, *syslog logging*, and *logging file creation/deletion/append*. The results of this RQ can be used by the software engineering research community to identify the potential areas of logging research. These topics belong to different domains, and hence, the results of this RQ have further motivated us to analyze logging questions on each website separately (see RQ 5).

[31] https://stackoverflow.com/questions/10738953/log-data-printing-multiple-times
[32] https://serverfault.com/questions/192134/bat-files-to-find-all-files-on-drive-bigger-than-x
[33] https://stackoverflow.com/questions/10437544/java-remote-logging-through-https-443
[34] https://dba.stackexchange.com/questions/119694/would-sql-in-a-cluster-failover-when-transaction-log-for-database-is-full
[35] https://docs.microsoft.com/en-us/sql/t-sql/database-console-commands/dbcc-transact-sql?view=sql-server-ver15
[36] https://serverfault.com/questions/278369/forward-unix-syslog-to-windows-event-viewer
[37] https://serverfault.com/questions/28268/how-old-is-the-severity-paradigm-in-logging

6.2 | RQ 5: Do different Q&A websites have different logging topics?

Motivation: In this RQ, we work on identifying logging topics across each of the six Q&A websites separately. A combined analysis of logging topics across different websites in RQ 4 has provided several interesting and novel insights. However, we notice that the SO website has substantial logging questions. Hence, we anticipate that the topics obtained in RQ 4 may have a bias towards the SO website. Determining logging topics individually for each website will help us in eliminating this bias. Additionally, there is a possibility that each website has its own set of logging topics, which would indicate specific logging issues faced by users of that particular website/domain. Notably, finding out these specific logging topics can be beneficial for the software practitioners interested in a particular website/domain.

Approach: We used the LDA topic modeling technique to answer this RQ. We considered the title and description of a question for topic modeling. We created six corpora, i.e., one corpus for each website, and ran LDA separately for each corpus. As there is no best value known for the number of topics (k), we vary k from 10 to 500 and select the value of k, which yields the best results. We select the best value of k by performing a manual analysis of the obtained results. For different websites, we obtained different values of k because these websites have a different number of logging questions. As discussed in Section 6.1, a document can have multiple topics with different membership values. Similar to RQ 4, we set a threshold (δ) as the membership cutoff equal to 0.10 and compute *Topic Presence (TP)*, as discussed in Section 6.1 in Equation 1.

Results: Table 10 presents a list of five topics with the highest topic presence across each website. We infer several interesting observations from this table. First, most logging topics are unique to each website, and there are only a few common logging topics among different websites. Second, a simple word cloud-based approach used in RQ 2 was useful; however, it could not entirely determine the logging topics for each website. The LDA-based approach is much more effective in identifying these logging topics. Thus, we discuss the results of this RQ in detail:

Common topics among different websites: We find some common topics among different websites. Firstly, SO and AE websites have a common topic on '**Android emulator logging'**. Android emulators are heavily used by software developers for programming in the Android operating system. To reiterate, SO is well-known for programming-related questions. Hence, we observe these topics on both SO and AE websites. Secondly, SU and SF both feature topics about **syslog configuration**. We already anticipated the presence of this topic in these websites after our word cloud analysis. To this end, we analyzed different tags associated with syslog on both SU (SU-Tags[38]) and SF (SF-Tags[39]) websites. We observed that SU has only syslog and syslogd tags, whereas SF has syslog, rsyslog, syslong-ng, syslogd, newsyslog, and rsyslogd tags. Finally, SO and DB have a topic related to **log file creation/deletion**. Log files are used in various applications, and hence, this topic is appearing on both websites. However, a manual analysis of questions under this topic shows that the SO website features questions on general log file creation/deletion under this topic, whereas DB consists of logging questions specific to files created by various database servers, e.g., PostgreSQL. For instance, in question SO-4345952[40], a user is asking about the empty log file creation. In contrast, in question DB-102385[41], a user is not able to create a log file because of the permission issues in PostgreSQL. In essence, this manual analysis indicates that although this topic appears to be the same on SO and DB, it covers a relatively different set of questions on these websites. Hence, a more in-depth investigation of this topic is required.

Unique logging topics on each website: We observed several unique and interesting logging topics on each website. In the AE website, we find two very interesting topics: **App Install** and **Usage tracking**. The *App install* covers the questions where users are searching for logs to identify issues after installing some new applications on the phone. In question AE-89644 [42], a user is asking about the *list of purchased apps and app install/uninstall history log anywhere*. Conversely, the topic *Usage tracking* delineates the requirement of the user to monitor logs related to various apps present in the phone. For example, when the app was started?, how much memory the app is consuming? etc. In addition, AE-26080 [43] features an example where the user is asking about log accessing from when the phone started. The Android operating system/app developers can use this information and brainstorm about providing easy ways to users to see the logs related to various apps. Also, on the DB website, we found two interesting topics: **Log shipping** and **Log file growing/shrinking**. The *log shipping* is the process of automating the backup of transaction log files on a primary (production) database server, and then restoring them onto a standby server. The topic *log file growing/shrinking* outlines the problem of large log files. The analysis of the questions shows that database administrators face issues with the growing size of log files. For

[38]https://superuser.com/tags
[39]https://serverfault.com/tags
[40]https://stackoverflow.com/questions/4345952/textwritertracelistener-does-not-work
[41]https://dba.stackexchange.com/questions/102385/could-not-open-log-file-var-log-postgresql-2015-05-24-181212-log-permission
[42]https://android.stackexchange.com/questions/89644/list-of-purchased-apps-and-app-install-uninstall-history-log-anywhere
[43]https://android.stackexchange.com/questions/26080/does-android-keep-a-log-of-when-it-starts

example, in question DB-11565 [44], a user is asking about the method to reduce the size of his 5 GB log file:'*Some of my SQL Server's log files (ldf) are huge, like 5GB. I Found an article about shrinking, but am not sure why the author repeated DBCCSHRINKFILE 6 time*.' To this end, the researcher working in this direction should try to facilitate a solution for reducing the gigantic size of log files.

The SE website depicts topics related to **exception logging** & **Client server logging**. The top words in the *exception logging* topic are the *exception*, *catch*, *exceptions*, *error*, *and throw*. This topic covers questions about logging exceptions in the code. A program can throw several exceptions at the run time. Here, users are concerned about when and which exceptions to log. For example, in question SE-173401 [45], a user is asking about best practices for the *exception handling frequency/log detail*. In some studies, researchers have tried to automate exception logging through proactive logging[5] or machine learning[6 1]. The occurrence of this topic on the SE website further motivates researchers to examine the issue of exception logging in detail. In addition, the topic *Client server logging* shows the questions about sending and retrieving logs from the server/client. The SF website features an interesting topic about *log analysis*. This topic consists of the words, such as *tool*, *analysis*, *analyzer*, *analyze*, *awstats*, *view*, *and statistics*. This topic highlights the need for the *network administrators* for analyzing the collected log. Notably, it is essential to provide reasonable methods/tools so that it will become easier for the network administrators to search and analyze logs. For example, in question SF-43825 [46], a user is requesting recommendations for a good W3C log analyzer. Also, the SU website shows a topic on *event logging*. This topic is particularly related to the *event log* feature in the Windows servers. System event logs provide a detailed log of the system, and these logs are used by administrators to diagnose the system. For example, in question SU-315836[47], a user is asking about how to log startup and shutdown events.

The SO website shows informative topics; however, most topics obtained from the SO website are largely covered in RQ4. As discussed earlier, this may be attributed to the SO website having the highest logging questions. Thus, topics obtained in RQ 4 had some bias towards the SO website. We notice that the SO website shows the topics *Logger class of log4j*, which indicates that the library Log4J is quite popular among the developers. Hence, issues related to this library should be given priority as compared to other logging libraries. Table 8 shows several other interesting and novel topics obtained for each website.

RQ 5 Summary: There exist very few common topics among these websites, such as the topic *Android emulator logging* on SO and AE, and the topic *syslog configuration* on SF and SU. However, the majority of the logging topics are specific to each website. For example, DB has *log file growing/shrinking*, SF has *log analysis*, SE has *exception logging*, and SU has *event log* topics. These results motivate further research on logging on different domains to identify the specific logging issues and improve their current tools and techniques.

6.3 | RQ 6: How does the interest in logging techniques has changed over the period of time?

Motivation: In this RQ, we aim to determine the changes in the interest in logging topics over time. Identifying these changes is beneficial for software practitioners in current as well as future software projects. First, it would help identify the logging libraries and tools that gather the most interest, as well as recent advancements in logging technologies. In addition, software practitioners would be able to identify the course of innovation and potential areas to emphasize. Second, it would help examine the lagers and blockers responsible for decreasing the interest of programmers.

Approach: To answer this RQ, we hypothesized that the logging trends could be identified using the posts of developers. We use the *Impact* metric for determining the temporal trend of logging topics. *Impact* metric is previously defined by Barua et al.[12]. Equation 3 shows the formula for computing the *Impact* metric. In Equation 3, N(m) represents the count of all the posts in month m. Barua et al. used the *Impact* metric for identifying changes in trends of various tools and libraries on the SO website. In contrast, we use the *Impact* metric for identifying changes in the logging interest on six Q&A websites. In this RQ, we compute the impact of logging topics obtained in RQ5 for each Q&A website.

$$Impact\ (Z_k, m) = \frac{1}{|N(m)|} \sum_{d_i \in N(m)} \theta(d_i, Z_k) \tag{3}$$

Results: Figure 6 illustrates the trend of topics obtained in RQ 5 for each website. A detailed analysis of the selected topics are as follows:

We obtained an increasing trend of logging topics for the SO, SU, and DB websites. Figure 6 shows several logging topics on SO, SU, and DB

[44] https://dba.stackexchange.com/questions/11565/shrinking-log-file
[45] https://softwareengineering.stackexchange.com/questions/173401/exception-handling-frequency-log-detail
[46] https://serverfault.com/questions/43825/w3c-log-analyzer
[47] https://superuser.com/questions/315836/windows-xp-is-it-possible-to-log-startup-and-shutdown-times-in-the-system-event

with an increasing trend. On SO, we observe that **topic 63 (Framework specific logging problems)** is rapidly increasing. Topic 63 is a generic topic, where software practitioners are asking for some general solutions for logging. This topic further motivates and supports the hypothesis that it is crucial to identify logging issues from Q&A websites for a comprehensive enhancement of logging tools and libraries. Another logging topic on SO is **topic 324 (Log file creation/deletion)**, which shows a constantly increasing trend. As the name suggests, this topic is related to log file creation/deletion/modification. An increasing trend for this topic indicates that developers are facing some issues with the log file. Hence, companies developing logging tools should make the process of log file creation/deletion/modification straightforward for software developers. In addition, the **Log shipping topic (topic 432)** in DB shows a rapidly increasing trend. Log shipping is an old problem, and several studies aimed to address this problem[47 48 49]. However, there is currently no study (at the time of writing this paper) that works on mining databases related to logging issues from Q&A websites. The increasing trend of logging issues shows that database administrators are still facing significant issues, and further research is required to address their problems. In the SU website, we notice that the **Event Log (topic 425)** is related to event logging in Windows. The Windows event logging system records information related to the system, applications, and security. This logged information consists of vital hardware and software instructions that help the system administrators to diagnose the system. Our analysis of some questions in this category shows that the users have asked questions about the display of logs as per the task, views in Windows, problems related to start and shutdown time of the logged event.

Interestingly, we also notice a decreasing trend of logging topics on the SF website after 2011. For instance, the trend of the topic **Apache web server logging (topic 346)** is decreasing on the SF website. A manual analysis of the internet reveals that the Apache webserver has the highest market share until 2011-2012. However, subsequently, its market share has started declining because of the arrival of new webservers in the market, such as Microsoft, SUN, Nginx, and Google[48]. A study by STH in 2017 shows that the *market share of Apache has declined by 50%* [49]. This can be one of the reasons for the decline in the logging questions on this topic. However, a more detailed study is required before reaching any conclusion.

We did not observe any particular trend on the SE and AE websites. On AE, we observe a mix of an increasing and decreasing trend of logging topics. For example, Figure 6a shows an increasing trend for the topic **Android Emulator Logging (topic 13)** on AE. In addition, we found an increasing trend of the same topic, i.e., **Android Emulator Logging (topic 80)** on the SO website. Logging on Android is a critical concern from the perspective of mobile app developers. Chowdhuray et al.[50] work on analyzing the impact of logging on the energy consumption of the mobile phone. Zheng et al.[51] performed an analysis of 1444 Android apps. Their results confirm that there exist differences in the logging practices of desktop applications and mobile apps. A study by Barua et al.[12] on the SO website reveals that the trend of mobile technology is increasing. All these studies indicate that there is a need to promptly address the logging issues of Android. As discussed in previous RQs, logging issues on the Android operating system are less explored and require additional studies. On the SE website, we do not observe any particular trend of logging topics.

RQ 6 Summary: We obtain an increasing trend of logging topics on SO, SU, and DB websites, whereas a decreasing trend is observed on the SF website. The increasing trend of some topics, such as *framework specific logging problems, Log file creation/deletion, log shipping, event log, android emulator logging*, etc., indicates that these topics require an in-depth analysis of the specific logging problems that software practitioners are facing. Similarly, the decreasing trend of some topics like *Apache web server logging* indicates that it would be beneficial if researchers and developers focus on competitive technologies rather than focusing on declining topics.

7 | DISCUSSION

In this section, we discuss some of the implications and benefits of the results to various software practitioners and software engineering research community.

Logging is used by different software practitioners with different logging needs: Source code logging is a vast discipline comprising several methods and their applications in domains ranging from web development, file handling, software design, and databases to android development. Hence, it has become essential for researchers and developers to understand the needs of different software practitioners and modify the course of research and development accordingly. Prior work on logging mostly considered logging from the software developer's perspective and worked on providing recommendations that can aid software developers in logging[6 7 9]. Our analysis of logging questions from six different Q&A websites

[48]https://news.netcraft.com/archives/2019/06/17/june-2019-web-server-survey.html
[49]https://www.servethehome.com/apache-drops-below-50-percent-market-share-nginx-claims-over-33-percent/

shows that different software practitioners have different needs. For example, in DB, i.e., a specialized website for database-related queries, there exists a significant discussion about *log file growing and shrinking*. Database administrators face problems with the growing size of log files over time as the log files continuously increase in size corresponding to database transactions. To this end, solutions have been proposed, such as creating distributed storage for log files or automatically truncating their frivolous parts. In addition, a significant discussion about the *'Logger'* class of log4j is observed on SO. It seems that the 'Logger' class of log4j requires an extended documentation along with detailed FAQs that would aid software practitioners in its usage, as well as provide solutions to frequently encountered errors and bugs. Also, we observe that the SF website has logging topics related to *log analysis*, implying that network administrators require tools to search the logs efficiently. Notably, we observe that current studies do not involve all these stakeholders while developing solutions. Software developers need to be fully aware of the requirements of different stakeholders to improve the current logging tools and techniques. In this regard, the results obtained from topic analysis exercises in RQs 4-6 would be beneficial. In essence, our results suggest that future logging studies should consider the needs of different stakeholders and take their feedback to improve logging tools and techniques. Notably, identifying the logging needs of different software practitioners is an open problem for researchers and developers as there is considerable scope for cutting-edge research.

Selection of programming languages based upon logging environment: We have observed that each programming language has a niche in the logging arena, with a few overlaps. Novice software practitioners often struggle to select the best programming language to complete the task. Although every programming language can achieve a certain set of instructions, the goal of a software practitioner is to implement the task at hand efficiently in the least amount of time. Thus, based on empirical evidence, we recommend that software practitioners select programming languages based on their logging environment and the nature of the task. For instance, we have observed that C is chiefly used to implement logs across multiple devices in a network or a server environment. C++ is considered widely for implementing logging in a multithreaded and multiprocessing environment. C# is used to implement logging in a .NET environment. With several efficient logging libraries, Java is utilized to implement logging in Software and Java-based Projects. JavaScript finds most use in implementing logging in the Web Applications, whereas Python is best used for log management and analytics.

Dissatisfaction in the Android Community: Although AE receives approximately the same number of views as that of any other analyzed website, it hosts an *extremely limited set of successful logging questions* (RQ1). Our recommendation to the Android Community, including Android developers at Google, is to be attentive towards the concerns of the android community. Therefore, future research should pay attention to improving the logging practices for Android development.

Stringent growth in the usage of NLog. NLog appears to be an emerging alternative to Log4Net. Log4Net is a widely used logging tool for .Net Framework, whereas NLog has gathered attention in recent years. With the categorization of Log4Net as a dormant or "inactive" project by Apache Logging Services on 1st April 2020, a stringent growth in the usage of NLog is expected. Our recommendation for NLog developers is to increase their resources similar to Log4Net and heed caution at each stage of the new release. For example, NLog has been associated in the past with bugs, such as "*Swallowing log messages*", along with "*skipping log messages*" in a multithreaded logging environment on a windows server[50][51].

The usage trend of logging libraries and tools is evolving. Our study reveals several interesting trends with respect to logging. For instance, a continuously increasing trend is observed for event logging on SU, which is related to Microsoft Windows. On the one hand, this increasing trend implies its popularity among IT practitioners; on the other hand, it depicts difficulty or ambiguity in its usage. Analysis of logging topics with a decreasing trend over time is essential for corresponding developers to salvage their products from getting obsolete. For example, since the discussion related to logging in Apache is decreasing on SF, the concerned team should audit if (1) there exists any lagers or blockers that are disabling IT-practitioners from using their product or (2) their product has significant ease of usage such that IT-practitioners rarely find its usage challenging.

Importance of SE for source-code logging research: Among all the analyzed websites, SE stands out when it comes to questions related to source code logging. With logging being an integral part of software engineering, users on this website discuss state-of-the-art enhancements. Empirical evidence indicates that a strong emphasis is given to logging design patterns, exception logging, architecture, and performance, along with debates on cutting-edge concepts like the internationalization of logs. In contrast, other websites, except SO, focus on the implementation of logging within a particular environment. In addition to that, SE invites a great deal of viewership and has the most percentage of successful logging questions compared to other websites. Thus, we recommend researchers researching source code logging to stay updated with the

[50]https://github.com/NLog/NLog/issues/1652
[51]https://stackoverflow.com/questions/34754681/nlog-is-skipping-log-messages-net-issue-multi-thread-issue

developments on SE since this would help them understand the perspective of software developers and help in bridging the gap between the research community and software engineers.

Linking Q&A websites with other software repositories: Prior studies on logging mainly focus on one type of software repository, such as source code[6 29 9] or issue tracker[11]. Our research shows a path towards a novel research direction in logging, which is integrating logging across the software engineering communities, i.e., integrating Q&A websites with the issue tracker and version control system. A survey performed by Correa et al.[52] on software developers from several open-source projects shows that approximately 50% of the developers conduct an extensive online search before filing any issue. In addition, Vasilescu et al.[53] compared the activities of the authors on SO and r-help mailing list. Their work indicates that the experts are migrating from the mailing list to SO. In addition, they notice that experts provide answers relatively promptly on SO than they do on the mailing list. Gomez et al.[54] analyze the data on SO and found that SO plays a crucial role in software innovation dissemination. The knowledge present on Q&A websites can guide researchers and practitioners who build logging tools and libraries to explore and adopt software development innovations.
We made the relevant files available at the following location so that our results can be accessed by the other researchers: https://github.com/newtein/EmpiricalAnalysesLogging

8 | THREATS TO VALIDITY

In this section, we discuss various threats to validity related to the results presented in this work.

8.1 | External Validity

In this study, we analyze six Q&A websites from the StackExchange network[52]. The findings, discussion on topics, and other results obtained in this study may not be generalized to other websites/systems. Additional case studies on different systems are required to obtain more universal results. However, we performed our case study on six different websites with different sizes (in terms of users and number of questions) and domains. Moreover, we conduct our experiments on SO, which is one of the widely used Q&A websites for software practitioners. Notably, these websites highlight generic concerns, e.g., SO, or specific domain-related concerns, e.g., AE, DB, SE, SF, and SU. Nevertheless, due to inter-dependencies among these domains, a slight overlap is anticipated. In the future, we plan to examine additional websites from different domains.

8.2 | Internal Validity

We studied the issues faced by the software practitioners through the consideration of Q&A websites. However, in addition to Q&A websites, software practitioners also use other platforms such as issue reports and mailing lists. Therefore, the results obtained may not provide a complete view of these issues. Nevertheless, we analyze the logging issues from six different large and popular Q&A websites. Thus, we expect that our findings should be representative of the issues faced by software practitioners. Also, this study aims to explore the logging issues faced by all types of software practitioners, not specifically by software developers or coding-related logging issues. Because the issue tracking system, mailing list, etc. are often used by software developers to ask coding related issues. Thus, in the future, we plan to combine the issues tracking system and Q&A websites, thereby performing a combined study.

8.3 | Construct Validity

8.3.1 | Selection of Questions for Empirical Analysis

We analyze logging questions about 6 websites and 6 programming languages. The data from other websites, domains, and languages are needed to do a more rigorous analysis of logging questions faced by users. Two authors of this paper perform an extensive internet search to find all the logging related tags corresponding to these six programming languages. However, due to substantial logging libraries and tools, there exists a possibility that we may have missed some logging related tags. Table A1 shows a list of logging tags used in this work. This list provides a comprehensive collection of all the logging tags searched in the selected six websites. The list consists of tags related to the transactions, log-files,

[52]https://stackexchange.com/sites

logging libraries, Tools, and APIs. In addition, we observed that sometimes developers use logging tags in a different context. For example, Lynx and Hugo can be inconsistent as Lynx is a logging library, as well as a text-mode web browser. Similarly, Hugo is a logging library and also a static site generator. We attempted to remove all such false positives and negatives; still, there is a possibility of error. Thus, a more robust method is required to identify all the logging tags. Currently, we used tags assigned to questions to retrieve all the logging questions. There is a possibility that there are some ongoing relevant questions that do not have a logging tag. Listing 1[53] shows some example of questions related logging without logging tag. We believe that such questions do not have logging as their main concern, and hence, the user or the community has not added any logging tag to it. However, a more detailed analysis is required with respect to logging questions that do not have a logging tag.

Listing 1: Example of logging question without logging tag (website: SE qid:277325)

```
Title: How would a one man team benefit from a continous integration
       setup?
Body: I am a one man team. I develop an in house database, and the
       webinterface.

My current workflow is code on my development machine. Test.
Push to a central git server.
Log into the production server pull the changes over. Restart the
       production server.

Its Python based. I tried setting up Fabric one time, but I felt it was a
       fair bit of setup, and I didnt really gain anything, as such I didnt
       end up using it.

Would someone like to convince me otherwise? (Or agree).

On a side note, is Hudson/ Jenkins doing the same job as Fabric? Or are
       these for different purposes. Am I misunderstanding the concepts?

Tags:continuous-integration, solo-development
```

In contrast, the selection of questions through the aforementioned tags could render a few questions where logging is not the major concern. However, the 'logging' tag is present. The inclusion of tags in any question is governed by the discretion of the community. Thus, based on this discretion, we include all the questions with relevant logging tags. Nevertheless, there exists the possibility of some low-quality questions in our corpus that need manual intervention for verification.

8.3.2 | Topic Modeling

Studies have shown that varying hyperparameters can modify the LDA results; however, determining the suitability of topics by manual intervention is not entirely feasible due to the unstructured nature of topics (k). Thus, this might pose as a construct validity for the LDA apparatus. The value of k is chosen on the basis of the problem at hand (Binkley et al.[45]). In this study, the value of k is taken as the function of the size of a website. Thus, the value of k chosen for smaller and larger websites are 100 and 500, respectively. We report all the unstructured topics across websites by making them publicly available; however, we focus on the top ten topics having the highest topic presence for labeling and trend analysis due to limited bandwidth available for research.

Interpretation of the topics generated by LDA is subjective and can vary from one researcher to another[55]. To this end, two authors of this paper performed brainstorming to identify the correct topics. In case of conflict or confusion, both authors analyzed several questions belonging to that topic to identify the most suitable topic in each category.

9 | CONCLUSION AND FUTURE WORK

To understand the logging needs of different software practitioners, we performed a three-level exploratory study of logging questions on six Q&A websites. The results showed that logging is used by different software practitioners and confirmed the assumption that the logging needs of software practitioners are different. This motivates future studies on logging to consider the logging needs of different software practitioners. This work revealed several logging issue hot spots, such as boostlog, syslog, and Log4j. Companies and open source developers can learn from our results

[53]https://softwareengineering.stackexchange.com/questions/277325/how-would-a-one-man-team-benefit-from-a-continous-integration-setup

to identify logging tools and libraries that need further improvements. This work unraveled several interesting and novel high-level logging topics (i.e., logging issues), such as *logging conversion pattern*, *android device logging*, *database logging*, *Linux logging, logging file creation/deletion/append, and logging level*. Additionally, this study showed logging topics that are domain specific, e.g., DB (*Log shipping, Log file growing/shrinking*) and SU (*Event log, Syslog configuration*). Future studies on logging improvement and automation can further explore these topics to provide better tools and techniques for logging improvement. The trend analysis of logging topics showed a rising trend (e.g., *event logging* for the SU website) and falling trend (*Apache web server logging* for the SF website) of some logging topics, which motivates the further studies to focus on rising logging topics. This work highlighted the importance of considering the logging needs of practitioners from different domains when conducting logging-related research. In addition, this study sheds light on a future direction towards linking the Q&A websites with other software repositories like the issue tracker system and version control systems to further understand the logging needs from the perspective of different software practitioners.

References

1. Zhu J, He P, Fu Q, Zhang H, Lyu M, Zhang D. Learning to Log: Helping Developers Make Informed Logging Decisions. In: *Proceedings of the IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)*. 1. ; 2015: 415-425.
2. Fu Q, Zhu J, Hu W, et al. 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 2014)*; 2014: 24–33.
3. Yuan D, Mai H, Xiong W, Tan L, Zhou Y, Pasupathy S. SherLog: error diagnosis by connecting clues from run-time logs. In: *Proceedings of the fifteenth International Conference on Architectural support for programming languages and operating systems*; 2010: 143–154.
4. Yuan D, Zheng J, Park S, Zhou Y, Savage S. Improving software diagnosability via log enhancement. *ACM Transactions on Computer Systems (TOCS)* 2012; 30(1): 1–28.
5. Yuan D, Park S, Huang P, et al. Be Conservative: Enhancing Failure Diagnosis with Proactive Logging. In: *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation (OSDI)*; 2012: 293–306.
6. Lal S, Sureka A. LogOpt: Static Feature Extraction from Source Code for Automated Catch Block Logging Prediction. In: *Proceedings of the 9th India Software Engineering Conference (ISEC)*; 2016: 151–155.
7. Lal S, Sardana N, Sureka A. LogOptPlus: Learning to Optimize Logging in Catch and If Programming Constructs. In: *Proceedings of the IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. 1. ; 2016: 215-220
8. Jia Z, Li S, Liu X, Liao X, Liu Y. SMARTLOG: Place error log statement by deep understanding of log intention. In: *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*IEEE. ; 2018: 61–71.
9. Li H, Shang W, Hassan AE. Which log level should developers choose for a new logging statement?. *Empirical Software Engineering* 2017; 22(4): 1684–1716.
10. Kabinna S, Bezemer CP, Shang W, Hassan AE. Examining the Stability of Logging Statements. In: *Proceedings of The 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*; 2016: 326–337.
11. Li H, Shang W, Adams B, Sayagh M, Hassan AE. A Qualitative Study of the Benefits and Costs of Logging from Developers' Perspectives. *IEEE Transactions on Software Engineering* 2020.
12. Barua A, Thomas SW, Hassan AE. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering* 2014; 19(3): 619–654.
13. Pinto G, Castor F, Liu YD. Mining questions about software energy consumption. In: *Proceedings of the 11th Working Conference on Mining Software Repositories*ACM. ; 2014: 22–31.
14. Linares-Vásquez M, Dit B, Poshyvanyk D. An exploratory analysis of mobile development issues using stack overflow. In: *Proceedings of the 10th Working Conference on Mining Software Repositories*IEEE Press. ; 2013: 93–96.
15. Hassan AE. The road ahead for mining software repositories. In: *Frontiers of Software Maintenance, 2008. FoSM 2008.*IEEE. ; 2008: 48–57.
16. Community S. stackoverflow home page. https://stackoverflow.com/; . [accessed 26-Dec-2017].

17. Community S. Serverfualt stack exchange home. https://serverfault.com/; . [accessed 26-Dec-2017].

18. Community S. Superuser Stack Exchange home page. https://superuser.com/; . [accessed 26-Dec-2017].

19. Community S. Database Administrators Stack Exchange home page. https://dba.stackexchange.com/; . [accessed 26-Dec-2017].

20. Community S. SoftwareEngineering home page. https://softwareengineering.stackexchange.com/; . [accessed 26-Dec-2017].

21. Community S. Android Enthusiasts home page. https://android.stackexchange.com/; . [accessed 26-Dec-2017].

22. Gujral H, Sharma A, Lal S, Kumar L. A Three Dimensional Empirical Study of Logging Questions From Six Popular Q&A Websites. *e-Informatica Software Engineering Journal* 2019; 13(1): 105–139. doi: 10.5277/e-Inf190104

23. Yuan D, Park S, Zhou Y. Characterizing Logging Practices in Open-source Software. In: *Proceedings of the 34th International Conference on Software Engineering*(ICSE). ; 2012: 102–112.

24. Shang W, Nagappan M, Hassan AE. Studying the relationship between logging characteristics and the code quality of platform software. *Empirical Software Engineering* 2015; 20(1): 1–27. doi: 10.1007/s10664-013-9274-8

25. Chen B, Jiang ZMJ. Characterizing logging practices in Java-based open source software projects–a replication study in Apache Software Foundation. *Empirical Software Engineering* 2017; 22(1): 330-374.

26. Kabinna S, Bezemer CP, Shang W, Hassan AE. Logging Library Migrations: A Case Study for the Apache Software Foundation Projects. In: *Proceedings of the 13th International Conference on Mining Software Repositories*MSR '16. ACM; 2016; New York, NY, USA: 154–164

27. Li H, Shang W, Zou Y, Hassan AE. Towards just-in-time suggestions for log changes. *Empirical Software Engineering* 2017; 22(4): 1831–1865.

28. Lal S, Sardana N, Sureka A. Two Level Empirical Study of Logging Statements in Open Source Java Projects. *International Journal of Open Source Software and Processes (IJOSSP)* 2015; 6(1): 49–73.

29. Lal S, Sardana N, Sureka A. Improving Logging Prediction on Imbalanced Datasets: A Case Study on Open Source Java Projects. *International Journal of Open Source Software and Processes (IJOSSP)* 2016; 7(2): 43–71.

30. Lal S, Sardana N, Sureka A. ECLogger: Cross-Project Catch-Block Logging Prediction Using Ensemble of Classifiers. *e-Informatica Software Engineering Journal* 2017; 11(1): 9–40. doi: 10.5277/e-Inf170101

31. Gujral H, Sharma A, Lal S. Empirical analysis of Q&A websites and a sustainable solution to ensure water-security. In: *2018 Eleventh International Conference on Contemporary Computing (IC3)*IEEE. ; 2018: 1–7.

32. Beyer S, Pinzger M. A manual categorization of android app development issues on stack overflow. In: *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*IEEE. ; 2014: 531–535.

33. Yang XL, Lo D, Xia X, Wan ZY, Sun JL. What security questions do developers ask? a large-scale study of stack overflow posts. *Journal of Computer Science and Technology* 2016; 31(5): 910–924.

34. Malik H, Zhao P, Godfrey M. Going green: An exploratory analysis of energy-related questions. In: *Proceedings of the 12th Working Conference on Mining Software Repositories*IEEE Press. ; 2015: 418–421.

35. Nagy C, Cleve A. Mining Stack Overflow for discovering error patterns in SQL queries. In: *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*IEEE. ; 2015: 516–520.

36. Blei DM, Ng AY, Jordan MI. Latent dirichlet allocation. *Journal of machine Learning research* 2003; 3(Jan): 993–1022.

37. Thomas SW, Adams B, Hassan AE, Blostein D. Studying Software Evolution Using Topic Models. *Science of Computer Programming* 2014; 80: 457–479. doi: 10.1016/j.scico.2012.08.003

38. Tian K, Revelle M, Poshyvanyk D. Using Latent Dirichlet Allocation for automatic categorization of software. In: *Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories (MSR)*; 2009: 163-166.

39. Pagano D, Maalej W. How Do Open Source Communities Blog?. *Empirical Software Engineering* 2013; 18(6): 1090–1124. doi: 10.1007/s10664-012-9211-2

40. De Lucia A, Di Penta M, Oliveto R, Panichella A, Panichella S. Using IR methods for labeling source code artifacts: Is it worthwhile?. In: *2012 20th IEEE International Conference on Program Comprehension (ICPC)*IEEE. ; 2012: 193–202.

41. De Lucia A, Di Penta M, Oliveto R, Panichella A, Panichella S. Labeling source code with information retrieval methods: an empirical study. *Empirical Software Engineering* 2014; 19(5): 1383–1420.

42. Thomas SW, Adams B, Hassan AE, Blostein D. Validating the use of topic models for software evolution. In: *2010 10th IEEE Working Conference on Source Code Analysis and Manipulation*IEEE. ; 2010: 55–64.

43. Li H, Chen THP, Shang W, Hassan AE. Studying software logging using topic models. *Empirical Software Engineering* 2018: 1–40.

44. TIOBE . TIOBE Index for January 2018. https://www.tiobe.com/tiobe-index/; . [accessed 1-Jan-2018].

45. Binkley D, Heinz D, Lawrie D. Understanding LDA for Software Engineering.

46. Geeks JC. Log4j Conversion Pattern Example. https://examples.javacodegeeks.com/enterprise-java/log4j/log4j-conversion-pattern-example/; . [accessed 21-Oct-2019].

47. Wang T, Johnson R, Pandis I. Query fresh: Log shipping on steroids. *Proceedings of the VLDB Endowment* 2017; 11(4): 406–419.

48. Cherkauer KJ, Pearson SR, Xue X, Zheng RL. Log-shipping data replication with early log record fetching. 2018. US Patent 9,864,772.

49. Qin D, Brown AD, Goel A. Scalable replay-based replication for fast databases. *Proceedings of the VLDB Endowment* 2017; 10(13): 2025–2036.

50. Chowdhury S, Di Nardo S, Hindle A, Jiang ZMJ. An exploratory study on assessing the energy impact of logging on android applications. *Empirical Software Engineering* 2018; 23(3): 1422–1456.

51. Zeng Y, Chen J, Shang W, Chen THP. Studying the characteristics of logging practices in mobile apps: a case study on F-Droid. *Empirical Software Engineering* 2019; 24(6): 3394–3434.

52. Correa D, Lal S, Saini A, Sureka A. Samekana: A browser extension for including relevant web links in issue tracking system discussion forum. In: *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*. 1. IEEE. ; 2013: 25–33.

53. Vasilescu B, Serebrenik A, Devanbu P, Filkov V. How social Q&A sites are changing knowledge sharing in open source software communities. In: *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*; 2014: 342–354.

54. Gómez C, Cleary B, Singer L. A study of innovation diffusion through link sharing on stack overflow. In: *2013 10th Working Conference on Mining Software Repositories (MSR)*IEEE. ; 2013: 81–84.

55. Hindle A, Bird C, Zimmermann T, Nagappan N. Do topics make sense to managers and developers?. *Empirical Software Engineering* 2015; 20(2): 479–515.

Field	**SO**	**SU**	**SF**	**DB**	**SE**	**AE**
Total Number of Unique Users	8287574	630516	346259	114789	241851	154687
Total questions	14995834	363915	252963	60948	47362	46559
Total questions with accepted answer	8034235	154322	125601	29400	27762	13316
Total logging questions	75185	1275	4227	1131	198	183
Total logging questions with accepted answer	39674	541	2163	555	110	50
Percentage of logging questions to total questions	0.19	0.14	0.62	0.78	0.10	0.17
Timestamp of the First Question	8/1/2008	7/15/2009	4/30/2009	10/22/2009	9/27/2010	9/1/2010
Timestamp of the Last Question	12/3/2017	11/30/2017	12/1/2017	12/1/2017	11/19/2017	11/28/2017

TABLE 1 Details of the experimental dataset of StackExchange websites: SO, SU, SF, DB, SE, and AE

Parameter	Value	Description
–num-topics	varied across websites	Number of Topics
–alpha	50.0	It is a hyperparameter that gives the measure of confidence of moving away from the prior.
–beta	0.01	The default value for beta is 0.01. This means that each topic has a weight on the uniform prior to equal to the size of the vocabulary divided by 100.
–num-iterations	10000	Number of iterations
–optimize-burn-in	1000	The number of iterations before hyperparameter optimization begins.
–optimize-interval	100	This option turns on hyperparameter optimization, which allows the model to better fit the data by allowing some topics to be more prominent than others.
–num-threads	4	The number of threads for parallelization.

TABLE 2 Configuration of topic modeling using MALLET

Research Level	Research Questions
Popularity and Frequency Analysis	RQ1: What domains are affected by logging? What is the response of the community towards logging questions in each domain?
Popular Words and Tags analysis w.r.t Websites and Programming Language	RQ 2: What are the differentiating words/terms present in the title, body, and tag of logging questions with respect to six websites (analysis with respect to each domain) ? RQ3: What are the differentiating words/terms present in the title, body, and tag of logging questions with respect to six programming languages (analysis with respect to each programming language) ?
Semantic Analysis	RQ 4: What are the most popular logging topics that software practitioners are talking about across the Q&A websites? RQ 5: Do different Q&A websites have different logging topics? RQ 6: How does the interest in logging techniques has changed over the period of time?

TABLE 3 Details of research levels and research questions

	AE		DB		SF		SE		SO		SU	
	Tag	Frequency	Tag	Frequency	Tag	Frequency	Tag	Frequency	Tag	Frequency	Tag	Frequency
1	applications	18	sql-server	553	log-files	924	java	31	java	10714	event-log	258
2	adb	14	transaction-log	490	linux	580	c#	23	log4j	7136	linux	235
3	crashes	14	mysql	230	syslog	515	design	20	c#	4830	windows	178
4	4.1-jelly-bean	9	log-shipping	192	apache-2.2	493	exceptions	15	logstash	4828	windows-7	172
5	security	8	backup	141	rsyslog	447	design-patterns	13	python	3484	syslog	88

TABLE 4 Top five tags in logging questions across websites.

Programming Language	Tags
Java	Log4J, SLF4J, tinylog, logback, apache commons logging, google-cloud-java, commons-logging, jboss-logging, java-util-logging, syslog4j, otroslogviewer, log4j, log4j2, log4jdbc, slf4j, slf4j, jul-to-slf4j, hyperloglog.java, java.util.logging
C	Log4C, sclog4c, syslog, zlog, zf_log, log4c
C++	glog, log4cplus, pantheios, boost::log, easylogging++, log4cxx, boost, boost-log, boost-logger, boost.log, spdlog, log4cpp
Python	pygogo, Logbook, google-cloud-python, django-logging, logger-python, hyper-loglog.python, unified-log, auth.log, graylog, graylog2
C#	log4net, NLog, Enterprise Library, Common.Logging, log4net-configuration, log4net-appender, log4net-filters
JavaScript	js-logging, Log4js, log4javascript, JSNLog, Node-Loggly, Bunyan, Winston, Morgan, Angular-Loggly, loglevel, jsnlog, log-level, logsene-js, node-nslog, truncate-logs-js

TABLE 5 Tags used to extract questions related to programming languages.

	C		C++		C#		Java		Javascript		Python	
	Tag	Frequency	Tag	Frequency	Tag	Frequency	Tag	Frequency	Tag	Frequency	Tag	Frequency
1	syslog	1261	boost	256	log4net	3265	log4j	7155	node.js	479	django	370
2	linux	302	boost-log	216	nlog	1448	logback	2243	winston	352	python-2.7	240
3	rsyslog	250	log4cxx	119	.net	1332	slf4j	2151	bunyan	77	graylog2	236
4	syslog-ng	94	log4cplus	102	asp.net	669	log4j2	2009	express	74	python-3.x	153
5	c++	78	glog	80	enterprise-library	643	spring	1081	console	56	flask	109

TABLE 6 Top five tags in logging questions across programming languages.

S.No	Website	Year	Platform	Views	Question
1.	SO	2013	Linux	5000	**Boost.Log linking** errors under GNU/Linux [54]
2.	SO	2014	Fedora (Linux)	30000	linker error while **linking boost log** tutorial (undefined references) [55]
3.	SO	2015	Linux	558	**boost** static library **link failed** in g++ [56]
4.	SO	2016	Linux	435	Can't **link boost logging** code [57]

TABLE 7 Examples of boost-log errors

S.No	Topic	Topic Score	Words Related to this topic
1.	Logging Applications	0.0164	logging good **application** approach question **solution** make time implement system based practice kind things lot wondering case idea thinking applications
2.	Logging Conversion Pattern	0.0093	log **appende**r apache org layout file **conversion pattern pattern layou**t logger properties debug info rootlogger logs rollingfileappender yyyy console threshold maxfilesize consoleappender
3.	Automated Logging	0.0086	log logging don **simple** make information capture single doesn set code logged specific things work question **automatically standard** add point
4.	Android Device Logging	0.0078	**logcat** android device app phone eclipse **adb emulator** application output apps tag debug running run devices **galaxy** show debugging **samsung**
5.	Log Files: Creation/Append	0.0073	file log files **created** path problem create **append** written set logfile content creates **appended** advance filename overwrite exists directory empty
6.	Log Files: Multiple Log Files	0.0063	log **files** file separate logs **multiple** based generate generated single create application logging names write created specific location achieve dynamically
7.	Searching Logging Solution	0.0062	find found question log read answer documentation work information lot didn couldn **google searched** don questions solution similar **stackoverflow** link
8.	Logging in Database	0.0057	log **database transaction file sql server** space recovery full size shrink backup **dbcc** simple disk model databases mode ldf growing
9.	Linux Logging	0.0053	**syslog** server messages **rsyslog** var local conf send facility **syslogd** remote udp logs port host ubuntu message daemon hostname source
10.	Logging Levels	0.0052	**level** log info **debug** logging levels **error** set **warn** logger messages change loglevel warning **fatal** verbose logs **trace** higher logged

TABLE 8 Top 10 logging topics across six websites

S.No	Topic	Website	Qid	Title and Summary
1.	Logging Applications	SF	194037	**Title:** good data visual package for unix? **Body:** Quite often I'm looking a large blocks of data (logs files) looking for oddities in timing from processes. While I can convert this to an .csv/tab file easy enough, I have issues trying to render it in a graph in unix. Many packages such as rrd are good, but are specific usage based. Does anyone know of a good open source package to generate an image(s) of data? Not too much programming wanted: Features wanted: 3d plots, histograms, variation (stddev_pop), etc.. The data set isn't that large, probably $<$ 1 million lines ever.
2.	Logging Conversion Pattern	SO	11245993	**Title:** grails / log4j / conversion pattern / %throwable /....... **Body:** I use log4j 1.2.16 regarding my "dependency-report". my **Conversion Pattern** is '%dyyyy-MM-dd HH:mm:ss,SSS %-5p %c: %m%n %throwableshort'. but %throwable is not recognized, instead the **loglines** contain '...hrowableshort...'. Any idea ?
3.	Automated Logging	SF	193100	**Title:** Log LDAP access of the Active directory. **Body:** I am looking for a method to log ldap access of a Active Directory domain controller. I want to be able to log the username and source IP address access to both 389, and 636(encrypted). A simple packet capture would get me the source IP, but getting the username will not be possible over ldaps so I am hoping there is some **built-in auditing/debug/logging** feature in Windows that will give me this information
4.	Android Logging	AE	142694	**Title:** Strange sound from an unknown application **Body:** From yesterday my Samsung Galaxy S3 (Android 4.4.2) is playing strange music (midi) from time to time and very often. Sometimes it goes as it comes, with no obvious reason. It looks like an alarm. There is no notification. I checked everything a have remember can cause this. The only way to stop music intentionally is to open task manager and clear RAM. I also have tried to stop services and running applications from Settings -> Application manager, but that didn't seem to make change. Can I detect somehow which process started this music? Because I can use **Android** Eclipse, is there any **log** that can help me to trace origin. In **logcat** I can see that the music player is running.
5.	Log File: Creation/Append	SO	10738953	**Title:** Log data printing multiple times **Body:** I have implemented a custom log file using log4j for my web application. But the problem is when I am printing any thing in that log file, it prints multiple times. Whether there is no loop or any kind of iteration in the code. Can any one please help me to sort out this problem.
6.	Log File: Multiple Log Files	SF	192134	**Title:** .bat files to find **all files** on drive bigger than X **Body:** So far I run a .bat file to find all the media files stored on our student shared drive, which then writes to a .txt file in my documents, for example: dir S:*.mp3 /s$>$ "M: \logs \student \mp3.txt", This is probably not the best way of doing it and I am aware there are probably better tools in 2008 R2 to do this for me. I am now trying to find a way to **only log files** that are, for example, bigger then 100mb. Is this doable in a .bat files/cmd.exe, am I better off using PowerShell, or use the fuctions in Server 2008R2?
7.	Searching Logging Solution	SO	10437544	**Title:** Java Remote Logging through HTTPS 443. **Body:** If I have a java applet for a site, is it possible to do remote logging through HTTPS? how does this work? I know there is java sockethandler, which could direct the log to a server, but what should I do in the server side, so I can get the log and save it to a file in the server. I try to create a logserver app and put it on server but if I specify port 443, it will say 'java. net. Bind Exception: Address already in use: JVM_Bind' Can someone **point me out and give some examples** what should I do in the applet side and server side?
8.	Logging in Database/-Transaction	DB	119694	**Title:** Would SQL in a cluster failover when transaction log for database is full? **Body:** I have a Microsoft SQL Cluster with one instance. If one of the databases is full, would SQL in a cluster failover when **transaction** log for database is full? If it depends on the reason the **transaction log** is full please refer to following:: 1. due to 'log_backup', 2. due to 'availability replica', 3. active_transaction, 4. 'replication', 5. CHECKPOINT
9.	Linux Logging	SF	278369	**Title:** Forward UNIX syslog to Windows Event viewer **Body:** I'm running a pfSense firewall which runs syslog and can forward it's logs to a remote syslog server. I'd like to be able to view these on my SBS 2011 server's event viewer, via a subscription. I assume there has to be some middleware which translates syslog messages to windows events, but I cannot find such a program. There are lots of programs for sending windows events to a unix syslog server, but that is the opposite of what I want...
10.	Logging Levels	SF	28268	**Title:** How old is the "severity" paradigm in logging? **Body:** Years of sysadmin left syslog's severity levels, as described by The BSD Syslog Protocol, clearly imprinted in my mind. You know the drill: Emergency, Alert, Critical, Error, Warning, Notice, Informational and Debug. This left traces elsewhere, such as Java's logger with its Severe, Warning, Info, Config, Fine/r/st.While discussing it with someone under the impression that Java's was a quick hack, a bad fit, and telling of the mindset, I wondered how old this actually is – syslog's dating back to the 80s with Sendmail. A quick search reveals that REXX has Termination, Severe, Error, Warning, Informational & Response, which seems to confirm my suspicions.......

TABLE 9 Examples related to the top-10 topics obtained from six websites

Website	S.No	Topic	Topic Number	Topic Presence	Words Related to this topic
SO	1.	Framework-specific logging problems	(63)	0.019796	logging application solution good approach implement question make don case system based specific practice write idea support standard framework handle
	2.	**Log file creation/deletion**	(324)	0.012734	log **file created** files **create written** write empty creating application path time problem filename writing creates append content logfile location
	3.	Logger Conversion/Layout	(160)	0.010308	log **appender** apache org **layout** file **pattern** layout **conversion pattern** properties logger debug info root logger yyyy rolling fileappender logs consoleappender true threshold maxfilesize
	4.	**Android Emulator Logging**	(80)	0.008433	**logcat android device app adb eclipse studio** phone emulator devices messages application tag show running debug run usb debugging showing
	5.	**Logger class of log4j**	(447)	0.00632	**logger class static public getlogge**r private log final info import void string **loggerfactory myclass java** getname code main logmanager logging
DB	1.	**Log shipping**	(432)	0.020187	**shipping** secondary **log** primary database **server** restore job setup mode servers standby restoring configured jobs fails restored fail copy configuration
	2.	**Log file growing/shrinking**	(355)	0.017648	file log **size shrink** database **growing** sql **shrinking** mode space backup full recovery grow reduce day server growth initial simple
	3.	Backup	(4)	0.01531	backup full log backups transaction restore differential taking minutes chain hour hours night scheduled day restored daily correct databases diff
	4.	SQL Server Management Studio	(119)	0.011605	server sql log run instance make existing scenario put running question ssms complete short happen settings information msdn cut explain
	5.	**Log file creation/deletion**	(215)	0.010949	**log file files time created** question don shows automatically running suggestions enabled recently deleted safe previous exists move group attempting
SU	1.	**Event Log**	(425)	0.023095	**event windows log viewer** events logs system running application service logged information manager **microsoft** shows source services desktop eventlog internet
	2.	Searching content in log files	(197)	0.012188	log file create directory files var find searching contents path entries show ubuntu purpose debugging created separate big provide missing
	3.	Command line log	(461)	0.006019	command line file log output text lines specific expected achieve great flag updated master put mode rules pipe answer appended
	4.	**Syslog configuration**	(403)	0.00526	**syslog** server remote send rsyslog syslogd **configure conf configured** standard host forward receive centralized docs sending level messages configurations capture
	5.	Logging system time	(21)	0.005201	computer specific times time track activity software running method work determine stops asked macbook win minutes started pro night advance
SF	1.	Linux var/log directory	(215)	0.010981	log var file empty conf files created cat config configuration directory lines wrong works grep restarted added output messages idea
	2.	**Apache web server logging**	(346)	0.007179	**apache** log access file logs **error web server** files requests php mod website **httpd** site found configuration doesn lamp htaccess
	3.	Log file changing or rotation	(350)	0.006754	log file files write written content tmp separate line don location open add logfile contents check put linux change option
	4.	**Log analysis**	(499)	0.006676	tool log logs web good tools time software **analysis analyzer analyze** based free files **awstats** give open view **statistics** source
	5.	**Syslog configuration**	(303)	0.005208	**syslog** messages message **conf** running send **configuration** problem source syslogd udp version receiving setting advance central sends forwarding difference assume
AE	1.	**Android Emulator Logging**	(13)	0.133604	logcat adb device android logs file read access output crash root command show system shell emulator terminal running run nexus
	2.	**App Install**	(7)	0.063215	**apps app google install play installed** store list don tablet question user android history back applications file exception crashes account
	3.	Mobile phone ROM issues	(0)	0.057305	problem update battery phone logcat device samsung rom started reboot running freeze installed determine button factory power alogrec happened home
	4.	**Usage tracking**	(14)	0.035013	**data usage** connection sending **track monitoring** http **background** texts resource suggestions server location site rate net collect display active type
	5.	Android boot-loop issues	(11)	0.032077	boot debugging log android phone kernel false bootloop bootloader messages emulator ret code nexus crashes kind fastboot panic usb stuck
SE	1.	**Client server logging**	(4)	0.061856	**log web service server client logs** app application services send email net call strategy asp queue session page calls user
	2.	Production log	(3)	0.054915	log level logs trace application debug information find production message logged questions state logging test environment tracing framework specific answer
	3.	Debug Logging	(16)	0.054375	string public log return class void method private methods logging engineruntimeexception property code thirdpartyclass true content interface trace telemetryclient wrap
	4.	Logger class log4j	(7)	0.049981	logger class object application thread instance objects getlogger loggers pattern warn create global structure static module myloghelper cases singleton level
	5.	**Exception logging**	(9)	0.043996	**exception catch exceptions error** throw step log dangerous externalapp public code void static handling application control errors event production class

TABLE 10 Top 5 topics and their respective topic presence across six websites

APPENDIX

A TAGS USED TO EXTRACT LOGGING QUESTIONS

TABLE A1 Tags used in this study to extract logging questions

S.No	Tag	S.No	Tag	S.No	Tag
1	logging	58	Log4C	114	kotlin-logging
2	logs	59	sclog4c	115	hugo-logging
3	transaction-log	60	syslog	116	ninject-logging
4	log-shipping	61	zlog	117	.net-core-logging
5	binary-log	62	zf_log	118	ms.extensions.logging
6	binlog	63	glog	119	powershell-logging
7	bin-log	64	log4cplus	120	java-util-logging
8	log	65	pantheios	121	syslog4j
9	error-logging	66	boost::log	122	sys-log
10	error-log	67	easylogging++	123	ms-extensions-logging
11	mysqlbinlog	68	log4cxx	124	nslogger
12	logparser	69	pygogo	125	get-eventlog
13	archive-log	70	logbook	126	custom-eventlog
14	unlogged-tables	71	log4net	127	stay-logged-in
15	logfiles	72	Nlog	128	android-logcat
16	logfile	73	Enterprise Library	129	otroslogviewer
17	logfile-analysis	74	Common.Logging	130	fusion-log-viewer
18	log-analysis	75	js-logging	131	log4j
19	log-files	76	Log4js	132	log4j2
20	syslog	77	log4javascipt	133	log4js-node
21	logparser	78	JSNLog	134	log4jdbc
22	logwatch	79	xlog	135	gwt-log4j
23	window-event-log	80	timber	136	log4jna
24	cronolog	81	hugo	137	slf4j
25	centralized-logging	82	frodo	138	slf4j-api
26	graylog	83	loglifecycle	139	jul-to-slf4j
27	mcelog	84	debugoverlay	140	slf4j-android
28	nslog	85	lynx	141	logback-groovy
29	nxlog	86	androlog	142	logstash-logback-encoder
30	logserver	87	slf4jandroid	143	logback-classic
31	graylog2	88	SLF4J-Android	144	log4c
32	rsyslog	89	Log4JAndroid	145	log4cpp
33	syslog-ng	90	Log4J-Android	146	logstash
34	syslogd	91	Android Logger	147	hyperloglog
35	eventlog-source	92	LOGBack	148	loglog
36	log-analysis	93	boost	149	xcglogger
37	iis-logs	94	boost-log	150	hg-log
38	event-log	95	boost-logger	151	log4net-configuration
39	exception-logging	96	boost::log	152	log4net-appender
40	transactionloganalysis	97	enterprise-library	153	log4net-filter
41	logged	98	enterprise-library-5	154	transactionlog
42	truncate-log	99	enterprise-library-6	155	winlogon
43	java.util.logging	100	error-logging	156	jsnlog
44	dyalog	101	apache-commons-logging	157	nlog-configuration
45	git-log	102	audit-logging	158	jxloginpane
46	logcat	103	google-cloud-logging	159	timber-android
47	log4php	104	common-logging	160	enterprise-library-4.1
48	log-viewer	105	jboss-logging	161	logback-android
49	loginfo	106	semantic-logging	162	boost.log

50	monolog	107	logging-application-block	163	spdlog
51	unified-log	108	django-logging	164	loglevel
52	general-log	109	ms-extension-logging	165	log-level
53	auth.log	110	scala-logging	166	hyperloglog.java
54	Log4J	111	iis-advanced-logging	167	hyperloglog.python
55	SLF4J	112	libcomponentlogging	168	logsene-android
56	tinylog	113	entlib-logging	169	logsene-js
57	logback				

B STATISTICS DETAILS OF RQ 1

Label	AE	DB	SO	SE	SF	SU
Total Questions	183	1131	53532	177	4215	1275
Questions with Accepted Answers	50	555	26503	102	2157	541
Percentage	27.32240437	49.07161804	49.50870507	57.62711864	51.17437722	42.43137255
Views Median	574	368	349	444	565	538
Views Quartiles (Q1, Q3)	(115.5, 2431.0)	(102.0, 1453.5)	(96.0, 1248.0)	(176.0, 1449.0)	(175.0, 1880.5)	(111.0, 2354.5)
Views Max Min (Min, Max)	(9.0, 5629.0)	(11.0, 3474.0)	(3.0, 2976.0)	(28.0, 3332.0)	(3.0, 4426.0)	(6.0, 5697.0)
Answer Count Median	1	1	1	2	1	1
Answer Count Quartiles (Q1, Q3)	(0.0, 1.5)	(1.0, 2.0)	(1.0, 2.0)	(1.0, 3.0)	(1.0, 2.0)	(1.0, 2.0)
Answer Count Max Min (Min, Max)	(0.0, 3.0)	(0.0, 3.0)	(0.0, 3.0)	(0.0, 6.0)	(0.0, 3.0)	(0.0, 3.0)

TABLE B2 RQ 1 details