An Empirical Study of Release Note Production and Usage in Practice

Tingting Bi, Xin Xia, David Lo, John Grundy and Thomas Zimmermann

Abstract—The release note is one of the most important software artifacts that serves as a communication bridge between development teams and users. Release notes contain a set of crucial information, such as descriptions of enhancements, improvements, potential issues, development, evolution, testing, and maintenance of projects throughout the whole development life cycle. A comprehensive understanding of the characteristics of release notes and how to best document one for different targeted users would be highly beneficial. However, the release note is often neglected and has not to date been systematically investigated by researchers. In this paper, we conducted a descriptive case study to investigate release note production and usage in practice. We first performed a large scale empirical study of 32,425 release notes in 1,000 GitHub projects to understand the characteristics of real-world release notes, and eight categories of information identified that are normally documented in release notes. We then conducted interviews with 15 professionals and an online survey with 314 respondents to investigate their opinions on release notes in practice. Our results show that both release note producers and users consider that well-formed release notes impact software activities (e.g., software evolution) positively. We summarised 27 statements about release notes grouped into eight topics based on participants' opinions. Our study uncovers significant discrepancies between release note producers and users in perceiving release notes. Based on these findings, we provide a set of release note production and usage guidelines for practitioners and highlight future research directions.

Index Terms—Release Note, Software Documentation, Empirical Study

1 Introduction

Release engineering focuses on building a pipeline to transform project requirements into an integrated, compiled, packed, and tested product [1]. In this process, the *release note* is an essential artifact that informs users about vital project changes from one version to another [2]. It serves as a critical means of interaction between development teams and users. The information in release notes (e.g., significant enhancements, changes, improvements, and risks) typically is related to a set of software activities spanning the whole development life cycle [3]. A comprehensive understanding of release notes (e.g., who produces release notes, who uses release notes, and the structure and information in release notes) would be helpful to guide more effective production and usage and better support related software activities [4] [5].

Release notes can be documented in different ways based on various projects, as there is no accepted standardization for documenting release notes [6]. Each release note should be targeted for different users who can effectively and efficiently take advantage of the information [7] [8]. For example, documenting new features or changes of systems in release notes helps keep end-users informed on what is new in a version; however, it is challenging to keep the release notes balanced between being sufficiently informative and not too lengthy. In addition, producing release notes is also a collaborative task and varies depending on the different stakeholders involved [9].

In this work, the definition we are using for a "release note" is a formal document distributed with software projects and delivered to users when an update is released. Release note information may vary greatly in focus and detail, depending on the software application domains, release note producers, and target release note users. We target release notes in GitHub projects that describe key changes made to features and the code in a release, with information intended for use by developers and a few end-users. However, compared to other formal software artifacts, such as requirement documents and the source code, or informal software artifacts, such as developer mailing lists [10]), release notes in GitHub projects have not yet been systematically investigated. Moreover, even though documenting release notes is critical for software development, development teams and users often neglect them [11]. There is limited evidence that captures and shares information via release notes in real-world organizations. A vague and imprecise release note will likely lead to inadequate considerations of current development. Incomplete or incorrect information in release notes will be problematic to users who need to accurately follow project development details. Producing quality release notes in software development is thus a critical task.

Existing works have only investigated release notes on a small scale and do not present a comprehensive understanding of the characteristics of release notes in practice [12] [3], and some studies did not involve realistic opinions

- *Tingting Bi, Xin Xia, and John Grundy are with the Faculty of Information Technology, Monash University, Melbourne, Australia. E-mail: {tingting.bi, xin.xia, john.grundy}@monash.edu*
- *David Lo is with the School of Information Systems, Singapore Management University, Singapore. E-mail: {davidlo}@smu.edu.sg*
- *Thomas Zimmermann is with Microsoft Research, Redmond, WA, USA. E-mail: {tzimmer}@microsoft.com*
- *Xin Xia is the corresponding author.*

Manuscript received ;

of diverse stakeholders towards understanding on release notes [6] [13]. Some key issues are yet not investigated, including:

- **Poor understanding of release notes**: the process of producing release notes has not yet been systematically investigated. For example, who produces release notes, and what information should be included in a release note regarding different software domains.
- **Discrepancies between release producers and users in perceiving release notes**: opinions of release note producers and users might be different. Release notes are not only for end-users but also contain rich information on design decisions and detailed project changes, which can facilitate internal developers to conduct various software activities better [14]. As such, incomplete input and specifications from internal users can result in inadequate or failed releases for project updates [15] [16].
- **Improperly linking release notes to other software artifacts and activities**: release notes contain a set of information on a project closely related to requirements changes, design decisions, and risk assessments. However, release notes are not sufficiently related to other software artifacts (e.g., requirements, design rationale, and bug reports) and software activities (e.g., testing). One potential reason is the lack of effective release note documentation and management ways.

These issues motivated us to investigate release note production and usage in practice systematically. Analysis and characterization of release notes would shed light on release notes production and usage effectively in practice.

- A large scale empirical study: we analyzed and characterized 32,425 release notes and relevant data of 1,000 GitHub projects. For example, the number of release note producers and the information contained in release notes regarding different software domains. The results show that the information in release notes can be classified into eight main categories, which span the whole software development cycle. The most frequently documented information is related to **Issues fixed** and **New features**.
- Interviews and one online survey with professionals: we gathered professionals' opinions on release notes. For example, testers perceive release notes as a rich source of information to test software projects, whereas architects picture release notes as key milestones about the evolution of projects. We identified that there are significant discrepancies between release note producers and users. For instance, most release note producers perceive non-functional requirements are important for documenting, but users expect new feature related information to be well documented.
- Associated release notes with software activities: we investigated how the information in release notes impacts various software activities (e.g., software evolution). We also summarized the factors that would affect release note usage in practice. For example, projects that adopt agile development methods often do not produce release notes properly, or the release notes are not consistent and low quality.

To make the process and results of this study easy to follow, we classified "stakeholders" into two distinct types: **Release Note Producer** and **Release Note User**. For example, the internal stakeholders, such as testers and developers of projects, are identified as "Release Note Users" if they use release notes of their own projects but do not create them. The key contributions of this work include:

- A better understanding of the characteristics and documented information in release notes.
- Key differences in perspectives of release notes between release note producers and users in practice.
- A catalog of eight empirically-justified topics from professionals that can help organizations and practitioners produce or use release notes effectively.
- A set of guidelines for effectively producing and using release notes, including what information should be included in release notes regarding different software domains and activities.

The rest of this paper is organized as follows: Section 2 describes key related works and compares those works on release note investigation to our work. Section 3 presents the details of the case study setup process. Section 4 shows the results of this study. Section 5 discusses and explains the results and implications of our study. Section 6 discusses the threats to validity. Finally, Section 7 concludes this study and outlines key future research directions.

2 Related work

Several research studies propose approaches and tools to produce and use release notes. In this section, we compare our work with other works from two aspects: (1) production and usage of release notes in practice; and (2) the key research gaps that our work aims to address.

2.1 Production and usage of release notes in practice

Moreno *et al*. [3] proposed a method (i.e., ARENA) to generate release notes automatically. ARENA extracts changes from the source code, and then the authors summarized and integrated these code changes with information extracted from software repositories to generate complete release notes. Four empirical studies were then performed to evaluate the performance of ARENA regarding completeness, importance, and usefulness of the generated release notes. The results show that ARENA-generated release notes are good approximations of the ones manually produced by developers. The authors also conducted an empirical study that targeted the analysis of the contents and characteristics of 990 release notes belonging to 55 open source projects. Their results show that 17 types of information are documented in release notes. The most frequent item included in a release note is related to bug fixes. The process and results of their empirical study are similar to the "What" aspect of Stage 1 (see Section 4.1.2) in this work. However, our empirical study analyzed the information in release notes

regarding different software domains and at a much larger scale. We also analyzed the granularity of the information (e.g., at the class level) in release notes for major and minor releases. In addition, we provided an analysis of release notes systematically from the "Who" aspect to better understand the characteristics of release notes (see the results in Section 4.1.1).

Abebe *et al.* [12] presented an empirical study to investigate what information is documented in release notes. The authors manually analyzed 85 release notes across 15 different software systems. Their results show that six information types are identified in release notes: title, system overview, resource requirements, installation, address issues, and caveats. In addition, nine different factors are found to better understand the likelihood of an issue being listed in the release notes. The authors then applied machine learning techniques to predict and suggest the issues listed in release notes automatically. The authors identified the information types provided in major and minor release notes that differ from system to system. This idea motivated us to investigate release notes of major and minor releases. Their work observed that release notes are mainly written for users who would not read through the source code of the software systems, and only 30% of the release notes have additional information for developers. In contrast, as the release notes we selected were from GitHub projects, 71.5% release notes of our analysis are developer-targeted.

Several empirical studies have conducted software documentation and software history investigation, and the results show that release note is an important artifact and plays a vital role in various software activities. For example, Tsay *et al.* [6] built a release history database of release notes for getting a broad picture of the open-source landscape. The authors standardized and inserted data into a release note database, which can help researchers and developers study and model the release engineering process in greater depth. Yu [17] conducted a keyword-based approach to mine logs and release notes to extract useful software maintenance and evolution information. For each changelog and release note, the author defined a mathematical framework to present, interpret, and associate the data with bug-fixing activities. Then the author incorporated or updated activities in the development of the new version. The results show that applying keyword-based text mining techniques to mine the changes described in release notes can help developers conduct maintenance and evolution activities. Codoban *et al.* [18] presented an empirical study on motivations and reasons that developers have of examining software history. The authors firstly conducted interviews with 14 developers. They then deployed a survey to quantify and extent the interview findings. The findings show that software history tools are ill-suited and can not provide explicit support for developers' needs from the history of releases. Fischer *et al.* [19] proposed an approach to build a SQL database of release notes that combines versions and bug report data. The approach can be used for building a software evolution analysis framework. Michlmayr *et al.* [13] performed an exploratory study to investigate the understanding of releases in practice and release management problems in Free and Open Source Software (FOSS). The authors interviewed 20 developers from various projects, and the interviews include a set of questions about release management. The results show that release managers in small and large projects play a vastly different role even when they essentially have the same responsibility.

2.2 Research gaps

While the research listed in Section 2.1 can be valuable for understanding release notes in practice, some gaps exist that have not yet been investigated in the literature. A comprehensive understanding of the characteristics of release notes and real opinions from professionals would provide better and more tailored support for producing and using release notes. We try to fill some of these gaps in this work. Table 1 compares the key-related works with our work from four perspectives:

- **Methodology** – what methods have the previous key research studies used to investigate release notes as compared to our work.
- **Results** – what were the goals and achievements of the previous key research studies compared to ours related to release note investigation.
- **Software artifacts** – what artifacts related to release notes were analyzed in the studies.
- **Software activities** – what were the relevant software activities that could benefit from better documentation of release notes. For example, software maintenance is an activity of modifying the project after it has been delivered to the customers for improving the project. Software evolution is triggered by chaining customer and user requirements [20].

3 Study Design

We first describe the process of releasing and producing release notes in GitHub projects in Section 3.1. To understand release note production and usage in practice, we then specify the goals and research questions in Section 3.2. The precise units of this study are presented in Section 3.3.

3.1 The process of releasing and producing release notes in GitHub projects

In this study, we focus on investigating release notes in open source projects, and we particularly target GitHub, which is one of the most important development platforms. Accordingly, the release notes we collected are mostly delivered to other software developers and some to end-users. GitHub allows users to release and manage a repository that includes creating, editing, and deleting a release along with an associated release note and binary links[1]. Despite the features and functions provided by GitHub, some projects may also follow their own specialized process for releasing and producing release notes. For example, the Visual Studio Code project not only updates the GitHub repository, but also uses a dedicated Wiki to manage the iteration plans of released projects[2]. However, in this study, we only focused on the release notes documented in GitHub projects, and

1. https://desktop.github.com/release-notes/
2. https://github.com/microsoft/vscode/wiki/Iteration-Plans

TABLE 1
Comparison of the key works with our work.

Work	Result	Methodology	Software artifact	Software activity
Moreno *et al.* [3]	(1) Generate release note automatically; (2) Investigate the information contained in 990 release notes.	(1) Apply tools to extract changes of source code and libraries, extract the textual information of repositories, and combine the changes and textual information to general release notes; (2) Empirical studies to evaluate the performance of these generated release notes.	990 release notes, source codes, libraries, documents related to architecture, requirements, and licenses	Software documentation and software evolution
Abebe *et al.* [12]	(1) Identify the information in 85 release notes of 15 projects and suggest (predict) the issues that should be listed in the release notes; (2) Identify the difference in release note contents between major and minor releases.	(1) Manually analyze the 85 release notes; (2) Machine learning to predict the issues that should be listed in release notes.	85 release notes	Software maintenance
Fischer *et al.* [19]	Extract bugs and release data to analyze software evolution.	Manually construct a Release History Database.	Bug reports and source code	Software evolution
Phillips *et al.* [21]	Extract information to understand group decision-making processes by analyzing release notes.	Semi-structured interviews.	Source code	Traceability and software evolution
Klepper *et al.* [22]	Generate targeted and informative release notes.	Apply a stakeholder targeted delivery model.	Requirement documents	Software maintenance
Our work	(1) Investigate the characteristics of release notes and their contents in real-world release notes of GitHub projects (i.e., from "who" and "what" aspects); (2) Understand how do release note producers and users perceive release notes, and what gaps exist between them.	A large scale empirical study, interviews, and online surveys.	(1) 32,425 release notes of GitHub projects, other relevant information (i.e., domain of the projects and timing of release note documentation); (2) Qualitative data (i.e., textual and audio information) collected in the interviews and surveys	Software development, design, testing, evolution, traceability, and software documentation

any referenced data sources and outside links were not studied.

Developers collaborate in developing projects and contributing software artifacts [23], but the collaboration approaches may vary from project to project. Before we conducted this study, we exchanged dozens of emails with several developers from different GitHub projects to discuss the process of releasing and producing release notes of their projects. In the following section, we describe the *general* process of releasing and producing release notes in GitHub projects. We present an example, which includes the key information about a new release and its release note of the Visual Studio Code project in GitHub.

- As shown in Fig. 1, one of the contributors, i.e., annotation (1), is in charge of releasing new versions and its release notes (the main release note producer).
- As shown in Fig. 2, the new releases and its release notes are based on the submitted commits, i.e., annotation (2). For example, there are 1,440 commits contributed to V1.39 and its release notes. Each contributor, i.e., annotation (3), is in charge of summarizing their own "release notes" to report the features and changes they have built. Then the main release note producer collects information that the other contributors summarized to produce the formal release notes.

In summary, the general process of producing release notes in GitHub is that each contributor is in charge of producing their release notes (based one the commits), and one of contributors collects all relevant information to produce a formal release note. We term the contributors who release new versions and release notes as the **release note producers** (i.e., annotation (1) and (3)).

3.2 Goals and research questions

Release notes should be periodically delivered to users to inform them about incremental updates of software projects. However, even though the release note is an important software artifact, there is a lack of a widely accepted and systematic understanding of what a release note is. To get a broad oversight on release notes, we prepared a research protocol to define key research questions. We initially identified seven concerns about release note production and usage in practice (see Table 2). These concerns include several primary considerations, e.g., what information is typically documented in release notes.

We decided to conduct a descriptive case study [24] that uses mixed qualitative and quantitative approaches to sys-

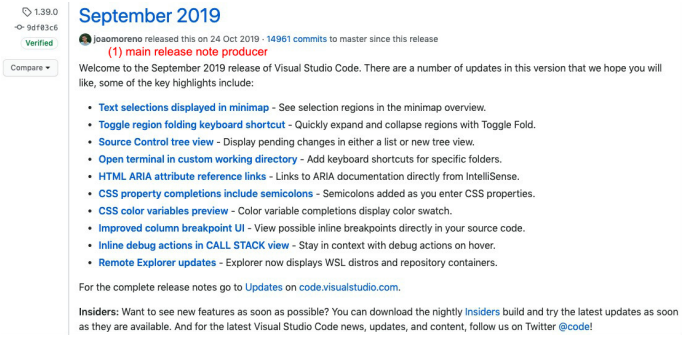

Fig. 1. V1.39 and its release note of the Visual Studio Code project in GitHub. Annotation (1) indicates the main producer of the release and the release note.

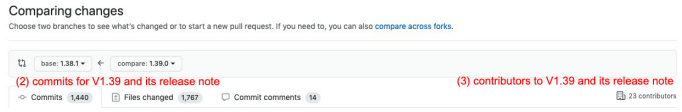

Fig. 2. Commits and contributors for V1.39 and its release note. Annotation (2) shows the number of commits related to V1.39 and the release note. Annotation (3) shows the contributors to V1.39 and the release note.

tematically analyze and portray release notes. We applied the Goal/Question/Metric (GQM) approach [25], which focuses on specific goals, and traced the goals from analyzing the data, and finally provide a framework for interpreting the data in terms of the goals. To be specific, we **analyze** and **characterize** the information in release notes of GitHub projects, and we interview and survey the opinions from the professionals who produce and use release notes. This is ***for the purpose of*** understanding the characteristics of release notes ***from the point of view of*** release note producers and users ***in the context of*** release note production and usage in practice. The metrics applied (for each RQ) in this work are listed in Table 9 to 11 of Appendix.

Based on seven key concerns summarised in Table 2, we defined two main RQs, and the answers to the RQs can be mapped to the goals of this study. A comprehensive understanding of release notes in practice, including who is involved in the release note production and what information is included in release notes (RQ1). What considerations professionals have of release notes (RQ2.1). What discrepancies exist between release note producers and users on perceiving release notes, and how might we address the discrepancies (RQ2.2).

RQ1: Who produces release notes, and what information is included in release notes? - (C1, C2, and C3)

RQ2: How do release note producers and users perceive release notes? - (C1 to C7)

RQ2.1: What are the typical considerations release note producers and users have of release notes?

RQ2.2: What are significant discrepancies between release note producers and users in perceiving release notes?

TABLE 2
Concerns about release note production and usage in practice.

ID	Concerns about release note production and usage
C1	What information is included in release notes, and how are they structured?
C2	Who produces release notes?
C3	Who uses release notes?
C4	What software activities benefit from the information contained in release notes?
C5	What are the relationships between release notes and other software artifacts?
C6	What are the limitations of release note production and usage in practice?
C7	Do release notes effectively help users during software development in practice?

3.3 Case study approach and process

To answer the RQs listed in Section 3.2, we conducted a descriptive case study to investigate release notes. An overview of the process in this case study is shown in Fig. 3. The descriptive case study[3] consists of three stages. ***Stage 1***: A large scale empirical study on release notes of GitHub projects (in Section 3.3.1); ***Stage 2***: Interviews with professionals on how do they perceive release notes in practice (in Section 3.3.2); and ***Stage 3***: An online survey for confirming and extending the conclusions about release notes derived from ***Stage 1*** and ***Stage 2*** (in Section 3.3.3).

3.3.1 Stage 1: A large scale empirical study on release notes

One of the key goals of this study is to understand the characteristics of release notes (i.e., RQ1). We targeted open-source projects and collected release notes from GitHub-hosted projects.

Data collection: To make sure that the GitHub projects we investigated are non-trivial, we defined three criteria for project selection: (1) the project was launched at least two years ago and is active (i.e., the software is working and the repository of the project is continually being updated); (2) the number of release notes in the project is more than 20; and (3) the project is contributed to by more than 20 contributors (i.e., committers). We used the search function provided by GitHub to target the projects with 6,000 or more stars, and the number of stars of a repository works like an easily accessible and reliable proxy to its popularity [26]. We got 3,142 projects and randomly selected 1,000 projects which meet the selection criteria. We did not include the projects in the Documentation domain, as this domain generally corresponds to books, tutorials, and source code examples (e.g., examples of Java design patterns). In addition, we excluded the projects, which follow a same pattern to generate release notes to ensure the data we collected are valid (i.e., the release notes are not potentially produced by bots[4]).

We finally collected 32,425 release notes and other the related information of the selected projects. The collected data items of Stage 1 are shown in Table 9 of Appendix.

3. The ethics approval number of this work from Monash University is 23040-40257.
4. https://github.com/apps/conventional-release-bot

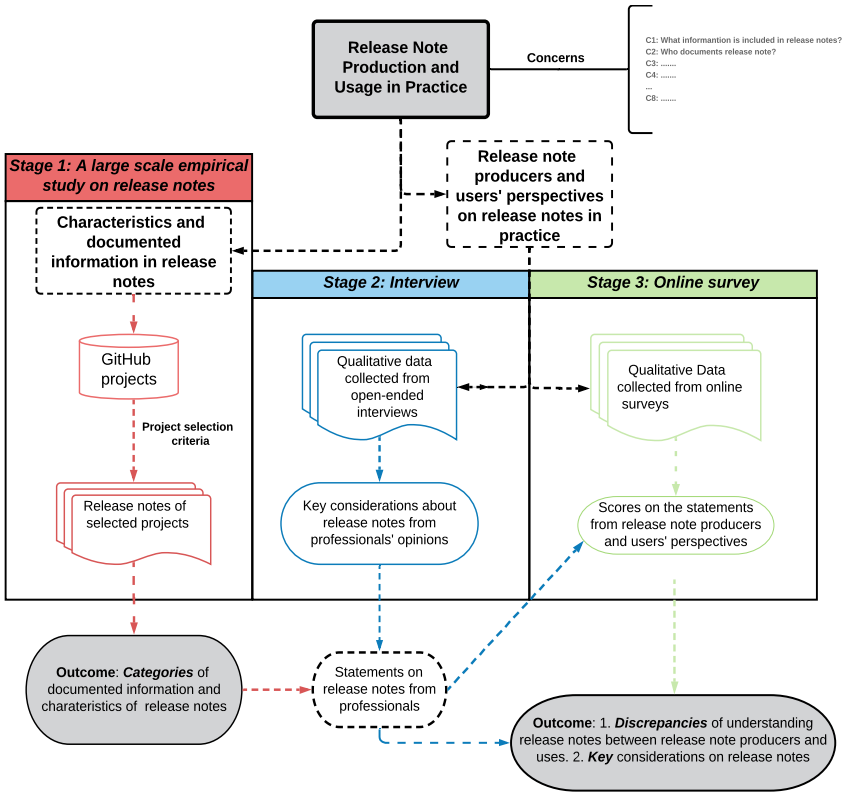

Fig. 3. Overview of the methodology.

Data analysis: As shown in Table 9, we used descriptive statistics to analyze the quantitative data items (i.e., D1-D3, D5, and D6). We analyzed data item D4 to classify the selected projects into five domains. The detailed data analysis of D4 is shown in Section 4.1.2. We applied a bottom-up method (i.e., Constant Comparison) to analyze the qualitative data (i.e., D6) to generate the topics of information documented in release notes. Bottom-up approaches are suitable for specific domain knowledge concepts when there are no predefined and existing concepts in that domain [27].

We first performed a pilot data labeling exercise by randomly selecting 150 release notes from 10 projects to mitigate personal bias in the labeling. In the formal data labeling, we invited another Ph.D. candidate, who specializes in software engineering, to label the information documented in release notes. All the collected release notes were manually labeled by two annotators (i.e., the first author and the Ph.D. candidate). Any disagreements on a labeled release note were discussed and confirmed with the second author. To facilitate the manual labeling, we used MAXQDA[5] (a tool for qualitative data analysis) to label the information of release notes into categories. The process of Constant Comparison comprises two steps: (1) *Open coding* splits the qualitative data into categories. The first author and the Ph.D. candidate separately coded the same release notes of the selected 500 projects to generate categories, and then they checked the categories that were coded by each other. We identified the categories through refinement and integration of the concepts generated in the open coding. (2) *Axis coding* was then executed by the first author and the Ph.D. candidate. They individually coded half of the release notes of remaining projects. All coding results were then confirmed by the second author. By the end of the labeling, we made a final reliability test and calculated Cohen's kappa reliability coefficient for categorizing the release notes between the two annotators, and the value is 0.81. This indicates a strong agreement between the two annotators. The coding results are shown in Table 5 of Section 4.1.2.

Among the selected 1,000 projects, it is possible that many projects use their specified ways (outside GitHub) for documenting release notes (see Section 3.1). Consequently, the release notes we collected from these projects (in GitHub) might contain incomplete information. To ensure the validity of the collected data, we conducted another round of manual check on the selected projects and their release notes. Specifically, we analyzed a representative sample size, and we used a sample size calculator to calculate how many projects should be checked. We set the error margin as 3% and the confidence level as 95% [28]. The calculation result shows that we should randomly select and check 516 out of the 1,000 projects and their release notes. The result of the manual check shows that 9 projects (i.e., 1.7% of the selected projects) explicitly mention that they document official release notes outside of GitHub. In addition, we compared the release notes documented in GitHub with official release notes documented outside GitHub for the same release, and we noticed that official release notes are documented with more detailed descriptions and rationale for key changes. However, in this work, we

5. https://www.maxqda.com/

classified the information in release notes into a relatively higher level (see Table 5). Based on the above, we believe our findings that are derived only from analysis of information on GitHub remains valid.

3.3.2 Stage 2: Interviews

The purpose of the interview was to understand how professionals perceive release notes. This section presents the process of the interview.

Data collection: We prepared a set of questions [29] to interview professionals: (1) Demographic questions about the background and the work experience of the interviewees; (2) 15 specific questions extended from the seven concerns (listed in Table 2) to interview professionals about their opinions and considerations on release note production and usage in practice. These questions cover a set of software activities, including software design, coding, and maintenance. We also discussed the results of Stage 1 with the interviewees.

We invited professionals from our networks in the software industry who are working full time in different roles and functions (e.g., developers and architects) to participate in the interviews. We used the snowballing sampling method to invite professionals who participated and willing to take an interview from their contacts to get us more samples [30]. We sent 20 formal invitations to invite potential interviewees, and 15 interviewees finally agreed to participate in the interviews from ten IT companies in different software domains worldwide. During the interviews, we asked their experience in release note production and usage, and the interviewees identified themselves as release note producers or users based on their work experience. A summary of the background information of the interviewees is shown in Table 14 of Appendix. After conducting all the interviews, we transcribed the audio answers of each interview into a text file. The detailed extracted data items are presented in Table 10 of Appendix.

Data analysis: Similar to Stage 1, we applied descriptive statistics to analyze the quantitative data item (i.e., D8) and Constant Comparison method to analyze qualitative data to generate topics and statements of the interview contents (i.e., D8). The data analysis methods are shown in Table 10 of Appendix. To reduce personal bias in qualitative data analysis of interview contents, the first author and the invited Ph.D. candidate separately analyzed and coded the qualitative data to generate the topics and statements of the interview contents. Any inconsistencies in the coding results were cross-examined by multiple authors. After completing the data analysis and labeling process, multiple authors discussed disagreements to reach a common consensus. The overall Kappa value of the data analysis is 0.82. Eventually, based on the results of Stage 1 and the interviews, we got 27 statements grouped into eight topics shown in Table 8 of Section 4.2.1.

3.3.3 Stage 3: Online survey

According to the guidelines for selecting empirical methods in software engineering research [31] [32], surveys aim to *"collect quantitative but subjective data and objective data such as demographic information, for example, a subject's age and educational level*." The purpose of the online survey in this work was to confirm the results of Stage 1 and the statements made by the interviewees (i.e., Stage 2) with more participants.

Data extraction: We adopted Non-Probabilistic Sampling methods [33], i.e., convenience sampling and snowball sampling, to invite participants. The target participants of the online survey were those who have experience producing and using release notes in practice. We followed two steps to invite participants:

- We invited potential participants from the selected GitHub projects of Stage 1 who are from various countries worldwide. We received 108 responses out of 1,715 sent emails (i.e., 6.3%).
- We invited professionals within our social network from various countries and IT companies and asked their help to disseminate our survey. The IT companies include Microsoft, Alibaba, Baidu, Google, Huawei, Hengtian, and other small to large companies worldwide. By following this strategy, we got 206 responses out of 1,000 emails sent.

In total, we received 314 valid survey responses. The participants' professional experience varies from 0.5 to 14 years, with an average of 5.8 years. The top two countries where the participants reside are China and Australia. Participants were expected to score statements (see Table 8) about release notes according to the "*Agreement Level*" (Strong Agree, Agree, Neutral, Disagree, and Strong Disagree).

The use of Likert scales to construct surveys has a long history [34]. The Likert scales are intended to map individuals' perceptions and attitudes, which are inherently difficult to measure. However, the use of the Likert scales leave researchers unsure about whether some participants select the most time-efficient opinion. In this study, the "Neutral" option in the online survey could be the most time-efficient option for participants, i.e., they do not need to commit an opinion one way or the other for a question. However, we wanted respondents to have an option to express that they did not have a preference or view regarding a particular statement.

We asked participants about their roles and functions in development, and we started the online survey with an optional question "*Are you a release note producer or users*?" that divides the questionnaire into two sections for **Release Note Producer** and **Release Note user**. The participants can also provide comments and rationale supporting their options. Our questionnaire can be found in the following links[6].

Data analysis: The analyzed data items are shown in Table 11 of Appendix. We analyzed the distributions of responses (in the Likert scale) from the participants (i.e., D9 and D10) and compared the distributions of the two groups (i.e., Release Note Producers vs. Release Note users) using Effect Size. The full results of the online surveys are shown in Section 4.2.2. We analyzed comments (i.e., D11) and described some of them in Section 4.2.2.

6. https://forms.gle/xULKydpakT1Kc59T8 (English version) https://www.wjx.cn/jq/54361171.aspx (Chinese version)

TABLE 3
Statistical information of the selected GitHub projects.

Statistic	Mean	Min	Max
project size	3,786,517	1,276,812	9,876,812
project star	7,198	6,003	15,909
team size	220	98	1,902

TABLE 4
Distribution of release notes in the selected projects.

Number of release notes	Number of projects
20-100	238
100-200	276
200-300	471
300-400	15

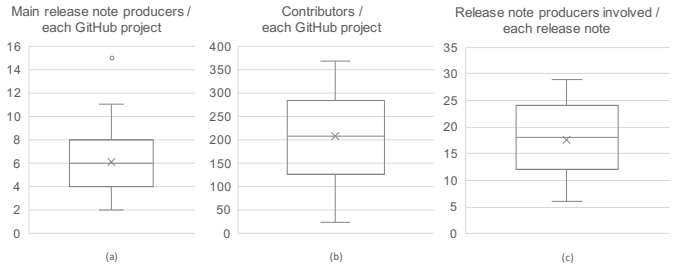

Fig. 4. Distribution of release note producers. We report box-plots to show the min, max, and average number of release note producers and contributors of each project.

4 RESULTS

We describe the results of the large scale empirical study (to answer to RQ1) in Section 4.1. The results of professionals' opinions on release notes are shown in Section 4.2.1 (to answer RQ2.1). The discrepancies between release note producer and users are presented in Section 4.2.2 (to answer RQ2.2).

4.1 Results of RQ1: Who produces release notes, and what information is included in release notes?

We report the results of RQ1 in this section. For statistical information about the 1,000 selected GitHub projects, we recorded: (1) Project size (i.e., the number of source code lines); (2) Popularity (i.e., stars of projects); and (3) Team size (i.e., how many contributors to the projects). We present the detailed results in Table 3. In addition, for the collected release notes, we made a tally of the distribution regarding release notes in the GitHub projects, and results show that the collected release notes are distributed (i.e., the collected release notes are not only from a few projects), for example, 238 projects have 20 - 100 release notes (see Table 4). We then extracted and analyzed the relevant information to answer RQ1 from two perspectives: the "**Who**" and "**What**", respectively.

4.1.1 Who produces release notes

As we described in Section 3.1, for a new release, each contributor documents (i.e., release note producers) their own "release notes", and then, one of the contributors collects individual's release notes for producing a formal release note.

Investigating the statistical information of how many producers are involved in contributing to release notes could help understand the process of release note production. We counted the number of the contributors (i.e., committers) of each release and their commits. The statistical results are summarised in Fig. 4. These indicate:

1) For each GitHub project, on average, there are six (6) **main release note producers** who lead collecting information for release note production (e.g., the annotation (1) of Fig. 1). The main release note producers vary between releases (see Fig. 4 (a)). 17 release note producers (e.g., annotation (3) of Fig. 1), on average, are involved in contributing to a release and its release note (see Fig. 4 (c)); 220 contributors, on average, are involved in contributing to a GitHub project (see Fig. 4 (b)).
2) In our sample, we labeled 4,343 release notes for major releases. If a release is tagged as a major release, we labeled its release note as a major release note. For the releases without any tags, we labeled their release notes as minor. 37 commits, on average, contribute a major release and its release note. 8 commits, on average, contribute to a minor release and its release note of each project.
3) The average number of commits that the **main release note producers** submitted is 97, which is 4.4 times higher than the number of commits that other release note producers (18 commits) submit for each project. One potential reason is that the main release note producers are core contributors (e.g., architects) of the projects. They are in charge of driving the projects forward. Identifying main release note producers and building an effective personalization strategy (i.e., establishing a collaborative network between core release note producers and other contributors) could be beneficial for information sharing, software development, and artifact documentation.

As release notes comprise the information related to new features, changes, and issues spanning the whole development life cycle, it is a vital artifact to development. The quality of release notes depends, among others, on its producers. The statistical results above could help to understand the process of release note production and some characteristics of release notes (i.e., who produces release notes of GitHub projects). However, it is unclear how release note producers choose the features and changes they use to summarize a release note adequately. It is unclear how the main release note producers document the release notes by selecting and grouping the critical information for release note users. These uncertainties motivated us to interview professionals and conduct surveys to better understand their opinions and experience of release note production and usage.

4.1.2 What information is included in release notes

Documenting release notes is a process to accommodate the domain-specific characteristics of the project. The informa-

TABLE 5
The categories of the documented information in release notes.

Category	Description	Example	Percentage
Issues fixed	Information about what issues have been fixed.	*Example 1: "Fix possible ESP8285 flash problem by updating Flash Chip Mode to DOUT during web upload" - web fixed* *Example 2: "Fix boot loop when selecting module Sonoff 4CH or Sonoff Touch on non ESP8285 hardware" - hardware fixed* *Example 3: "Fix client not working caused by reconnecting" - network fixed* *Example 4: "Fix firewalls for OpenStack" - firewall issue fixed* *Example 5*: *"Fix the HTTP response code when downloading URLs" - broken link fixed* *Example 6*: "*Revert fix connection issues with withings API by switching to a maintained codebase*" - *API issues fixed* *Example 7*: "*Compiler: handle negative length in code frame repeat*" - *compiler issues fixed*	79.3%
New features	Information about new features and functions that are newly added into the systems.	*Example*: *"The v1.1 release brings many new features since the last stable one. It is focused on two major themes, which are showing workload resources and user experience redesign"*	55.1%
System internal changes	Information about internal changes of the systems, including changes of high-level design (e.g., component changes), low-level design (i.e., methods/instance variables/deprecated class added, renamed, and removed), and other changes (i.e., libraries/APIs).	*Example 1*: *"Not that when use as a route component in vue-router, these properties will be ignored because async components are resolved upfront before the route navigation happens. You also need to update vue-router to 2.4.0+ if you wish yo use the new syntax for route components" - Component level changes* *Example 2*: "*Remove the unnecessary newline and unused vars*" - *Event removal* *Example 3*: "*Added the third parameter column for Table's row click event*" - *event addition*	25.1%
Non-functional requirements	Information about a set of quality attributes of systems (for example, security and performance).	*Example 1*: *"Significant kube-proxy performance improvements for non UDP ports" - non functional requirement improvement* *Example 2*: *This release will be supported from ESP8266/Arduino library core version pre-2.6.0 due to reported security and stability on previous core version - non functional requirement issues solving* *Example 3*: *"Vue congig. performance now defaults to false due to this impact on dev mode performance. Only turn it on when you need it" - potential problem of non-functional requirements*	10.3%
Documentation update	Information about updated software artifacts, e.g., architecture documents.	*Example 1*: *"Significant change: upgrades calico/canal for security vulnerability" - architecture documentation update* *Example 2*: "*Existing Calico user on clusters that were created prior to kops 1.8.0 need to be updated for the new "Default-Deny" behavior for Kubernetes Network Policies*" - *policy document update* *Example 3*: "*Move kops-controller to use a yaml configuration file*" - *configuration documents update* *Example 4*: "Lots of documentation have been polished" - *other software artifact update*	9.5%
Configuration	Information about installation, hardware, and network requirements to run the software.	*Example*: "*For initial configuration this release supports webserver based WifiManager or serial based command interfact only. Support for WPS and SmartConfig has been removed*"	2.8%
Required further actions	Information about further actions if using the software.	*Example*: "*Required action: Please ensure you have backed up you data before upgrading*"	2.1%
Refactoring and reuse	Information about components or the software refactoring.	*Example*: "*Refactor instance group/rolling-update code*"	1.9%

tion contained in release notes is context-dependent for each project and not directly generalized to the entire software [35] [36]. In this work, we only considered release notes of regular releases (i.e., major and minor releases). In contrast, we did not include release notes for beta, pre-release, or alpha releases, as we found those releases normally have no release notes, or the release notes are often very shot and not informative [37].

We analyzed D4 and D6 data items shown in Table 9 of Appendix, and we focus on the correlation between the project domain and the information documented in release notes. GitHub does not include information about the domain of a project, but according to Borges *et al.* [26], the domains of GitHub projects could be classified into six domains (i.e., *Application Software*, *System Software*, *Web Libraries and Frameworks*, *Non-web Libraries and Frameworks*, *Software Tools*, and *Documentation*) [37]. To identify the domains of the selected projects, two annotators (i.e., the first author and the invited Ph.D. candidate) manually checked the domains of the selected projects. Specifically, the first author and the Ph.D. candidate labeled half of the selected projects independently. They then cross-checked the results

that were labeled by each other, and any disagreements were discussed and resolved with the help of the second author. We carried out a reliability test for the domain labeling between two annotators, which is 0.80. This means a strong agreement between the two annotators regarding the domain classification. Please note that we did not include the projects in the Documentation domain, as this domain generally corresponds to books, tutorials, and source code examples (see Section 3.3.1).

As described in Section 3.3.1, we (i.e., the first author and the Ph.D. candidate) classified the 32,425 release notes into eight main categories. Table 5 shows the classified categories, brief descriptions, representative examples, and percentages. Note that a release note can include multiple information categories; as such, the total percentage of all categories is greater than 100%. Information related to **Issues fixed** (i.e., 79.3% of the release notes) and **New features** categories (i.e., 55.1% of the release notes) is the most documented in the release notes. These two categories are significantly larger than other categories. **Issues fixed** category includes several sub-categories (e.g., web fixes and hardware fixes), which describe various tasks during the development. **New features** category describes the new features and functions, which are newly added into the projects. **System internal changes** category ranks the third most common documented information (i.e., 25.1% of the release notes), which includes several sub-categories, e.g., component level changes, event removal, event addition, library, and API changes). **Non-functional requirements** category ranks the fourth (10.3%) most common documented information in release notes, and the most documented non-functional requirements are performance and security related. The **Documentation update** category accounts for 9.5% in all the release notes. This was divided into four sub-categories: architecture documentation update, policy update, configuration files changes, and other software artifacts update. Much smaller numbers of release notes document **Configuration** information, which accounts for 2.8% of the release notes, and this category is about software installation and user notifications. The final categories are **Required further actions** and **Refactoring operation** categories, which account for 2.1% and 1.9% of the release notes, respectively. For **Required further actions** category, we found several release notes of 11 projects that set up a poll to get users' feedback about the new releases. The percentages of categories in Table 5 indicate that the information related to **Issues fixed** and **New features** categories has priorities to be documented in the release notes. We also found that release notes in GitHub projects are quite long, and most of the release notes tend to document a set of issues and bugs that have been fixed.

In addition, we classified and analyzed the information in release notes regarding different domains. The results of this analysis are shown in Table 6. The second column shows the top three documented information categories and their percentages in the five domains. We observe that the most documented information category varies from domain to domain. For example, for Application Software and System Software domains, the most documented information category is related to **New features**. A potential reason is that projects in these two domains provide a range of functionalities for end-users (e.g., MS Word and Google Doc), which need to be frequently changed and documented. The most documented information category in the release notes of Web-libraries and Framework, Non-web Libraries and Framework, and Software Tools domains is all related to **Issues fixed**. A potential reason is that most users of projects in these domains tend to be developers, who are more likely to know lower-level information related to new releases.

TABLE 6
Information categories in release notes of different domains. We reported the top three documented categories that documented in release notes in the five domains.

Domain	Top three categories
Application Software	**New features** - 65.7% Issue fixed- 65.3% System internal changes - 39.4%
System Software	**New features** - 89.7% Issue fixed - 67. 3% System internal changes - 34.5%
Web-libraries and Framework	**Issue fixed** - 65.7% New features- 23.4% Configuration - 7.8%
Non-web Libraries and Framework	**Issues fixed** - 79.7% New features - 65.3% System internal changes - 23.1%
Software Tools	**Issue fixed** - 65.7% System internal changes - 36.3% New features - 31.4%

To investigate the granularity of information in the release notes at different abstraction levels for the major and minor releases, we analyzed the 4,343 release notes of major releases and randomly selected 4,050 release notes of minor releases. We then coded the information into four types: "System level", "Package level", "Class level", and "Cannot be specified". The results are shown in Table 7. The results indicate that information at "Class level" (41.5%) is the most documented in the minor releases, e.g., information variables and objects. Information is classified into the package level is related to the communications between packages or the changes to packages. The information is classified into "System level" (40.5%) is mostly in major releases, for example, information about architecture patterns and models. This is reasonable in that the focus of software design normally starts from the high (e.g., system and package) level and shifts to a lower (e.g., class) level when the software is refined, implemented, and refactored. We also listed one category of information in Table 7 that can not be specified, for example, the method information highly related to project contexts is difficult to classify. In addition, we observe that 71.5% of the information in release notes of GitHub projects is developer-oriented and technical, i.e., those who would read through the source code of projects.

As we discussed in Section 2, some key prior works have investigated and categorized the contents of release notes. For example, Abebe *et al*. identified six different information types that exist in 85 release notes [12]. Their content categories include tiles, an overview of the system, recourse requirement, installation, addressed issues, and caveat. These categories are relatively high-level. The potential reason is that release notes collected from the projects (e.g., Dropbox and Firefox) are targeted more towards the software end-

TABLE 7
Information at the abstraction level in release notes. We reported the results of the documented information at abstraction levels (i.e., system, package, class levels) in release notes for major and minor releases.

Abstraction level	Description	Main release	Minor release
System level	Information documented at the system level or system architecture	40.5%	14.5%
Package level	Information documented at the package level	34.7%	33.2%
Class level	Information documented at the class level, for example, fixed issues of class instances names	13.0%	41.5%
Cannot be specified	Information is closely related to projects that hard to specified (e.g., a method)	11.8%	10.8%

users. We focused on the release notes in GitHub projects that are more likely about the detailed changes about the projects, such as the specific bugs that have been fixed. As such, the categories of their work and ours have significant differences. Moreno *et al.* identified 17 different types (e.g., changes to documents and files) from 990 release notes [3]. Some categories of their work are similar to ours, but their work focuses on a finer level of granularity. The categories they summarised are the foundation for the Change Extractors usage (i.e., capture changes to the source code, libraries, documentation, and license). There are some overlaps between their 17 categories and ours, such as New Features and Issues Fixed. However, we classified the information into a relatively higher level, for example, we classified all changes, including source code changes (e.g., methods and instances changes), libraries changes, architectural changes into the **System internal changes** category (see Table 5).

Regarding our empirical study of analyzing release notes, there are a set of significant differences between our work to these prior works:

- We investigated release notes at a much larger scale (i.e., 32,425 release notes). This provides a more comprehensive understanding of release note characteristics in GitHub projects.
- We investigated the general process of release note production (e.g., how many release note producers) in GitHub projects.
- We analyzed the information in release notes at the abstraction levels for major and minor releases.
- We analyzed the information contained in release note for major and minor releases in different software domains.

Project size, popularity (i.e., stars), and team size of the projects (see Table 3) have an effect on the productivity of software activities [38]. Based on the results of RQ1 (the "who" and "what"), the information contained in release notes vary from project to project, from domain to domain. Our statistical analysis results were based on the projects in the range described in Table 3, and being more aware of the factors impacting release note production and usage, practitioners could produce and exploit more effectively.

Key findings of RQ1
Who: There are many release note producers (i.e., contributors) who document release notes in GitHub projects. There are usually **main release note producers** who play critical roles and are more influential in project development [39]. Building an effective personalization strategy between the core release note producers and other contributors would help information sharing, artifact documentation, and software development.
What: Release notes contain rich and important information, which spans the **whole development life cycle**. The information in release notes for major and minor releases is significantly different, and the information in release notes of various domains is also dissimilar.

4.2 Results of RQ2: How do release note producers and users perceive release notes?

We describe the results of the interviews and the online survey in this section. The results of what common considerations professionals hold of understanding release notes are shown in Section 4.2.1. The discrepancies between release note producers and users in perceiving release notes are presented in Section 4.2.2.

4.2.1 Results of RQ2.1: What are the typical considerations release note producers and users have of release notes?

We conducted interviews with professionals, and we asked them a set of open-ended questions about release note production and usage. During the interviews, eight interviewees identified themselves as primarily release note producers (i.e., I1, I2, I4, I10, I11, I13, I14, and I15); the rest of the interviewees identified themselves as release note users. The detailed background information of the interviewees is listed in Table 14 of Appendix.

We concluded eight topics based on the results of Stage 1 and the interviews, and each topic includes several statements (see Table 8). We linked interviewees' considerations to the survey responses by referring to survey statements. We numbered the statements in order (S1 through S27) and we annotated the statements with statistically significant differences as **❗[Sx]**, which means Statement x has a statistically significant difference between **Release Note Producer** and **Release Note User** that is confirmed by the participants of the online survey.

Topic 1: Human and organizational factors involvement. Human and organizational factors are vital for software development, for example, the project organization, the size of the development team, and challenges in communicating with users. Such factors impact the effectiveness of software activities as well as software documentation. Five (5) interviewees (release note producers) said that producing release notes is normally done by several core contributors to the projects (e.g., architects and project managers) **[S1]**. A consequence of this is that other internal developers might not be aware of the detailed contents of release note documentation and associated development activities **❗[S3]**. Two (2) interviewees (release note producers) mentioned that collecting complete and detailed information of all changes, features, and functionalities is quite challenging

TABLE 8
Survey results. Green cells represent statistically significant differences between *Release Note Producer* and *Release Note User*. Blue cells indicate where Release note producer agrees more. Orange cells indicate where Release note users agree more. The number in the Likert Distribution column indicates the size of each group. The bars in the Likert distributions from left to right are: Strongly Disagree (1 score), Disagree (2 scores), Neutral (3 scores), Agree (4 scores), Strongly Agree (5 scores).

Topic	Statement		Likert Distributions: Producer (98)	Overall Score	Likert Distributions: User (210)	Overall Score	P-values Producers vs. Users	Effect Size Producers Users
T1. Human and organizational factors involvement	Building a personalization strategy could be helpful for release note producing.	S1		4.09		4.10	0.751	-0.01
	Core developers involved in documenting release notes are also critical for software development.	S2		4.14		4.01	0.056	0.13
	Not clear of my responsibilities aligned in documenting release notes.	S3		3.53		3.87	0.000	-0.34
	The effectiveness of producing release notes is slow.	S4		3.71		3.56	0.232	0.15
T2. Tool usage	General management tools to manage and produce release notes are helpful.	S5		4.11		3.93	0.138	0.094
	An automatic tool is needed for producing release notes effectively.	S6		3.45		3.78	0.120	-0.18
T3. Key information and documenting styles	Information related to issues fixed, such as bug fixed information, should be made more clear.	S7		3.20		4.04	0.000	-0.84
	Non-functional requirements are important and needed to included in release notes.	S8		4.04		3.38	0.000	0.66
	Release notes provide the historical guidance of projects (e.g., software evolution).	S9		3.98		3.80	0.148	0.18
	Collecting (searching) key information to produce (use) release notes is timing-consuming.	S10		4.05		3.92	0.266	0.13
	Potential issues (risks) of current projects should be included in release notes.	S11		3.98		3.91	0.266	0.07
	The documentation style (e.g., the length of release notes and users targeted) for release notes is critical.	S12		4.18		4.16	0.947	0.02
T4. Software requirements	Closely relate release notes with certain incremental requirements of projects.	S13		3.59		3.75	0.282	-0.16
	Include expecting release updates (coming requirements for the next release).	S14		4.03		3.30	0.000	0.73
	Release notes could facilitate internal users to discuss key requirement changes of projects.	S15		4.06		3.32	0.000	0.74
T5. Software design	Information in release notes can facilitate to trace design decisions and rationale, and vice versa.	S16		3.98		3.67	0.108	0.31
	Significant architectural changes can be found in release notes of major releases.	S17		3.94		3.66	0.306	0.147
	Release notes could provide users with an official description of the design decisions.	S18		3.85		3.39	0.000	0.28
	Information in release notes could help software refactoring and re-architecting.	S19		3.85		3.75	0.090	0.10
T6. Software testing	Release notes benefit and are highly related to software testing activity.	S20		3.45		3.78	0.001	-0.33
	Testers engaged later will cost more time and efforts of releases and producing release notes.	S21		4.02		3.80	0.241	0.22
	Release notes with unclear descriptions of issues are limited in helping the test team.	S22		3.01		3.89	0.000	-0.88
T7. Software evolution	Information in release notes indicates how projects have evolved.	S23		3.74		3.70	0.995	0.04
	Release notes help internal release note user to be aware of the risk (debt) exists of the projects.	S24		4.02		3.40	0.000	0.62
	Information of major release notes would be useful for understanding software architecture evolution.	S25		4.01		3.82	0.166	0.19
T8. Software artifacts	The agile development process impacts the quality of release notes.	S26		3.91		3.76	0.063	0.15
	Establishing traceability between other software artifacts and release notes.	S27		3.71		3.65	0.072	0.06

and time-consuming. In [3], Moreno *et al*. also reported that participants explained the time needed for creating a release note between four and eight hours. Two (2) interviewees (i.e., release note producers) mentioned that release note producers vary between releases (i.e, not always certain people who document release notes). Three (3) interviewees (two release note producers and one user) mentioned that being aware of the process of release note production would give a clue to the hierarchy of developers in software development, and it is important to be clear "who knows what" **[S4]**.

As results shown in Section 4.1.1, 17 contributors, on average, are involved in release note production for each release. However, the interviewees' feedback shows that the current information-sharing hinders producing and using release notes effectively. One promising way to improve the productivity of development and artifact documentation is to identify the core developers and to build effective communication and organizational strategy. This would facilitate better information sharing for documentation and positively impact development **[S2][S5]**. For example:

- "*Update the milestones tab on release tracking spreadsheet. Check milestones tab of release template for how to. This will ideally be ready at least two weeks prior to the start of the cycle, with* ***feedback received from key stakeholders (QA, RelEng, RelMan) prior to wider publishing***."
- "*It is the need for the* ***release manager*** *to email the release-drivers list prior to the creation of the builds (ideally shortly after the decision is made to go forward with the release) to notify all stakeholders of the forthcoming release. Also, the release manager should verify that the rollout percentages in Balrog and Google Play for the current release are set as expected (taking into account any blocking quality issues) to avoid unexpected fallback versions when the new release ships. Finally, if there are security fixes being included in the release, email abillings (or whoever from the* ***security team*** *handles CVEs and security advisories) to ensure that they are aware of the bugs being fixed*."
- "*The details of how a release plan created* ***depend greatly on the size and composition of a team***."

Topic 2: Tool usage for release note production and management. As discussed in Section 2.1, some works have developed tools that can be used for (semi) automatically producing release notes. However, our interviews found no participants that have adopted such tools (e.g., ARENA [3]) for automatically producing release notes. Seven (7) interviewees mentioned that they applied basic text management tools (e.g., issue trackers and Wikis) to help them keep track of changes for producing and managing release notes **[S6]**. However, they also mentioned that they still need to spend considerable manual effort creating the release note text from these sources. Three (3) interviewees (two release note producers and one release note user) mentioned that developing tools for automatically documenting release notes would be promising. For example, tools for finding what information should be documented in release notes, and linking software artifacts to release notes would be useful for making release notes more informative **[S5]**. Examples:

- "*With the right process and **tools**, we can significantly increase the usefulness of release notes and leverage them to create happier customers. We can view this as a "release notes maturity model" with increasing sophistication of release notes leading to increased quality*."
- "*If you use any issue-tracking system or a product **management tool** such as all-in-one product management platform Hygger.io, you may definitely apply it to generate your Release Notes*."

Topic 3: Key information in release notes and general documenting styles. We identified six statements within this topic, such as interviewees' common considerations on key information needed, general documenting styles, and information collection considerations. Eleven (11) interviewees (six release note producers and five release note users) mentioned that information on "**New features**" in new releases is the most important to them. However, they also said that the descriptions of this type of information should be improved, and they could often not get enough and detailed descriptions about the new features **!**[**S7**]. Three (3) interviewees (all release note users) mentioned that sometimes it is changeling to find key information, which is currently scattered in release notes. Four (4) interviewees (all release note users) mentioned that risk management is a key activity for project releases, and they expect information about evolution (e.g., historical information) and potential risk issues to be in release notes **!**[**S8**]. However, they did not extract much-related information when reading the most current release notes [**S9**][**S11**]. Five (5) interviewees (release note producers) considered that they normally follow a pattern of documenting release notes, and almost every release note has similar writing styles. Even though they realize that such documenting styles might hinder the release note usage, collecting information is time-consuming and would take them around six hours on average [**S10**][**S12**]. Furthermore, fourteen (14) interviewees (seven release note producers and seven release note users) mentioned the importance of clear writing and well-organized release notes (e.g., having appropriate length release notes that are not too long or short, and release notes that are better targeted to their users) [**S12**] [40] [41].

- "***Bug fixes and performance improvements** is completely meaningless. What was fixed? How will performance improve? Where along the user experience can we expect improvements?*"
- "*Release notes can be a great way to increase customer satisfaction and confidence as well as attract new customers. **However, most information on release notes treat as an afterthought**, something that just has to be done in order for the version to ship. This leads to a deliverable is barely informative at best."*
- *"**Nobody likes to read a descriptive information**. Keep it bullet points separated to make it easily understandable. If required you can add a line or two for a better explanation."*
- "*It is **really tedious when I did release notes documents**, and I usually spent a couple of hours on it*."

Topic 4: Release notes and software requirements. Iterations of releases allow developers to change and add requirements to products. At a high level of abstraction, release planning and documentation could be described as selecting an optimal subset of realization requirements in a particular release. Release planning is where requirements engineering for market-driven software product development addresses the market perspective. The requirements embodied in releases determine what users can get from the new releases [42]. Release notes contain rich information about the new and implemented requirements of the projects. We identified three statements of this topic. Eight (8) interviewees (four release note producers and four release note users) mentioned that there exists a close relationship between information documented in release notes and iterative requirements of projects [**S13**]. For example, a set of software requirements may be included in the next release, but it is necessary to select a part of the requirements to implement. Five (5) interviewees (all release note users) stated that they expect more information about new functions in release notes of coming releases, and with these they can take actions and discuss with other developers before changes are introduced **!**[**S14**] **!**[**S15**].

- "*You already know how to prepare a product specification, how to build great plans and strategies using smart road maps, prioritize tasks and objectives, **create product requirements document** (PRD) and so on and so forth. It's time to pay some attention to one more document – Release Notes*."
- "*I'm simply including a list of **features that they've requested which were implemented in the current release, along with the descriptions from the requirements documents***."
- "*Another release problem occurs when **someone who doesn't actually know what's going on decides to release the software**. Deb, a QA manager, found herself in a tough spot. Development had turned over part of the software, and her team had finished preliminary planning and exploratory testing*."

Topic 5: Release notes and software design. The changes between releases usually involve multiple design decisions [43]. Software designers have to make changes in a product over time based on the requirements. The release note is one of the high-level software artifacts that can highlight the significant changes between releases. These changes are highly related to software design decisions, e.g., adding new functions. We identified four key statements of this topic. Eight (8) interviewees (five release note producers and three release note users) said that release notes should be a rich source for capturing high-level design decisions and rationale that can be used for software re-architecting, refactoring, and reuse [**S16**]. Four (4) interviewees (all release note users) mentioned that focus on release notes of major releases would help extract significant architectural changes [**S17**], and such information could help avoid architectural erosion [**S19**]. Three (3) interviewees (all release note users) mentioned that release notes facilitate some other internal users (e.g., developers or clients) to discuss significant design decisions **!**[**S18**]. Example statements:

- "*Release note as **a retrospective review of the project and architecture evolution***."

- "*Release notes serve as a great "source of truth" (at least at a **high level**) for what has changed. Sales and marketing teams can use release notes as a resource for when they talk to **customers and plan new content.** The very practice of writing release notes as a team can **improve communication** and get more team members aligned on the release. Support teams can reference release notes, or point customers directly to them, as they receive questions or feedback about the product.*"

Topic 6: Release notes and software testing. Three statements were identified related to this topic. Five (5) interviewees (two release note producers and three release note users) mentioned that release notes are supposed to be helpful for software testing, as release notes contain a set of issue fixing information. Testers could better understand issue fixed information in release notes that will positively influence testing ❗[**S20**][**S21**]. However, four (4) interviewees (release note users) mentioned that although release notes may contain issues fixing information, some users (e.g., end-users) would not be interested in which bugs have been fixed. One interviewee (release note user) claimed that for internal release note users (e.g., testers), the documented issue fixing information in current release notes is useful. However, they also commented that the way organizing the related information (e.g., linking to the issue tracking system) should be improved ❗[**S22**]. Example statements:

- "*It's better to keep track of all development activities right from the beginning. Don't aim to prepare the notes at the very end because you'll surely miss something. Release notes have separate sections for **testing activity** and a set of stakeholders. Based on and working with your target audience you'll need to work out which bits of information to include, and which can be omitted.*"
- "*Good release testing practices often lead to smoother, more regular releases. Since testers excel at exploratory and confirmatory testing, **release testing is a place that testers** can directly contribute their skills to releases. *"

Topic 7: Release note and software evolution. Developers undertake a great variety of software activities to accomplish new versions of projects. Software projects evolve, but the original design decisions and rationale could be scattered and lost. Release notes could be a data source, which supports understanding software evolution. We identified three statements in this topic. Four (4) interviewees (release note producers) mentioned that extracting information in release notes would be a guideline for understanding software's change history and how it evolved [**S23**]. However, three (3) interviewees (two release note producers and one release note user) also stated that release notes should be improved regarding the description of major release notes ❗[**S24**] [**S25**]. Example statements:

- "*Internal Release Note is for company's own use and mostly prepared for the Software Testing Team and **define for the next releases**. External Release Note is for Customer and End Users containing information about latest production release.*"
- "*Release Notes tell your product's story. **As your product evolves over time, having a historical log of these changes, improvements, and fixes helps tell your product's story**. Release notes are a celebration of your team's work and the evolution of your product's growth. A good change log or set of release notes can be your product's historical diary.*"

Topic 8: Release note and other software artifacts. A set of software artifacts is produced during the software development, and there is a close relationship between various software artifacts. Three (3) interviewees (all release note producers) stated that documenting release notes for agile development is quite challenging. The potential reason is that a set of unstructured information produced during development, and documenting and tracing key information for release note production is time-consuming and difficult. Four (4) interviewees (all release note producers) mentioned that projects adopt agile methods or small companies could not produce release notes very well, and few software artifacts are available for referring to when documenting release notes [**S26**]. Seven (7) interviewees (release note producers) highlighted the importance of other software artifacts for release note production. Establishing traceability between other software artifacts (e.g., source code) and release notes would be helpful for software development [**S27**], e.g., enriching requirement documents and architecture documents. Example statements:

- *"You could also include specific **links to other documentation or issues in your issue tracker as well**. The goal here is to include enough information to stakeholders so they **can be knowledgeable about what changes are coming up**."*
- *"Smaller groups might be able to do more informal, lightweight plans while larger enterprise groups need **formal documentation** and approvals. The details of how a release plan is created depends greatly on the size and composition of a team."*
- *"A single release pipeline can be linked to **multiple artifact sources**, of which one is the primary source. In this case, when you create a release, you specify individual versions for each of these sources."*

Key findings for RQ2.1
We summarized a catalog of eight empirically-justified topics that highlight release note producers and users' key opinions. The summarized topics and statements about release notes are related to **a set of software activities** that can shed light on ways to produce and use release notes more effectively.

4.2.2 Results of RQ2.2: What are significant discrepancies between release note producers and users in perceiving release notes?

We conducted an online survey and invited participants to score and confirm the 27 statements. Participants were classified into two groups: **Release Note Producer** and **Release Note User**, and we also identified the roles of the participants, i.e., architects, developers, team managers, testers, and operators (see Fig. 5). Our results show that most participants are involved in software development and design (283 out of 314, 90.1%). Architects account for 69.3% (i.e., 68 out of 98) of the Release Note Producers, and developers account for the largest group of the Release Note

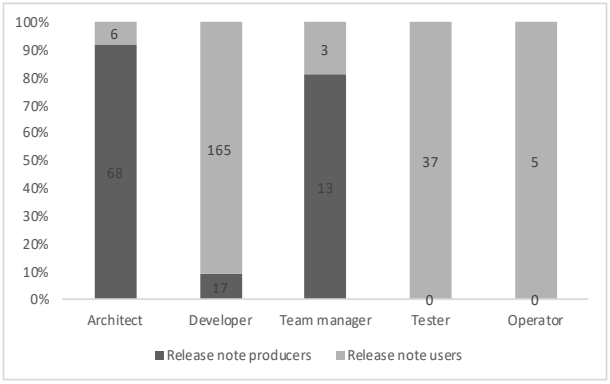

Fig. 5. Main tasks and functions of the participants in the online survey.

Users. Most of the participants are from GitHub projects and industrial IT companies. As such, the participants are mainly from technical backgrounds, and few end-users are included.

The results of the online survey are summarised in Table 8. We analyzed the Likert scale "Agreement" to each statement, and P-value is applied to test the discrepancies between the two groups (i.e., Release Note Producer and Release Note User) whether the differences in the "agreement" for each statement are statistically significant differences between the two groups at a 95% confidence level. The results show that nine statements with statistically significant discrepancies between the two groups are highlighted with Green colored cells.

The Effect Size quantifies the difference between **Release Note Producer** and **Release Note User**, and the values of Effect Size are shown in the last column of Table 8. For example, the mean score of S3 for release note producers is 3.53, whereas the mean source for release note users is 3.87; as a consequence, the effect size is 3.53 - 3.87 = - 0.34, which means release note users agree this statement more. We use a blue color to indicate the former group (i.e., Release Note Producer) is more likely to agree with the statement, and orange color indicates the latter group (i.e., Release Note User) is more likely to agree with the statement.

Based on the results of Effect Size, the **Release Note Producer** group is more likely to agree with five statements with statistically significant difference. The **Release Note User** group tends to agree more with four statements. We interpret the statements, which show the statistically significant difference between Release Note Producers and Release Note Users. We applied "✓" to present "*agree*" and "✗" to denote "*disagree*" of each statement with comments provided by the participants in Table 12 and 13 of Appendix.

1. Release Note Producer group is more likely to agree with the following statements.

- Release note producers are more likely to focus on non-functional requirements **!**[**S8**].
- Release note producers would like to map release notes to a set of requirements of projects **!**[**S14**].
- Release note producers emphasize the high-level information (e.g., major changes) that need to be documented in release notes **!**[**S15**].
- Release note producers consider that release notes are a rich source of design decisions **!**[**S18**].
- Release note producers consider that release notes can help internal release note users to be more aware of the risks for projects **!**[**S24**].

2. Release Note User group is more likely to agree with the following statements.

- Some internal release note users are not clear on how to align documenting and using release notes **!**[**S3**].
- Release note users expect detailed descriptions of new features, but the information is not well documented in most current release notes **!**[**S7**].
- Release note users consider that release notes contains too much bug-fixing information, which is supposed to be helpful for testing **!**[**S20**]. However, they consider the way of documenting such related information should be improved **!**[**S22**].

Moreover, the agreements without significant statistically difference confirm that the importance of release note production and usage in practice. Such statements help understand the characteristics of release notes, and being aware of the statements with scores of average 4 or higher also would facilitate release note production and usage effectively in practice, for example:

- Release note producers tend to be the core developers (architects, team managers, and core testers) in software development, and identifying the core developers and building an effective personalization strategy is beneficial for documenting release notes and other software activities [**S1**][**S2**].
- Documenting style is critical for producing quality release notes [**S12**].

In addition, we found that several statements with more comments are also worth to be paid attention, for example,

- The release note plays an important role to facilitate software evolution (with 31 comments) [**S23**].
- Clear structured and the writing styles of release note documentation are vital (with 20 comments) [**S12**].

Key findings for RQ2.2
Several significant discrepancies exist between Release Note Producers and Release Note Users in perceiving release notes, and being aware of the common considerations (statements) and the discrepancies would help fill the gaps between them.

5 Discussion

We interpret the results and discuss the implications for researchers and practitioners in this section.

5.1 Implications for researchers

Release note production and management tools. The results of RQ1 (the "What") show that various information could be included in release notes (e.g., the new features and improvements made of projects). However, based on the results of the interviews and the online survey, participants stated the information in release notes is currently

scattered and not well structured, and the information tends to be described vaguely. Consequently, release notes sometimes provide limited help to release note users. To improve the effectiveness of release note management and production, some interviewees mentioned that they applied management tools (i.e., Wikis and issue trackers). Besides these basic management tools, researchers could consider developing new tools and employing new methods, such as automatically extracting more comprehensive information from software artifacts to document release notes. As we described in Section 2, there have been several attempts to develop tools for (semi) automatically producing release notes [3]. Interviewees and participants mentioned that they almost never used such tools. We encourage researchers to investigate the relationships between release notes and other software artifacts and develop novel and practical tools to automatically document release notes. They could look into ways to better classify and structure the information in release notes and automatically link it to other software artifacts.

The discrepancies between release note producers and users in perceiving release notes. The release note is a key software artifact in practice to provide an overview of new and changed features of releases for users. However, based on the results of RQ2.2, we found that a considerable gap exists between release note producers and users in perceiving release notes (see Table 8). Few works to date investigate the impact of release notes on software activities and how to fill the gaps regarding producing and using release notes. For example, a better understanding of what information could be included targeting different users and domains would be beneficial for release note production. In addition, addressing current gaps helps internal release note users better collaborate, e.g., sharing key information about artifacts during development.

Assessment of release note quality. Release note quality is important for usage in practice. However, there are no standards or guidelines for documenting release notes. Based on the results of RQ1, much of the information in release notes are not documented very clearly and poorly structured (e.g., bugs fixed and documentation updated). In addition, the results of RQ2 show that the importance of release note quality and current quality needs to be improved (i.e., four interviewees and 56 participants of the survey mentioned). Our findings of RQ1 and RQ2 show that a set of factors can be considered to produce quality release notes. For example, projects domains, relevant software activities, and targeted users. As such, a guideline with those factors for assessing and producing release notes would be helpful.

5.2 Implication for practitioners

Core developers in software development and documentation. Based on the results of RQ1 ("**Who**"), we found that there are many contributors who are involved in contributing release notes in GitHub projects, but only several contributors (i.e., six, on average, of a project) who lead the formal release note production. These core release note producers normally differ between releases. As we discussed in Section 4.1.1, these core release note producers are more likely to be core developers of the projects. In addition, based on the interviews' results, five interviewees mentioned that release note documentation normally takes place between core developers of the projects (e.g., architects or project managers). As such, identifying these core contributors, especially in open source projects, would help improve the collecting and sharing of key information of design decisions that impact development and software documentation (e.g., release note documentation).

Difficulties of release note usage in practice. We identified a set of key information that is included in release notes regarding different software domains (see Table 6). For example, users of Application and Software System domains would like to know more about new features instead of the changes made to source code, while users of Web-libraries and Framework domain would like to know about issue related information. However, based on the results of our interviews and the online survey, participants stated that it is often difficult to find useful information. We suggest that practitioners need to make release notes more focused to their target users. For example, to be clearer about the purposes of releases and ensure that documented information is appropriate regarding the domains.

Release notes and software evolution. The results of RQ1 show that information in release notes for major and minor releases is significantly different. For example, the information in the major release notes would help to understand the process and evolution of the projects (e.g., system level information). In addition, the results of RQ2 confirm that documenting release notes is one of ways to effectively communicate changes and enhancements between all team members. Release notes explain what changes are in the latest version, and this can improve development efficiency by providing all necessary information in advance. Another important benefit is that new developers can more easily track the main changes a project has undergone to understand the project and its development process and evolution more quickly. However, a lack of or insufficient release notes (e.g., agile projects) can hinder the refactoring of projects and understanding of software evolution.

Design decisions included in release notes. Release management is the process of planning what, when, and how to release, and the time horizon of planning is dictated by internal and external factors. Based on the results of RQ2.2, 26 participants (release note producers) said that they would select a set of important information in the release notes and they prefer the information is at the package and system level. This information indicates the design decisions, trade-offs, and optimal solutions for projects. As such, investigating the information in release notes would provide a rich source of how design decisions have been made to projects.

A general guideline for producing and using release notes. To more effectively produce and use release notes, based on the results of RQ1 and RQ2 for both researchers and practitioners, we propose the following general guidelines. This list is not meant to exhaustive. As we discussed in Section 1, the ways to produce release notes can also vary from project to project. Therefore, there may be additional guidelines on how to produce and use release notes depends on project sizes, contents, and domains.

- **Who writes release notes**: projects appear to have multiple people contributing to release note authoring and contents, but a small number of main release note producers. These main release note producers play a key role in the development team of deciding when a release note should be produced, what goes into the release note, collecting this information from other developers, and authoring the bulk of the release note. Projects should carefully identify these key developers responsible for its release notes, potentially different developers for main vs. minor release notes, and how they will work with other developers to gather required release note information.
- **The information in release notes**: release note producers need to carefully consider the project domains and targeted users when producing the release notes. For example, in the Application and System Software domains (e.g., Google doc), including New features related information in release notes would be useful for users (see Table 6).
- **Using tools to manage and produce release notes**: release note producers might consider using management tools to improve the productivity of producing release notes and make release notes more organized. This may include tools to support semi-automated production of release notes and may include external information management tools such as Wikis for richer release note information capture and management. In addition, more closely linking release notes with other software artifacts would also be a way to improve the productivity of producing release notes as well as release note usage. Traceability link discovery tools may assist with this.
- **Writing styles of release notes**: release note producers need to make the included information clear, logical, fit to the purpose of release note users, and they might also consider the length of release notes and granularity of the contained information. For example, documenting key System and Package level information for **major** releases. Class level information would instead be documented in release notes for **minor** releases (see Table 7).
- **Engaging with core internal release note users**: release note production should include other developers (e.g., developers and testers) for their opinions. Release notes are not only for outside end-users but also vital for internal users. Release note producers can consider to document the information that would benefit various software activities for different internal users (see results of Section 4.2.1).

6 Threats to validity

We use the guidelines in [44] to discuss key threats to the validity in this work.

Internal validity focuses on factors that may influence the validity of the results. A threat in Stage 1 (i.e., the empirical study) is that we may not include a representative data set of release notes for analysis, as we applied three criteria to filter GitHub projects with 6,000 stars. Moreover, some GitHub projects use their specialized ways to produce and manage release notes outside GitHub, but we only included the release notes that were documented in GitHub. To mitigate threats, we randomly selected 1,000 projects that met the selective criteria and included 32, 424 release notes to analyze. It is possible that this filtering omitted release notes of both small projects and outside data sources. More release notes from various size projects and information from external data sources are required to be included for a better and more comprehensive analysis. We leave this as future work. The other key threat of Stage 1 is that if we correctly label and encode the data. To mitigate this threat, we employed a descriptive statistics method to present the results. For data labeling and encoding, we conducted a pilot analysis to ensure all authors reached agreement on the data labeling and encoding used. The first author and another Ph.D. candidate then used MAXQDA for the formal data labeling and encoding, and the second author of this paper reviewed the labeling and encoding results. This partially reduces the threat of bias in our qualitative data analysis, though this threat still exists.

Construct validity reflects what extent the research questions and the methodology are appropriately used in a study. The main threat to this validity in our study is whether the data we analyzed and coded can answer the research questions. Before we performed this case study, we prepared a research protocol to identify the potential research questions and appropriate ways to collect and analyze the data to mitigate this threat. Additionally, two annotators (i.e., the first author and the Ph.D. candidate) analyzed and coded the collected data independently, and the second authors reviewed their results to reduce any personal bias from the data analysis. One threat exists in Stage 3 (i.e., our online survey) in our use of the "Neutral" option on the Likert scale. Participants could choose this middle option to easily opt out of a question. However, as shown in Table 8, the "Neutral" option was rarely used, and we checked the results of the online surveys to verify that if this Neutral option was not considered, the results have no significant difference to the current reported ones. An alternative is to use a bipolar scale [32] without the "Neutral" option (i.e., Strongly Disagree, Disagree, Slightly Disagree, Slightly Agree, Agree, and Strongly Agree) that can convert the respondent answers into a bipolar statement.

External validity concerns the generality of the study results in other settings. This depends on the sampling methods we employed and representativeness of the data used. A threat in Stage 1 is that if the release notes we collected are representative. To reduce this threat, we used a large number of representative GitHub repositories and extracted a large number of release notes to analyze. One threat exists in Stage 2 is that if the interviewees who participated in this study could be representative. To mitigate this threat, we conducted the interviews with 15 professionals who work at different companies, domains, and countries. In addition, the participants who were involved in the online survey worldwide. Further studies with more release notes, interviewees, and survey participants would need to be done to demonstrate the generality of our results to other projects and more participants.

Reliability refers to whether the study gets the same results when other researchers repeat it. The threats to the

replicability of this study are the applied data collection and analysis methods. To mitigate the threats, we have described each step of this work explicitly. By making explicit the process of data collection and analysis of this study (see Section 3.3), we believe that this study can be replicated.

7 Conclusion and future work

We report a study to understand and characterize release notes. Our main findings are that various information is documented in release notes that can be classified into eight topics. A number of contributors are intensively involved in producing release notes, and some of contributors play a key role in information sharing during development. The production information of release notes for major and minor releases regarding different domains are significantly different (see Table 6). 27 statements concluded from release note producers and users' considerations of perceiving release notes, and the discrepancies between them were identified. Both release note producers and users confirmed that the release note is a vital artifact of software development. The information in release notes could be extracted and used to enrich other software artifacts and trace various software activities. For example, "Software internal changes" related information can help to better support understanding software evolution and enriching architecture documents.

We see several promising research directions: (1) Comparing release notes among open source projects, commercial projects, and Apps. We plan to investigate the characteristics of release notes of different venues and identify the differences of release notes usage. (2) Studying the interactions between release notes and other software artifacts, e.g., differences and similarities between "Read me" files and release notes. (3) Developing new tools that could greatly help improve release note production and usage, e.g., generating release note contents, linking release notes to other key artifacts, and summarising or indexing release note contents for different release note users. (4) Investigating end-users' opinions about release notes that would be helpful in documenting end-user targeted release notes.

Acknowledgment

The authors would like to thank all the interviewees and participants of this study, without them, this work will never be accomplished. In addition, the authors would like to thank our anonymous reviewers for their constructive feedback and suggestions that greatly improved our paper. This research was partially supported by the Australian Research Council's Discovery Early Career Researcher Award (DECRA) funding scheme (DE200100021), ARC Laureate Fellowship (FL190100035), and ARC Discovery grant DP200100020.

References

[1] Bram Adams, Stephany Bellomo, Christian Bird, Tamara Marshall-Keim, Foutse Khomh, and Kim Moir. The practice and future of release engineering: A roundtable with three release engineers. *IEEE Software*, 32(2):42–49, 2015.

[2] Hyrum K Wright and Dewayne E Perry. Release engineering practices and pitfalls. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 1281–1284. IEEE, 2012.

[3] Laura Moreno, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, Andrian Marcus, and Gerardo Canfora. Arena: an approach for the automated generation of release notes. *IEEE Transactions on Software Engineering*, 43(2):106–127, 2016.

[4] Gregorio Robles-Martínez, Jesús M González-Barahona, José Centeno-González, Vicente Matellán-Olivera, and Luis Rodero-Merino. Studying the evolution of libre software projects using publicly available data. In *Proceedings of the 3rd Workshop on Open Source Software Engineering, 25th International Conference on Software Engineering*, pages 111–115, 2003.

[5] Pamela Bhattacharya, Marios Iliofotou, Iulian Neamtiu, and Michalis Faloutsos. Graph-based analysis and prediction for software evolution. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 419–429. IEEE, 2012.

[6] Jason Tsay, Hyrum K Wright, and Dewayne E Perry. Experiences mining open source release histories. In *Proceedings of the 2011 International Conference on Software and Systems Process*, pages 208–212, 2011.

[7] André Van der Hoek and Alexander L Wolf. Software release management for component-based software. *Software: Practice and Experience*, 33(1):77–98, 2003.

[8] Hennie Huijgens, Arie Van Deursen, and Rini Van Solingen. The effects of perceived value and stakeholder satisfaction on software project impact. *Information and Software Technology*, 89:19–36, 2017.

[9] Jim Whitehead. Collaboration in software engineering: A roadmap. In *Future of Software Engineering (FOSE'07)*, pages 214–225. IEEE, 2007.

[10] Filippo Lanubile, Christof Ebert, Rafael Prikladnicki, and Aurora Vizcaíno. Collaboration tools for global software engineering. *IEEE software*, 27(2):52–55, 2010.

[11] Foutse Khomh, Bram Adams, Tejinder Dhaliwal, and Ying Zou. Understanding the impact of rapid releases on software quality. *Empirical Software Engineering*, 20(2):336–373, 2015.

[12] Surafel Lemma Abebe, Nasir Ali, and Ahmed E Hassan. An empirical study of software release notes. *Empirical Software Engineering*, 21(3):1107–1142, 2016.

[13] Martin Michlmayr, Francis Hunt, and David Probert. Release management in free software projects: Practices and problems. In *IFIP International Conference on Open Source Systems*, pages 295–300. Springer, 2007.

[14] Günther Ruhe, Mark Stanford, et al. Intelligent support for software release planning. In *International Conference on Product Focused Software Process Improvement*, pages 248–262. Springer, 2004.

[15] Günther Ruhe. *Product release planning: methods, tools and applications*. Auerbach Publications, 2010.

[16] Inge Van De Weerd, Sjaak Brinkkemper, Richard Nieuwenhuis, Johan Versendaal, and Lex Bijlsma. Towards a reference framework for software product management. In *14th IEEE International Requirements Engineering Conference (RE'06)*, pages 319–322. IEEE, 2006.

[17] Liguo Yu. Mining change logs and release notes to understand software maintenance and evolution. *CLEI Electron Journal*, 12(2):1–10, 2009.

[18] Mihai Codoban, Sruti Srinivasa Ragavan, Danny Dig, and Brian Bailey. Software history under the lens: A study on why and how developers examine it. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 1–10. IEEE, 2015.

[19] Michael Fischer, Martin Pinzger, and Harald Gall. Populating a release history database from version control and bug tracking systems. In *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.*, pages 23–32. IEEE, 2003.

[20] Keith H Bennett and Václav T Rajlich. Software maintenance and evolution: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 73–87, 2000.

[21] Shaun Phillips, Guenther Ruhe, and Jonathan Sillito. Information needs for integration decisions in the release process of large-scale parallel development. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1371–1380, 2012.

[22] Sebastian Klepper, Stephan Krusche, and Bernd Bruegge. Semi-automatic generation of audience-specific release notes. In *2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED)*, pages 19–22. IEEE, 2016.

[23] Ines Mergel. Open collaboration in the public sector: The case of social coding on github. *Government Information Quarterly*, 32(4):464–472, 2015.
[24] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131, 2009.
[25] Victor R Basili-Gianluigi Caldiera and H Dieter Rombach. Goal question metric paradigm. *Encyclopedia of software engineering*, 1:528–532, 1994.
[26] Hudson Borges, Andre Hora, and Marco Tulio Valente. Understanding the factors that impact the popularity of github repositories. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 334–344. IEEE, 2016.
[27] Klaas Andries de Graaf, Peng Liang, Antony Tang, Willem Robert van Hage, and Hans van Vliet. An exploratory study on ontology engineering for software architecture documentation. *Computers in Industry*, 65(7):1053–1064, 2014.
[28] Astride Aregui and Thierry Denœux. Constructing consonant belief functions from sample data using confidence sets of pignistic probabilities. *International Journal of Approximate Reasoning*, 49(3):575–594, 2008.
[29] Tingting Bi, Xin Xia, David Lo, John Grund, and Thomas Zimmermann. *What Make a Good Release Note: Complementary Material: https://tinyurl.com/y92yl9pq*.
[30] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*, pages 285–311. Springer, 2008.
[31] Forrest Shull, Janice Singer, and Dag IK Sjøberg. *Guide to advanced empirical software engineering*. Springer, 2007.
[32] Valerie M Sue and Lois A Ritter. *Conducting online surveys*. Sage, 2012.
[33] Barbara A Kitchenham and Shari L Pfleeger. Personal opinion surveys. In *Guide to advanced empirical software engineering*, pages 63–92. Springer, 2008.
[34] I Elaine Allen and Christopher A Seaman. Likert scales and data analyses. *Quality progress*, 40(7):64–65, 2007.
[35] Teemu Karvonen, Woubshet Behutiye, Markku Oivo, and Pasi Kuvaja. Systematic literature review on the impacts of agile release engineering practices. *Information and Software Technology*, 86:87–100, 2017.
[36] Gail C Murphy. Beyond integrated development environments: adding context to software development. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pages 73–76. IEEE, 2019.
[37] Mika V Mäntylä, Foutse Khomh, Bram Adams, Emelie Engström, and Kai Petersen. On rapid releases and software testing. In *2013 IEEE International Conference on Software Maintenance*, pages 20–29. IEEE, 2013.
[38] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. Quality and productivity outcomes relating to continuous integration in github. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 805–816, 2015.
[39] Zhifang Liao, Haozhi Jin, Yifan Li, Benhong Zhao, Jinsong Wu, and Shengzong Liu. Devrank: Mining influential developers in github. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.
[40] Erik Berglund and Michael Priestley. Open-source documentation: in search of user-driven, just-in-time writing. In *Proceedings of the 19th annual international conference on Computer documentation*, pages 132–141, 2001.
[41] Remco C De Boer and Hans Van Vliet. Writing and reading software documentation: How the development process may affect understanding. In *2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, pages 40–47. IEEE, 2009.
[42] Pär Carlshamre. Release planning in market-driven software product development: Provoking an understanding. *Requirements engineering*, 7(3):139–151, 2002.
[43] Gunther Ruhe and Moshood Omolade Saliu. The art and science of software release planning. *IEEE software*, 22(6):47–53, 2005.
[44] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

TABLE 9
Extracted data items and data analysis methods for Stage 1 (i.e., the large scale empirical study).

No.	Data item	Data analysis method	Relevant RQ
D1	The number of release notes of each project.	Descriptive statistics	RQ1
D2	The duration of each project.	Descriptive statistics	RQ1
D3	The number of commits of each release and contributor.	Descriptive statistics	RQ1
D4	The domain of the selected GitHub projects.	Descriptive statistics.	RQ1
D5	The number of release notes producers.	Descriptive statistics	RQ1
D6	The information documented in release notes.	Descriptive statistics and constant comparison	RQ1

TABLE 10
Extracted data items and data analysis methods for Stage 2 (i.e., interviews with professionals).

No.	Data item	Data analysis method	Relevant RQ
D7	The functions of the interviewee in software development.	Descriptive statistics	RQ2.1
D8	The contents and elements, benefits, limitations, software activities, and suggestions related to release notes interviewees discussed (Based on the interview questions).	Descriptive statistics and constant comparison	RQ2.1 and RQ2.2

8 APPENDIX

TABLE 11
Extracted data items and data analysis methods for Stage 3 (i.e., Online survey).

No.	Data item	Data analysis method	Relevant RQ
D9	The functions of the interviewee in software development of participants.	Descriptive statistics	RQ2.2
D10	The scores of statements.	Descriptive statistics and constant comparison	RQ2.2
D11	The comments of surveys that participants provided.	Descriptive statistics and constant comparison	RQ2.1 and RQ2.2

TABLE 12
Comments from Release Note Producers and their roles

Comment from release note producer	Role
Comment ✓"*A release plan should be available to all individuals involved for the entire release.* ***This may include developers, release engineers, testers, managers and system admins, or operations personnel***", and "*Knowing who your stakeholders are is key here. Think about who they are, what they care about, and how best to communicate with them. As well, consider* ***if they're internal or external to your team or organization****. But some developers* ***are interested in documenting, and they don't know what should to document***	Architect
Comment ✓ "*Some post-release behaviors might include end users logging in and then logging out of the app for changes to take place, or as a requirement of using the newly released version of the software.* ***Stakeholders should be aware of details that may affect their work during a release process.*** *Customers should also be aware of changes that might impact their work as well. Notifying a customer of a planned outage or steps they will need to take after an update,* ***ahead of the release, should be considered critical communication***."	Architect
Comment ✓ " *Even if a team has a good plan for releasing, informing stakeholders of all or parts of this plan is equally as important. It's key for setting expectations and for allowing the entire release to go smoothly. Shareholders that experience problems during, or soon after, a release may take some actions which may end up reducing frequency and/or quality of future releases if they perceive a major issue. Often, this occurs due to a communication failure instead of a technological failure.* ***Solid information about non-functional requirements to all stakeholders*** *can prevent these communication breakdowns from happening*."	Architect
Comment ✓ *"These are the people who're getting the most value from your product, who're emotionally invested in your team's success.* ***I would like to focus on the high level information when I document release notes***."	Architect

TABLE 13
Comments from Release Note Users and their roles

Comment from release note users	Role
Comment ✓*"Release notes are a really interesting engagement opportunity to me—most people don't read them, but those that do represent a highly targeted audience of very engaged users. Every company with an app has to write them, and I love to see* ***what issues have been fixed, but they are not always clear****."*	Developer
Comment ✗"*The release contents is an internally-consumed document that fosters communication between teams on a project, it might start with a high-level description of the release's* ***major themes instead of a list of fixed bugs***"	Architect
Comment ✓ *"****Bug fixes and performance improvements" is completely meaningless****. What was fixed? How will performance improve? What along the user experience can we expect improvements?*	Tester
Comment ✗ "***Quality attributes, like response time, are very important information to me, but bug fixes and performance improvements****" on the box and call it a day, but that doesn't really tell your users and internal stakeholders anything, maybe link an issue number?*"	Tester
Comment ✓ "*Release notes are a really interesting engagement opportunity to me—most people don't read them, but those that do represent a highly targeted audience of very engaged users.* ***I also have no idea what information I should provide to contribute the release notes****."*	Developer

TABLE 14
Background information of the interviewees.

Interviewee	Professional years	Professional role	Project domain	Release note documentation/use experience	Country
I1 and I15	12 years	Architect	Application system (Financial system)	Release note producers	China
I2	12 years	Architect	Application system (Transport systems)	Release note producer	China
I12	10 years	Manager	Application system (Shopping system)	Release note user	Japan
I3	8 years	Pre-sale	Application system (Entertainment system)	Release note user	Australia and China
I4, I13, and I14	5 years	Architect	Application System (Financial system)	Release note producers	Sweden and China
I5	5 years	Team leader	System Software (Database)	Release note user	The Netherlands
I6 and I7	4 years	Developer	Application system (Manufacturing system)	Release note users	China
I10 and I11	4 years	Developer	Application system (Entertainment system)	Release note producer	China
I8	3 years	Developer	Application system (Entertainment system)	Release note user	Canada and Australia
I9	2 years	Developer and tester	Application system (Entertainment system)	Release note user	China