Software Professionals' Information Needs in Continuous Integration and Delivery

Azeem Ahmad
Linköping University
Linköping, Sweden
azeem.ahmad@liu.se

Ola Leifler
Linköping University
Linköping, Sweden
ola.leifler@liu.se

Kristian Sandahl
Linköping University
Linköping, Sweden
kristian.sandahl@liu.se

ABSTRACT

Continuous integration and delivery consolidate several activities, ranging from frequent code changes to compiling, building, testing, and deployment to customers. During these activities, software professionals seek additional information to perform the task at hand. Developers that spend a considerable amount of time and effort to identify such information can be distracted from doing productive work. By identifying the types of information that software professionals seek, we can better understand the processes, practices, and tools that are required to develop a quality product on time. A better understanding of the information needs of software practitioners has several benefits, such as staying competitive, increasing awareness of the issues that can hinder a timely release, and building a visualization tool that can help practitioners to address their information needs. We conducted a multiple-case holistic study with 5 different companies (34 unique participants) to identify information needs in continuous integration and delivery. This study attempts to capture the importance, frequency, required effort (e.g., sequence of actions required to collect information), current approach to handling, and associated stakeholders with respect to identified needs. We identified 27 information needs associated with different stakeholders (i.e., developers, testers, project managers, release team, and compliance authority). The identified needs were categorized as testing, code & commit, confidence, bug, and artifacts. We discussed whether the information needs were aligned with the tools used to address them.

KEYWORDS

Information Needs, Continuous Integration and Delivery, Question Developers Ask

ACM Reference Format:
Azeem Ahmad, Ola Leifler, and Kristian Sandahl. 2021. Software Professionals' Information Needs in Continuous Integration and Delivery. In *The 36th ACM/SIGAPP Symposium on Applied Computing (SAC '21), March 22–26, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3412841.3442026

SAC '21, March 22–26, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8104-8/21/03…$15.00
https://doi.org/10.1145/3412841.3442026

1 INTRODUCTION

Continuous integration and deployment is a widely adopted approach in which changes to a codebase are committed, built, tested, and deployed to customers frequently. Apart from developers' contributions in continuous integration and delivery, testers also frequently submit new test cases (unit, system or regression, etc.) to test suites, as well as monitoring test execution results on code changes. Stakeholders such as project managers, test leaders, team managers, and product owners seek the information that is generated through the continuous integration pipeline on an as-needed basis. As part of their daily routine, software professionals seek answers to questions such as *"Where is the definition of a specific method?"* or *"Which of the test suites contains my test case?"* Answering these questions requires little effort because the questions are unambiguous and can be answered by focusing on one kind of artifact— in this case, the source code or test suite. However, when a company adopts a continuous integration and delivery pipeline, the effort associated with seeking answers to questions such as *"How much confidence do we have in a specific test suite?"* or *"Are we ready to release a specific version of the software?"* increases significantly because the answers integrate information from a combination of different kinds of artifacts. The fact that these questions can be interpreted differently makes it more challenging to answer them correctly.

In their daily activities, these stakeholders have to answer a variety of questions about code, tests, builds, releases, product quality, etc. [5–7, 13]. Software practitioners have different information needs depending on their particular roles and responsibilities [5]. Although it is very important to catalog and understand the questions and challenges faced by practitioners when attempting to answer those questions [7], little is known about (1) what are the information needs of practitioners [5, 6] and (2) unfulfilled information needs in the field of software engineering [3]. Identifying these information needs can help us better understand the tools, practices, and processes that are important when addressing those information needs [5, 6]. Prior research has been conducted on the questions that were asked by practitioners regarding code [7], configurations as code [10], work support [4], evaluation tasks [13], and artifact traceability [15].

Once the true information needs have been identified, another challenge is to assign priorities to information needs based on importance, frequency and effort. It is equally important to investigate the importance of the identified need, how frequently practitioners seek this type of information, how much effort is involved in answering it (e.g., sequence of actions that must be performed) and the extent to which (i.e., complete, partial or none) available software tools can be utilized to address the information needs.

This study attempts to answer the following questions:

RQ1: What are practitioners' information needs in continuous integration and deployment?
RQ2: To what extent are software tools utilized in industry to address the identified information needs?
RQ3: Do practitioners assign priorities to identified information needs based on importance, frequency and effort?

The detailed contributions of this paper are as follows:

- C1 We identified practitioners' information needs in continuous integration and deployment. (Section 4)
- C2 We investigated the extent (i.e., complete, partial or none) to which the availability of tools supported the identified needs. (Section 5)
- C3 We investigated the following questions related to identified information needs: (1) Primary stakeholders related to each need, (2) How frequently these identified needs were sought by the participants, (3) How long does it take to address each identified need using the participant's current setup, and (4) How much effort is devoted to answering the identified needs? (Section 6)

Section 2 presents related work. Research methods were discussed in Section 3 followed by identified information needs in Section 4. Detailed information about strategies, tools, and stakeholders concerning identified information needs were presented in Section 5. Different attributes such as importance, frequency, effort, and time with respect to each identified information need were discussed in Section 6. Discussion, implication, and conclusion were presented in Sections 7, 8, 10 respectively.

2 RELATED WORK

Researchers [4–7, 10, 11] have been putting effort into identifying the information needs of software teams. Prior research has been conducted to identify the questions that developers ask in configuration as code [10], while performing development activities [4, 6, 7, 11], and during software evolution [13]. Ebert et al. concluded that all of the questions developers ask in their day-to-day routines serve information-seeking goals [2]. LaToza and Brad also observed that 9 of the 10 most time-consuming activities in the day-to-day routines of developers were associated with reachability questions [7]. In another study, LaToza and Brad categorized the questions asked by developers into different domains such as rationale (why it is done this way?), intent (what does this do?), debugging (how is it resolved?), refactoring (how can I refactor this?) and history (who did this?) [8]. Another work to understand the information needs of developers, during software evolution, was conducted by Sharma et al. in which the authors asked 25 developers to prioritize 27 predefined information needs [12]. Also, they proposed the "Smart Advisor" approach to provide automated advice regarding insights about important queries by analyzing data generated during the software development life cycle [12]. Josyula et al. conducted interviews with 17 practitioners to identify information needs related to the software development life cycle [5]. The most frequented information needs were related to clarification of the requirement, design of the products, and skills in programming languages [5].

Fritz et al. [4] developed the information fragment model, which collected information from different sources. This model helped developers to answer 47 pre-identified questions. Sillito et al. in [13] investigated the information needs of programmers regarding a codebase while performing a change task. However, all of these studies were limited to single stakeholders (i.e., developers, testers, or project managers) and a single activity (i.e., development, software evolution, or software configurations).

Our work takes a different approach in that we study continuous integration and delivery, which consists of several activities. Our work is complementary to the study of the information needs of software practitioners and provides a notable connection between needs and the tools that can play an important role in addressing those needs. We also seek to find the importance, frequency, required effort, and total time to answer the identified information needs.

Table 1: Detailed Information about Case Companies

Case	Business Sector	Employees	Claimed Continuous Integration Maturity Level [1]
A	Networks Equipment	25k	High
B	Surveillance Equipment	5k	High
C	Water Equipment	3k	Medium
D	Medical Equipment	3k	High
E	Software Modeling	1k	High

[1] High: CI pipeline support automation from commits to deployment with none or very little human intervention
Medium: CI pipeline support automation from commit to testing where human intervention is required for deployment

3 RESEARCH METHODS

For this work, we presented a multiple-case holistic study [16], in which we studied one phenomenon in five cases. Each case in a multiple case study should be selected carefully so that it either predicts a) similar results or b) contrasting results [16]. We picked (a) so that results from different cases would complement each other to identify the true information needs of practitioners. Yin [16] recommended that a multiple case study would provide more compelling and robust evidence than the findings from a single case study. We maintained a degree of anonymity when describing the participating companies due to the non-disclosure agreements between the companies and our university.

3.1 Cases

Table 1 provides detailed information about cases. Companies were labeled from A-E. We attempted to study different business sectors to avoid generalization and biases. The continuous integration maturity level mentioned in Table 1 was claimed by all practitioners in the investigated cases. Explanations of these levels are provided in the footnote of Table 1.

Table 2: Data Collection

Case	Partic.	Total exp. (Yr)	Designations
A	P_1	10	CI architect, tester, test manager
A	P_2	12	CI architect, developer
A	P_3	15	Test architect, developer, project manager, tester, product owner
B	P_4	10	Developer, software architect
B	P_5	8	Developer, business developer
B	P_6	3	Developer, software architect
C	P_7	10	Software architect
C	P_8	8	Developer, team leader
D	P_9	10	Developer, project manager
D	P_{10}	8	Developer
E	P_{11}	10	CI architect, developer, tester
E	P_{12}	5	Developer
E	P_{13}	5	Developer

Table 3: Cataloging Information Needs from Different Texts

Original Text	"Want to see confidence level of X test suite" - *P_6* "What is the confidence level of this test suite?" - *P_8* "Confidence in test suite is important, before we make release decisions or merge branches"- *P_3*
Identified Information Need	How much confidence do we have in a specific test suite?
Category	Confidence

3.2 Data Collection, Preparation, Analysis and Validation

In all five cases, we collected data through individual interviews with 13 software practitioners. These participants had different roles in the same company. Table 2 presents detailed information about the participants. All of the interactions were conducted online through Zoom or Skype. We recorded the conversations and took notes during the meetings with prior permission from the participants. We conducted 60 minutes long individual semi-structured interviews[1] to collect data from all participants. We filled in a separate Google document with each participant during the interview. Informed consent was obtained from the participants for audio recording. The audio recordings were then transcribed word-by-word into the Google spreadsheet. Audio clips were played 3 times, before writing the text into the Excel sheet. Each cell in the Excel sheet contains texts from one person during a conversation. The names of participants were also anonymized with PX, where X is a number assigned by us.

We read all of the transcripts from the interviews and coded them according to *open coding* [14]. Each paragraph of the transcript of each case company was studied to determine what was said and each paragraph was labeled with one or more codes. Later, we compared all paragraphs from different case companies to collect similar codes (axial codes). These codes were used to categorize information needs. Most of the time, the participants shared the information need directly, e.g., *"Which change requests does a specific commit serve?"* whereas other times, we had to combine different texts/quotes to identify the exact question. Table 3 presents an example of how we extracted information needs and corresponding categories from original texts.

After analysis, the initial codes were checked by one of the researchers. Later, we conducted a physical workshop with 21 different participants from five industries to validate the results. For the data validation exercise, we selected different participants from the ones who had participated in the earlier data collection phase to avoid biases in the results. This workshop was conducted for 120 minutes. Participants were provided with the list of information needs and asked to rank the information needs as "Very Important", "Important", "Moderately Important", "Of Little Importance", and "Good to Know". Since all information needs were identified through extensive interviews, we replaced the "Not Important at all" option in the Likert scale with "Good to Know". Each information need was discussed with the participants to ensure that they understood it.

4 INFORMATION NEEDS

The information needs identified in this study have been categorized as pertaining to *testing*, *code & commit*, *confidence*, *bug*, and *artifacts*.

4.1 Testing

(T_1) Which test cases/test suites have been run on which product? ($P_{1\text{-}4,7\text{-}10}$)
(T_2) Which test cases/test suites have been run on which branch? ($P_{1\text{-}4,7\text{-}10}$)
(T_3) In which environment/machine do specific test cases fail? ($P_{2,3,6,8,12,13}$)
(T_4) Which test suites' execution times have increased recently? ($P_{1\text{-}3,9,11,12}$)
(T_5) What are the build/test results of my commits? ($P_{1\text{-}3,5,6,8,11}$)
(T_6) What are the unstable areas of the code that require more testing/attention? ($P_{3,5}$)
(T_7) Which of the test cases are flaky? ($P_{7,8,11}$)
(T_8) What is a test execution history of a specific test case? ($P_{9,12,13}$)

Testing practitioners in the investigated companies consider planned vs actual test executions as one of the metrics to evaluate software quality before releasing the software to the customers. These planned/actual executions can be either on specific branches such as stable, master or beta (T_1) or on products such as hardware or software services (T_2). Testers and managers preferred that a percentage of executed test cases (i.e., actual/planned*100) should always be available for decision-making regarding software release. An outcome with the value of greater than 80% can be considered as a release candidate provided that all tests pass. Practitioners may also seek an answer to a similar question: *"How many of my test cases have not been executed in the last month on specific requirements, branches, or products?"* The answer to this question is also important for resource allocation. Apart from what has and

[1] https://tinyurl.com/y2uztk5w

has not been executed, developers spend a considerable amount of time to reproducing failures experienced by testers or real users. The availability of the test case failure information — time of execution, or environment information (i.e, SUT configurations, SUT health/unavailability, or OS configuration) — can save time (T_3). Another important piece of information sought by test managers is whether the test execution time has increased recently (T_4). Managers investigate each test suite execution to determine if there is a resource shortage in test equipment. One of the key performance indicators is to look at trends such as how quickly submitted code can be tested and released to customers. Developers in the investigated companies are interested in knowing about the build/test results of their commits (T_5). Developers want to know *"which of the test cases failed on my commits?"*

If the information about unstable (i.e., often buggy) areas of the code is available, practitioners can conduct more testing on those code areas (T_6). In addition to this, the visualization of unstable areas can help in answering questions such as *"Is it suitable for developers to start a new feature in this area of code?"* As far as test instability is concerned, developers spend a significant amount of time on detecting and resolving test case flakiness [9] (T_7). A re-run is a widely adopted approach to detect test instability. One way to avoid test case instability is to monitor the test case history (i.e., comparison of test executions on different builds over time) (T_8).

4.2 Code & Commit

(CC_1) Does the final release to customers include my code? ($P_{1\text{-}13}$)
(CC_2) What is the status/health of new code changes? ($P_{1,2,4,7,8,10,11,12}$)
(CC_3) Which requirement does the specific commit implement? ($P_{2,6,7\text{-}9,10\text{-}13}$)
(CC_4) Which change request does the specific commit implement? ($P_{2,6,7\text{-}9,10\text{-}13}$)
(CC_5) Is the given new feature implemented? ($P_{1\text{-}3,5,6,9,10}$)
(CC_6) Is the given feature ready to release to customers? ($P_{1\text{-}3,5,6,9,10}$)
(CC_7) How often does a specific employee deliver new code to the system? ($P_{1,2,4,9,10}$)
(CC_8) How has my code affected non-functional properties of the product? ($P_{1\text{-}3}$)
(CC_9) Which internal release notes have my comments/code? ($P_{4,5}$)

Developers prefer to know whether or not, customers have received their code changes (i.e., a new feature or a fixed bug) in the latest release (CC_1). Developers receive feedback (i.e., test cases passed/failed) on their changes, but are unaware whether the customers, either real or beta, are making use of their code. The availability of such information encourages developers to write quality code, as shared by one the participants" *"I really want to know where my commit went?"* Managers, in addition to developers and testers, want to ensure that new code changes do not break existing functionality and that all test cases (i.e., manual or automatic) have been executed on the code changes (CC_2). The answer to (CC_2) serves as a critical information before releasing the product to customers. If the changes break existing functionality, it is equally important to know why these changes were introduced into the code base. Developers want to know about the requirements or change request that a specific commit implements (CC_3 and CC_4). One of the participants shared, *"Requirements connections are underestimated. Five years ago, we asked about this only once but now it is a frequently asked question ($CC_{3\text{-}4}$)"*. In addition to *"Why"* changes were introduced, project managers might be interested in *"Who submitted these changes?"* They prefer to know if the given new feature has been implemented or delivered to the CI pipeline (CC_5). If it has been delivered, has it also reached the status where it can be released to the customers (CC_6).

Managers are also interested to know about the frequency of commits by specific developers to a specific product (CC_7). Managers in the investigated companies shared that at this moment they do not care about this type of information, but that once it is available, it can be used for evaluating the efficiency of developers. In addition to monitoring the frequency of developers' commits, managers shared that they were also concerned about the effect of those commits on non-functional properties of products (CC_8). Release notes are distributed with each software release to the customers. In one of the cases, managers shared that, as per company policy, some information is for internal use only and cannot be shared with customers. These internal versions of release notes may be connected to several code changes and practitioners would like to make use of this information later (CC_9).

4.3 Confidence Level

(C_1) How much confidence do we have in the release to deploy to the customers? ($P_{1\text{-}6,\ 9,10}$)
(C_2) How much confidence do we have in the test suites? ($P_{3,4,7\text{-}10}$)
(C_3) How much confidence do we have in stand-alone projects to be merged into the master branch/baseline? ($P_{11\text{-}13}$)

Attaining confidence in a software release is related to many factors such as planned vs actual test executions, achieved coverage, open bugs, etc. Managers want to know the confidence level of the release before making a final decision to release it to the customers (C_1). A similar concept of confidence is applied to specific test suites (C_2). Test Managers worry about test suite stability and whether a particular test suite is capable of revealing bugs. Attaining confidence in stand-alone projects (C_3) to be merged into the master branch is also important. Confidence is considered as an aggregate of all data sources, and an informed opinion based on these data sources can result in the approval of the artifact under consideration. How to interpret the data and how to attain confidence are separate processes.

4.4 Bug

(B_1) Which bugs have been fixed in the specific release? ($P_{1\text{-}6,\ 8\text{-}11}$)
(B_2) How many bugs are still open with specific release? ($P_{1,2,5,6,8,9}$)
(B_3) Is the bug fix ready to release to customers? ($P_{1\text{-}3,5,6,9,10}$)
(B_4) Who broke the build? ($P_{1\text{-}13}$)

All of the investigated companies make use of a ticket/issue management system to track reported issues. Developers may work with one or more issues at a time but managers are interested in the big picture such as *What bugs have been corrected in a release?* (B_1). The ideal situation for a new release is to fix all bugs to gain customers' confidence and improve quality. In addition to fixed

bugs, managers would be interested in knowing how many bugs are still open in a specific release (B_2). A high number of open bugs results in a release delay. Knowing which bugs have been corrected in a specific release (B_1) and whether they are in the state of release to customers (B_3) are separate and important concerns, as raised by the participants.

When the build is broken (B_4), the debugging process is started with the person who submitted the changes, but it is equally important to know what else (external dependencies, network connections, etc.) has been updated together with code changes resulting in a build failure. Is it only one person or a series of events that caused the build failure?

4.5 Artifacts

(A_1) What tasks are pending in the pipeline for a long time? ($P_{1\text{-}3}$)
(A_2) Why and why was this artifact created/modified? ($P_{1\text{-}6,\ 12,13}$)
(A_3) Who created this artifact? ($P_{1\text{-}6,\ 12,13}$)

Apart from pending bugs or increased test suite time, A_1 is concerned with the reasons for pending tasks such as resources shortage, complex environment setup, product unavailability, or time for reviewing changes. Although source code and test cases are software artifacts, in our investigation, artifacts also refer to build scripts, binary files, or simulation models. Practitioners want to know who created this artifact, when, and, why it was created ($A_{2\text{-}3}$) so that queries can be directed to the relevant persons. Practitioners also want to know how a specific stage was reached (missing or duplicated artifacts).

> To answer RQ1 we identified a total of 27 information needs. `Testing` and `Code& Commit` each had eight associated needs. Three needs were identified in the category of `Confidence Level`, four were identified in the `Bug` category, and five information needs were assigned to `Artifacts`."

5 STRATEGIES, TOOLS AND STAKEHOLDERS WITH RESPECT TO INFORMATION NEEDS

In addition to the information needs that practitioners have, this study captures how practitioners are handling these needs in their day-to-day routines. Table 4 presents a summary of the results concerning (1) how companies address identified information needs, (2) the tools that are used by companies to address the identified information needs, and (3) the interested stakeholders. *"In-house Visualization"* refers to the visualization tools that have been built by the company's IT department to address the information needs. *"External Visualization Tool"* refers to situations wherein stakeholders can seek answers from the direct output of external tools or plugins. For example, answers for T_1 & T_2 can be directly sought through the default output from the Jenkin's test framework, or through the use of plugins with no manual intervention necessary. *"Manual Inspection"* is a combination of internal/external tools and human intervention such as manually traversing the logs, sending emails to colleagues, or querying databases. We observed that only case A has in-house visualization tools that provide complete answers to T_4, $CC_{1\text{-}4}$, $C_{1,3}$ and partial answers to T_3 and C_2 due to the fact that this company needs external tools to get a complete answer. On the other hand, ten information needs out of twenty-seven, such as $T_{5\text{-}6,8}$, $CC_{5\text{-}7,9}$, B_3, and $A_{2\text{-}3}$, can only be answered through manual inspections of output from tools.

Jenkins [1] is a widely adopted CI tool for addressing the identified information needs, followed by GitHub [2] and Jira [3]. Gerrit [4] was mentioned only once by case E to address CC_1. We asked participants about stakeholders that would be interested in knowing the answers to the identified information needs as represented in Table 4. We observed that none of the information needs belonged to just one stakeholder but several. Different stakeholders are connected to needs in different capacities. *"Compliance Authority"* is an external stakeholder that requires answers to certain questions before the product can be released to customers. This was only applicable to case E.

> To answer RQ2, we observed that out of the 5 investigated companies, only 1 company has dedicated resources to build in-house visualization tools. Unfortunately, practitioners revealed that their companies do not consider these efforts (e.g., building in-house tools) worthy and spend time exploring the options of third-party tools or plugins, resulting in more manual work.

6 QUANTIFYING INFORMATION NEEDS

Participants were asked to rate the identified information needs based on a Likert scale of 1 (Good to know), 2 (Of little importance), 3 (Moderately important), 4 (Important) and 5 (Very important). Table 5 summarizes the results of importance, frequency of their occurrence, effort and time with respect to each identified information need sorted in descending order. Eight (29%) information needs, as presented in Table 5, are marked as either "important" or "very important". Participants consider information needs such as $C_{1\text{-}3}$, $CC_{1,2,4,6}$, and B_3 significantly important. On the other hand, six (22%) information needs have been marked as either "good to know" or "of little importance". We did not see any pattern in these needs because these needs cover different activities such as test execution time (T_4), test case life cycle (T_8), internal release notes (CC_9), artifact related question ($A_{2\text{-}3}$) and employee performance (CC_7).

During the survey, we asked participants about how frequently they search for each identified information need. Participants were asked to rate identified information needs based on a Likert scale of 1 (Depends on context), 2(Rarely), 3(Occasionally), 4(Frequently) and 5(Very frequently). Seventeen needs (62%) were marked as either "frequently" or "very frequently" as represented by the bold text in Table 5. Three (11%) needs — which received a low ranking for importance — were marked as either "depends on context" or "rarely". All important needs (i.e., moderately to very important) were sought either "frequently" or "very frequently" in day-to-day to routines of the practitioners.

[2] https://github.com
[3] https://www.atlassian.com/software/jira
[4] https://www.gerritcodereview.com/

Table 4: Strategies, Tools and Stakeholders With Respect To Information Needs

		Current Handling Approach			Tools Strategies Used					Stakeholders				
Label	ID	In-house Tools	External Tools	Manual Inspection of Tools' output	Jenkins	GitHub	Gerrit	Jira	Manual (e.g., send email, call, etc.)	Development	Testing	Project Management	Release Team	Compliance Authority
T1	Which test cases/suites have been run on which product?		ABCD		ABCD					X	X	X	X	X
T2	Which test cases/suites have been run on which branch?		ABCD		ABCD					X	X	X	X	X
T3	In which environment/machine do specific test cases fail?	A		ABDE	BDE					X	X	X		X
T4	Which test suites' execution times have increased recently?	A								X	X	X	X	
T5	What are the build/test results of my commits?			ABDE	ABDE				ABDE	X		X		
T6	What are the unstable areas of the code that require more testing/attention?			BCE					BCE	X	X	X		
T7	Which of the test cases are flaky?		ABCDE	ABCDE	ABCDE				ABCDE	X	X	X	X	
T8	What is a test execution history of a specific test case?			DE	DE	DE			DE	X	X	X		
CC1	Does the final release to customers include my code?	A	E	BDC			E		BDC	X	X	X	X	
CC2	What is the status/health of new code changes?	A	BCE	BCE	BE				C	X	X	X	X	
CC3	Which requirement does the specific commit implements?	A	BDE		BDE			BDE		X	X	X	X	
CC4	Which change request does the specific commit implements?	A	BDE		BDE			BDE		X	X	X	X	
CC5	Is the given new feature implemented?			ABDE		AB			ABDE			X		
CC6	Is the given feature ready to release to customers?			ABCDE	AB				ABCDE	X		X	X	
CC7	How often does a specific employee deliver new code to the system?			AD					AD			X		
CC8	How has my code affected non-functional properties of the product?		B	BD					BD	X		X	X	
CC9	Which internal release notes have my comments/code?			AD					AD	X			X	
C1	How much confidence do we have in the release to deploy to the customers?	A		BDE	B	BDE			BDE	X	X	X	X	X
C2	How much confidence do we have in the test suite?	A	A	BDE	ABDE				BDE		X	X	X	X
C3	How much confidence do we have in stand-alone projects to be merged into the master branch/baseline?	A		BDE	B	BDE			BDE	X	X	X		
B1	Which bugs have been fixed in the specific release?	A		ABDE				A	BDE			X	X	
B2	How many bugs are still open with specific release?		ABCDE	ABCDE				ABCDE	ABCDE			X	X	X
B3	Is the bug fix ready to release to customers?			ABCDE	ABC			ABC	ABCDE	X		X	X	
B4	Who broke the build?		BDE	BDE	BDE	BDE				X	X	X		
A1	What tasks are pending in the pipeline for a long time?		AB	AB	AB				AB	X	X	X		
A2	When and why was this artifact created/modified?			CDE		CDE			CDE	X	X	X		
A3	Who created this artifact?			CDE		CDE			CDE	X	X	X		

This study attempts to capture the effort required to answer information needs. In this study, effort is regarded as sequences of steps that must be performed until the required information has been collected. Participants were asked to assign effort to information needs based on a Likert scale of 1 to 5 from "very easy" to "very difficult". Information needs that lie between "difficult" and "very difficult" are marked in bold in Table 5. Seventeen information needs (63%) were marked either as "difficult" or "very difficult". One can imagine the magnitude of difficulty practitioners faced due to the fact that all frequently sought needs were mentioned either as "difficult" or "very difficult". Only four (14%) needs were marked either "easy" or "very easy".

In addition to effort, this study attempts to capture how much time practitioners spend searching for information, as presented in Table 5. Twenty four (87%) information needs required more than 10 minutes to answer, whereas only 3 information needs required less than 10 minutes. The needs that were marked as "frequently", all take more than 10 minutes to address.

RQ3 We concluded that it is as equally critical to prioritize information needs based on their importance, frequency of their occurrence, effort involved in identifying them before developing, or selecting an appropriate tool. Information needs that are most important and sought frequently should be prioritized to improve productivity and save time.

7 DISCUSSION

The activity of identifying and cataloging information needs should serve as an opportunity to discuss each need critically among team members. Information needs such as $C_{1\text{-}3}$ require an understanding of the concept "confidence". For example, *"What constitutes as acceptable confidence for a software release?"*, *"Can we calculate confidence through automation or do we require human intervention?"*, or *"What information is required to achieve acceptable confidence in a software release and from what sources?"*. Similar discussions can be raised about $CC_{5\text{-}6}$ to understand *"What does the feature mean?"* or *"Is secure communication or standard compliance a feature?"*. These types of micro-level details will refine each information need further, thus creating a similar level of understanding among team members.

We observed that the identified information needs were known to practitioners. However, information about effective tools that

Table 5: Importance, Frequency, Effort and Time With Respect To Information Needs

ID	Information Need	Importance	Frequency	Effort	Time
C1	How much confidence do we have in the release to deploy to the customers?	4,8	**4,9**	**5,0**	15-20
CC6	Is the given feature ready to release to customers?	4,5	**4,6**	**5,0**	15-20
B3	Is the bug fix ready to release to customers?	4,5	**4,6**	**5,0**	15-20
C2	How much confidence do we have in the test suite?	4,1	**4,9**	**5,0**	**>20**
C3	How much confidence do we have in stand-alone projects to be merged into the master branch/baseline?	4,1	**4,7**	**4,8**	**>20**
CC2	What is the status/health of new code changes?	4,1	**4,6**	**4,8**	15-20
CC4	Which change request does the specific commit implements?	4,0	**4,7**	3,5	10-15
CC1	Does the final release to customers include my code?	4,0	3,7	2,5	5-10
T3	In which environment/machine do specific test cases fail?	3,8	**4,7**	**4,5**	>20
T7	Which test cases are flaky?	3,7	**4,7**	**5,0**	**>20**
CC5	Is the given feature implemented?	3,6	**4,6**	**4,5**	10-15
B1	Which bugs have been fixed in the specific release?	3,2	**4,3**	3,3	10-15
T5	What are the build/test results of my commits?	3,1	**4,7**	**5,0**	**>20**
CC3	Which requirement does the specific commit implements?	3,0	**4,7**	3,0	10-15
B4	Who broke the build?	3,0	3,5	**4,0**	**>20**
T6	What are the unstable areas of the code that require more testing/attention?	2,9	**4,3**	**5,0**	**20**
CC8	How has my code affected non-functional properties of the product?	2,7	**4,6**	3,4	**>20**
A1	What tasks are pending in the pipeline for a long time?	2,6	2,7	**4,0**	15-20
B2	How many bugs are still open with specific release?	2,4	2,6	2,0	10-15
T1	Which test cases/suites have been run on which product?	2,2	**4,3**	**5,0**	15-20
T2	Which test cases/suites have been run on which branch?	2,1	**4,3**	**5,0**	15-20
T4	Which test suite' execution times have increased recently?	1,9	3,6	1,0	10-15
T8	What is a test execution history of a specific test case?	1,5	3,6	1,0	10-15
CC9	Which internal release notes have my comments/code?	1,5	1,9	**5,0**	**>20**
A2	When and why was this artifact created/modified?	1,3	1,7	3,5	15-20
A3	Who created this artifact?	1,3	1,7	3,0	15-20
CC7	How often does a specific employee deliver new code to the system?	1,0	1,1	**5,0**	**>20**

can address those identified needs was scarce. Practitioners actively looked for tools such as visualization solutions that they expected could be useful. Most of the time, the tools only answer specific questions and are limited to containing a single source of information. Since the questions raised by the study participants require information from multiple sources, it is evident that multiple tools are required. We observed that questions asked by participants and the tools selected to provide answers were imperfect. Six information needs (e.g., $C_{1\text{-}3}$, $CC_{2,6}$, and B_3) were marked as the most important, frequently sought, and time-consuming to handle. Unfortunately, four ($C_{1,3}$, CC_6, and B_3) out of the six information needs can only be addressed through manual inspection of output from different tools as presented in Table 4. Jenkins, GitHub, and Jira were used by different companies to address these needs. Given the availability of different tools, practitioners may find it hard to determine which tools to use to seek answers to their information needs more effectively and efficiently. As participants revealed, all of the above-mentioned information needs take between 15-20 minutes to address under the current setup.

There are many challenges associated with the tool selection to address information needs. For example, software practitioners have specific information needs based on their designation and It is not an easy task to build a tool that can address all of the needs of multiple types of practitioners. In addition to this, each CI tool prefers a specific format of input and output format resulting in more difficulty in addressing information needs because one answer may require information from different CI logs or output. Identifying needs of the teams is a first step towards stable tools repertoire.

We noticed that company A utilizes an in-house visualization tool to fully address T_4, $CC_{1\text{-}4}$, $C_{1,3}$ and partially address T_3, C_2, we could argue that one of the solutions to the above-mentioned problems and challenges is encouraging in-house visualization tools. Unfortunately, only case A has dedicated resources to building a few in-house visualization tools, which underscores that companies do not pay attention to this area.

Continuous integration and delivery focuses on delivering the code changes to customers as fast as possible. Practitioners are spending significant time manually collecting scattered information about a software release such as confidence level, test suite passing/failing, open-bugs, stable areas, etc., which will delay the release. The answers to $T_{3,6}$ and A_1 provide answers to complete task at hand as well as a means for optimizing the CI pipeline in terms of performance and faster delivery.

The utility of continuous integration and delivery ensures faster delivery of software services to customers that is reflected in the first eight information needs as presented in Table 5. These needs involve the reason for code changes (CC_4), the status of the code changes (CC_2), the confidence in test suites (C_2), and whether the code changes or software release are ready to reach to the customers ($C_{1,3}$, $CC_{1,6}$, B_4). However, we consider it a challenge for practitioners to address these needs through automatic verification because these needs are simply irreducible to single commits and notoriously un-quantifiable.

8 IMPLICATIONS

We expect that our research provides context for tool development in CI. Practitioners can select from identified information needs, in this study, to raise discussions about the tools that can be applied in their CI infrastructure. By cataloging information needs and possible ways to address it, redundancy of similar information being sought can be reduced thus leading the efforts to be directed towards real tasks at hand. Researchers can use our work to investigate: what identified information needs mean to different practitioners? Besides, researchers can conduct further studies to capture what practitioners state and what they need.

9 VALIDITY THREATS

9.1 Internal Validity

An internal validity threat could be that participants did not understand the scope of the study or how to state their information needs. We tried to reduce this by conducting interviews in person, allowing us to explain the scope of the research and the expectations to the attendees.

9.2 Construct Validity

We tried to reduce the researchers' bias by involving all 3 researchers in the design of the interviews as well as in identification of information needs.

9.3 External Validity

We tried to eliminate the external validity threat by selecting five different companies that work in different domains. we cannot eliminate external validity completely.

10 CONCLUSION

The data produced in the continuous integration and delivery can provide significant insights such as teams performance, possible bottlenecks, or areas for improvements, etc. Providing an efficient and effective tool to visualize this data is a challenge. This challenge becomes more difficult when we do not know what information software professionals seek. In answering RQ1, we cataloged 27 of software professionals' information needs and categorized them as *testing*, *code & commit*, *confidence level*, *bug*, and *artifacts*. In addition to developers, many other software professionals such as managers, testers, etc., seek different information during their day-to-day routines. We observed that the identified needs are not aligned with the tools that are used to address them. Also, several information needs cannot be addressed through available tools, requiring manual inspections and thus taking more time (RQ2). Information needs that are related to *code & commit*, *confidence level*, and, *testing* are marked as most important and most frequently sought (RQ3). This study concluded that companies do not put enough effort into development resources for building in-house tools, thus spending a considerable amount of time selecting unsuitable outsourced tools or plugins.

ACKNOWLEDGMENT

The authors would like to thank the participants in our multiple case study for their availability to help us in data collection and data validation, and to ask clarifying questions about the data as well as the engaging and insightful discussions.

REFERENCES

[1] 2019. jenkinsci/eiffel-broadcaster-plugin. https://github.com/jenkinsci/eiffel-broadcaster-plugin original-date: 2019-01-22T15:04:13Z.

[2] Felipe Ebert, Fernando Castor, Nicole Novielli, and Alexander Serebrenik. 2018. Communicative Intention in Code Review Questions. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 519–523. https://doi.org/10.1109/ICSME.2018.00061 ISSN: 2576-3148.

[3] Martin S. Feather, Tim Menzies, and Judith R. Connelly. 2003. Matching Software Practitioner Needs to Researcher Activities. In *Proceedings of the Tenth Asia-Pacific Software Engineering Conference Software Engineering Conference (APSEC '03)*. IEEE Computer Society, USA, 6.

[4] Thomas Fritz and Gail C. Murphy. 2010. Using information fragments to answer the questions developers ask. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1 (ICSE '10)*. Association for Computing Machinery, Cape Town, South Africa, 175–184. https://doi.org/10.1145/1806799.1806828

[5] Jitendra Josyula, Sarat Panamgipalli, Muhammad Usman, Ricardo Britto, and Nauman Bin Ali. 2018. Software Practitioners' Information Needs and Sources: A Survey Study. In *2018 9th International Workshop on Empirical Software Engineering in Practice (IWESEP)*. 1–6. https://doi.org/10.1109/IWESEP.2018.00009 ISSN: 2333-519X.

[6] Andrew J. Ko, Robert DeLine, and Gina Venolia. 2007. Information Needs in Collocated Software Development Teams. In *Proceedings of the 29th international conference on Software Engineering (ICSE '07)*. IEEE Computer Society, USA, 344–353. https://doi.org/10.1109/ICSE.2007.45

[7] Thomas D. LaToza and Brad A. Myers. 2010. Developers ask reachability questions. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1 (ICSE '10)*. Association for Computing Machinery, Cape Town, South Africa, 185–194. https://doi.org/10.1145/1806799.1806829

[8] Thomas D. LaToza and Brad A. Myers. 2010. Hard-to-answer questions about code. In *Evaluation and Usability of Programming Languages and Tools (PLATEAU '10)*. Association for Computing Machinery, Reno, Nevada, 1–6. https://doi.org/10.1145/1937117.1937125

[9] Qingzhou Luo, Farah Hariri, Lamyaa Eloussi, and Darko Marinov. 2014. An empirical analysis of flaky tests. ACM Press, 643–653. https://doi.org/10.1145/2635868.2635920

[10] Akond Rahman, Asif Partho, Patrick Morrison, and Laurie Williams. 2018. What questions do programmers ask about configuration as code?. In *Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering (RCoSE '18)*. Association for Computing Machinery, Gothenburg, Sweden, 16–22. https://doi.org/10.1145/3194760.3194769

[11] Khaironi Yatim Sharif and Jim Buckley. 2009. Observation of Open Source programmers' information seeking. In *2009 IEEE 17th International Conference on Program Comprehension*. 307–308. https://doi.org/10.1109/ICPC.2009.5090071 ISSN: 1092-8138.

[12] Vibhu Saujanya Sharma, Rohit Mehra, and Vikrant Kaulgud. 2017. What Do Developers Want? An Advisor Approach for Developer Priorities. In *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 78–81. https://doi.org/10.1109/CHASE.2017.14

[13] Jonathan Sillito, Gail C. Murphy, and Kris De Volder. 2006. Questions programmers ask during software evolution tasks. In *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering (SIGSOFT '06/FSE-14)*. Association for Computing Machinery, Portland, Oregon, USA, 23–34. https://doi.org/10.1145/1181775.1181779

[14] Anselm Strauss and Juliet Corbin. 1998. *Basics of qualitative research: Techniques and procedures for developing grounded theory, 2nd ed.* Sage Publications, Inc, Thousand Oaks, CA, US.

[15] Daniel Ståhl, Kristofer Hallén, and Jan Bosch. 2017. Achieving traceability in large scale continuous integration and delivery deployment, usage and validation of the eiffel framework. *Empirical Software Engineering* 22, 3 (June 2017), 967–995. https://doi.org/10.1007/s10664-016-9457-1

[16] Robert K. Yin. 2009. *Case study research design and methods* (4th ed ed.). Thousand Oaks, Calif Sage Publications. https://trove.nla.gov.au/work/11329910