Received: 28 February 2021 | Accepted: 19 April 2021

DOI: 10.1049/sfw2.12030

IET Software

The Institution of Engineering and Technology WILEY

CASE STUDY

Data visualisation in continuous integration and delivery: Information needs, challenges, and recommendations

Azeem Ahmad | **Ola Leifler** | **Kristian Sandahl**

Department of Computer Science, Linköping University, Linköping, Sweden

Correspondence

Azeem Ahmad, Department of Computer Science, Linköping University, 581 83 Linköping, Sweden.
Email: azeem.ahmad@liu.se

Funding information

Linköping Universitet

Abstract

Several operations, ranging from regular code updates to compiling, building, testing, and distribution to customers, are consolidated in continuous integration and delivery. Professionals seek additional information to complete the mission at hand during these tasks. Developers who devote a large amount of time and effort to finding such information may become distracted from their work. We will better understand the processes, procedures, and resources used to deliver a quality product on time by defining the types of information that software professionals seek. A deeper understanding of software practitioners' information needs has many advantages, including remaining competitive, growing knowledge of issues that can stymie a timely update, and creating a visualisation tool to assist practitioners in addressing their information needs. This is an extension of a previous work done by the authors. The authors conducted a multiple-case holistic study with six different companies (38 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. 27 information needs associated with different stakeholders (i.e. developers, testers, project managers, release team, and compliance authority) were identified. The identified needs were categorised as testing, code & commit, confidence, bug, and artefacts. Apart from identifying information needs, practitioners face several challenges in developing visualisation tools. Thus, 8 challenges that were faced by the practitioners to develop/maintain visualisation tools for the software team were identified. The recommendations from practitioners who are experts in developing, maintaining, and providing visualisation services to the software team were listed.

1 | INTRODUCTION

Continuous integration and deployment is a widely adopted approach of committing, building, checking, and delivering improvements to customers on a regular basis. Aside from developers' contributions to continuous integration and delivery, testers often submit new test cases (unit, system, regression etc.) to test suites and track test execution results on code changes. On a case-by-case basis, stakeholders such as project managers, test leaders, team managers, and product owners, seek out the information generated by the continuous integration pipeline. 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 focussing on one kind of artefact—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 artefacts. 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. [1–4]. Software practitioners have different

information needs depending on their particular roles and responsibilities [1]. Although it is very important to catalogue and understand the questions and challenges faced by practitioners when attempting to answer those questions [3], little is known about (1) the information needs of practitioners [1, 2] and (2) the unfulfilled information needs in the field of software engineering [5]. Identifying these information needs can help us better understand the tools, practices, and processes that are important when addressing those information needs [1, 2]. Prior research has been conducted on the questions that were asked by practitioners regarding code [3], configurations as code [6], work support [7], evaluation tasks [4], and artefact traceability [8].

Another task is to assign priorities to information needs based on significance, frequency, and commitment once the true information needs have been identified. It is also vital to look into the nature of the established need, how often practitioners pursue this type of knowledge, and how much time it takes to respond to 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 utilised to address the information needs.

This study is an extension of our previous work [9]. After our previous work in information needs [9], we investigated various challenges that were faced by practitioners when designing visualisation tools or choosing from a wide variety of external (e.g. commercial or open-source) visualisation tools. A good visualisation tool improves decision making, ad hoc analysis of data, collaboration and communication between users as well as the return of investment [10]. The aim of this study is to recognise the challenges that practitioners faced while designing visualisation tools. We mapped the identified challenges to the identified information needs. In addition, we assembled a list of practitioner recommendations that can assist practitioners in developing and maintaining visualisation software.

This study attempts to answer the following questions:

RQ1 What are practitioners' information needs in continuous integration and deployment?

[RQ1.1:] To what extent are software tools utilised in the industry to address the identified information needs?

[RQ1.2:] Do practitioners assign priorities to identified information needs based on importance, frequency and effort?

RQ2 What challenges are faced by the practitioners who develop and maintain visualisation tools for the software team?

[RQ2.1:] To what extent can the identified challenges in RQ2 be mapped to the identified information needs in RQ1?

RQ3 What are the recommendations from practitioners that develop and maintain visualization tools for software teams concerning challenges, identified in RQ3?

The detailed contributions of this work are as follows:

C1 We identified practitioners' information needs in continuous integration and deployment. (Section 4–RQ1).
C2 We investigated the extent (i.e. complete, partial or none) to which the availability of tools supported the identified needs (Section 5–RQ1.1).
C3 We investigated the following questions related to the identified information needs: (1) Primary stakeholders related to each need, (2) How frequently these identified needs were sought by the participants, (3) How long it takes 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–RQ1.2).
C4 We investigated the challenges faced by the practitioners who develop and maintain visualisation tools for software teams (Section 7–RQ2).
C5 We investigated to what extent the identified challenges in RQ2 can be mapped to the identified information needs in RQ1 (Section 8–RQ2.1).
C6 We list the recommendations shared by the practitioners who develop and maintain visualisation tools for software teams concerning challenges, identified in RQ2 (Section 9–RQ3).

Section 2 presents related work. Research methods are discussed in Section 3 followed by identified information needs in Section 4. Detailed information about strategies, tools, and stakeholders concerning identified information needs is presented in Section 5. Different attributes such as importance, frequency, effort, and time with respect to each identified information need are discussed in Section 6. Identified challenges and their mapping to information needs are presented in Section 7 and 8, followed by the recommendations in Section 9. Discussion, implication, validity threat and conclusion are presented in Sections 10, 11, 12, and 13, respectively.

2 | RELATED WORK

Researchers [1–3, 6, 7, 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 [6], while performing development activities [2, 3, 7, 11] and during software evolution [4]. Ebert et al. concluded that all of the questions developers ask in their day-to-day routines serve information-seeking goals [12]. 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 [3]. In another study, LaToza and Brad categorised 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?) [13]. Another work to understand the information needs of developers, during software

TABLE 1 Detailed information about case companies

Case	Business sector	Employees	Claimed continuous integration maturity level[a]
A	Network equipment	25k	High
B	Surveillance equipment	5k	High
C	Water equipment	3k	Medium
D	Medical equipment	3k	High
E	Software modelling	1k	High
F	Industrial manufacturing	300k	High

[a]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.

evolution, was conducted by Sharma et al. in which the authors asked 25 developers to prioritise 27 pre-defined information needs [14]. Also, they proposed the 'Smart Advisor' approach to provide automated advice regarding insights about important queries by analysing data generated during the software development life cycle [14]. Josyula et al. conducted interviews with 17 practitioners to identify information needs related to the software development life cycle [1]. The most frequented information needs were related to clarification of the requirement, design of the products, and skills in programing languages [1].

Fritz et al. [7] developed the information fragment model, which collected information from different sources. This model helped developers answer 47 pre-identified questions. Sillito et al. in [4] 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.

3 | RESEARCH METHODS

For this work, we presented a multiple-case holistic study [15] 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 [15]. We picked (a) so that results from different cases would complement each other to identify the true information needs of practitioners. Yin [15] 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 labelled from A to F. We attempted to study different business sectors to avoid generalisation 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.

3.2 | Data collection, preparation, analysis and validation

Data collection, analysis, and validation were conducted in two phases as shown in Figure 1. During phase 1, we identified and validated information needs, their importance, frequency, required effort, the current approach of handling the needs, and associated stakeholders. During phase 2, we identified challenges that practitioners face while developing visualisation tools to address identified information needs. We explain each phase in detail in the sections below.

3.2.1 | Phase 1

We collected data by conducting individual interviews with 13 software practitioners from case A to E. These participants had different roles in the same company. Table 2(row 1) presents detailed information about the participants in phase 1. 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 min long individual semi-structured interviews[1] to collect data from all the 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

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

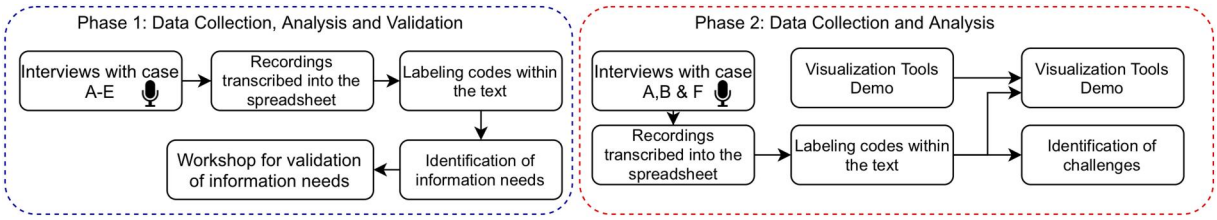

FIGURE 1 Data collection, analysis and validation: Phase 1 & 2

TABLE 2 Data collection

Phase	Case	Partic.	Total exp. (Yr)	Designations
1	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
2	A	P_{14}	8	CI architect, quality manager, test manager
	B	P_{15}	10	CI architect, CI maintainer
	B	P_{16}	10	Test architect, CI architect, project manager
	F	P_{17}	8	CI architect, CI maintainer
	F	P_{18}	7	CI architect, CI maintainer

during a conversation. The names of the participants were also anonymised 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* [16]. Each paragraph of the transcript of each case company was studied to determine what was said and each paragraph was labelled with one or more codes. Later, we compared all the paragraphs from the different case companies to collect similar codes (axial codes). These codes were used to categorise information needs. Most of the time, the participants shared the information need directly, for example '*Which change requests does a specific commit serve?*' whereas at 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 min. 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 the 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.

3.2.2 | Phase 2

On a separate occasion, to identify the challenges we collected data through individual interviews with five software practitioners from cases A, B, and F. In addition, we asked participants to demonstrate the visualisation tools they have been using/

TABLE 3 Cataloguing information needs from different texts

Original text 1	'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
Original text 2	'Each teams has different workflows of CI for their products. Different activities in their workflow that might require different UI. The challenge is to provide them visualisation according to their workflows'
Identified challenge	Different CI WorkFlows

developing in their team. These participants were responsible for providing visualisation support to developers and testers. Table 2 (row 2) presents detailed information about the participants in phase 2. A similar protocol (i.e. phase 1 above) was used for conducting, recording, analysing, reporting and validating the data/findings. Table 3 presents an example of how we extracted information needs and corresponding categories from original texts.

4 | INFORMATION NEEDS-RQ1

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

4.1 | Testing

- (T_1) *Which test cases/test suites have been run on which product?* ($P_{1–4,7–10}$)
- (T_2) *Which test cases/test suites have been run on which branch?* ($P_{1–4,7–10}$)
- (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?* ($P_{1–3,9,11,12}$)
- (T_5) *What are the build/test results of my commits?* ($P_{1–3,5,6,8,11}$)
- (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 the test execution history of a specific test case?* ($P_{9,12,13}$)

Testing practitioners in the investigated companies consider planned versus 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 a value 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 has not been executed, developers spend a considerable amount of time 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 a 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 the 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 visualisation 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 [17] (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?* ($P_{1–13}$)
- (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?* ($P_{2,6,7–9,10–13}$)
- (CC_4) *Which change request does the specific commit implement?* ($P_{2,6,7–9,10–13}$)

(CC_5) *Is the given new feature implemented?* ($P_{1–3,5,6,9,10}$)
(CC_6) *Is the given feature ready to release to customers?* ($P_{1–3,5,6,9,10}$)
(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?* ($P_{1–3}$)
(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 (*$CC_{3.4}$*)'*. 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 in knowing 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 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?* ($P_{1–6,\ 9,10}$)
(C_2) *How much confidence do we have in the test suites?* ($P_{3,4,7–10}$)
(C_3) *How much confidence do we have in stand-alone projects to be merged into the master branch/baseline?* ($P_{11–13}$)

Attaining confidence in a software release is related to many factors such as planned versus 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 artefact under consideration. Interpreting the data and attaining confidence are separate processes.

4.4 | Bug

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

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 the bugs to gain customers' confidence and improve quality. In addition to fixed bugs, managers are 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 the code changes, resulting in build failure. Is it only one person or a series of events that caused the build failure?

4.5 | Artefacts

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

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 the source code and the test cases are software artefacts, in our investigation artefacts also refer to build scripts, binary files, or simulation models. Practitioners want to know who created this artefact, when, and, why it was created ($A_{2\text{-}3}$) so that the queries can be directed to the relevant persons. Practitioners also want to know how a specific stage was reached (missing or duplicated artefacts).

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 Artefacts.

5 | STRATEGIES, TOOLS AND STAKEHOLDERS WITH RESPECT TO INFORMATION NEEDS-RQ2

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 Visualisation'* refers to the visualisation tools that have been built by the company's IT department to address the information needs. *'External Visualisation 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 visualisation 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, 10 information needs out of 27, such as $T_{5,6,8}$, $CC_{5\text{-}7,9}$, B_3, and $A_{2\text{-}3}$, can only be answered through manual inspections of the output from the tools. A general summary to increase readability about information needs, their stakeholders and how it is addressed is presented in Table 5.

Jenkins [18] 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. They belonged to several stakeholders. Different stakeholders are connected to the 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 five investigated companies, only 1 company has dedicated resources to build in-house visualisation 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-RQ3

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 6 summarises 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 6, 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), artefact 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). 17 needs (62%) were marked as either 'frequently' or 'very frequently' as represented by the bold text in Table 6. 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 the day-to-day to routines of the practitioners.

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

[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 the 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 implement?	A	BDE		BDE			BDE		X	X	X	X	
CC4	Which change request does the specific commit implement?	A	BDE		BDE			BDE		X	X	X	X	
CC5	Is the given new feature implemented?			ABDE		AB			ABDE			X		

TABLE 4 (Continued)

		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**
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 the 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 artefact created/modified?			CDE		CDE			CDE	X	X	X		
A3	Who created this artefact?			CDE		CDE			CDE	X	X	X		

TABLE 5 A summary of information needs concerning who and how. The 'who' represents the companies that have the information needs. The 'how' represents the means of answering the information needs. The symbol ^represents the word 'except'

ID	Information need	Who	How
T1	Which test cases/suites have been run on which product?	All, ^E	External
T2	Which test cases/suites have been run on which branch?	All, ^E	External
T3	In which environment/machine do specific test cases fail?	All, ^C	Manual, ^A (in-house)
T4	Which test suites' execution times have increased recently?	A	In-house
T5	What are the build/test results of my commits?	All, ^C	Manual
T6	What are the unstable areas of the code that require more testing/attention?	All, ^BD	Manual
T7	Which of the test cases are flaky?	All	External & manual
T8	What is the test execution history of a specific test case?	All, ^ABC	Manual
CC1	Does the final release to customers include my code?	All	A (in-house), B (external), CDE (manual)
CC2	What is the status/health of new code changes?	All, ^D	A (in-house), BCE (external & manual)
CC3	Which requirement does the specific commit implement?	All, ^C	A (in-house), BDE (external)
CC4	Which change request does the specific commit implement?	All, ^C	A (in-house), BDE (external)
CC5	Is the given new feature implemented?	All, ^C	Manual
CC6	Is the given feature ready to release to customers?	All	Manual
CC7	How often does a specific employee deliver new code to the system?	AD	All manual
CC8	How has my code affected non-functional properties of the product?	BD	B (manual), D (external)
CC9	Which internal release notes have my comments/code?	AD	Manual
C1	How much confidence do we have in the release to deploy to the customers?	All, ^C	A (in-house), BDE (manual)
C2	How much confidence do we have in the test suite?	All, ^C	A (in-house), BDE (manual)
C3	How much confidence do we have in stand-alone projects to be merged into the master branch/ baseline?	All, ^C	A (in-house), BDE (manual)
B1	Which bugs have been fixed in the specific release?	All, ^C	A (in-house), BDE (manual)
B2	How many bugs are still open with the specific release?	All, ^C	A (in-house), BDE (manual)
B3	Is the bug fix ready to release to customers?	All	Manual
B4	Who broke the build?	All, ^AB	A (in-house), external
A1	What tasks are pending in the pipeline for a long time?	AB	A (in-house), external
A2	When and why was this artefact created/modified?	All, ^AB	Manual
A3	Who created this artefact?	All, ^AB	Manual

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 6. 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 the effort, this study attempts to capture how much time practitioners spend searching for information, as presented in Table 6. Twenty four (87%) information needs required more than 10 min to answer, whereas only three information needs required less than 10 min. The needs that were marked as 'frequently', all take more than 10 min to address.

RQ3 We concluded that it is equally critical to prioritise 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 prioritised to improve productivity and save time.

TABLE 6 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 implement?	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 implement?	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 the 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 suites' execution times have increased recently?	1,9	3,6	1,0	10-15
T8	What is the 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 artefact created/modified?	1,3	1,7	3,5	15-20
A3	Who created this artefact?	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**

Note: The information needs were sorted based on importance column from high values (5) to low values (1). The values in second column (frequency) that lie within 4-5 were made bold to draw user attention. We added a text in section 6 para 2 (i.e., Seventeen needs …..). Similar justification is for column 3 and 4. The paragraphs are added in section 6.

7 | CHALLENGES IN PROVIDING VISUALISATION TOOLS-RQ4

We identified 8 challenges in this study from cases A, B, and F. We present, in the following section, the participants' quotes when describing the challenges in detail. Table 7 presents the mapping between the identified challenges and the investigated companies. CH1–5 was mentioned by all case companies, CH6-7 was mentioned by two case companies (i.e. A and F), and CH8 was mentioned by only one case company (i.e. B).

7.1 | Visualisation tools' development & maintenance

It is equally challenging to develop and maintain visualisation tools as business products. Unfortunately, due to the difficulty in determining whether to create an in-house solution or explore online resources (i.e. open-source, commercial off-the-shelf etc.) visualization tools receive little attention and resources. In contrast to developers and testers, it has been found that CI architects or maintainers in companies are in short supply. One or two people are in charge of delivering tools (visualisation) and

TABLE 7 Mapping between identified challenges and investigated companies

ID	Identified challenges	A	B	F
CH1	Visualisation tools' development & maintenance	[✓]	[✓]	[✓]
CH2	Information quality & trust in tooling	[✓]	[✓]	[✓]
CH3	Tool creators versus tool users	[✓]	[✓]	[✓]
CH4	Effort versus worth	[✓]	[✓]	[✓]
CH5	Confidentiality versus transparency	[✓]	[✓]	[✓]
CH6	Different CI work flows	[✓]	-	[✓]
CH7	Result interpretations	-	[✓]	[✓]
CH8	User interface complexities	-	[✓]	-

services to 12 separate software teams, each of which has a large number of developers, testers, product owners, and managers, as stated by one of the participants:

> Every day we have many requests from different teams about some results, they want to explore a product under investigation. We do not know why developers do not help us in developing visualisation tools. [….] We need to send email back and forth to know their requirements and deploy the tool

In addition, practitioners stated that visualisation tools are developed as ad hoc solutions for developers and testers. Thus, they did not maintain it properly over time. One of the participants reported

> […] this visualisation tool was developed around eight years ago, but it is very complex to modify at the moment. It is just providing some answers and developers are still using it. It is a challenge to maintain it with new requirements

7.2 | Information quality & trust in tooling

Information quality refers to *'The fitness for use of the information provided [19]'*. The struggle to gain the trust of developers or testers in visualisation tools built by other teams was defined by practitioners.:

> We receive emails [during release] from users with screenshots inquiring that the information, they see, is correct. We receive too many emails [during release time] from teams. The quality of information and gaining user trust is very important

When the visualisation tool is newly developed, the task of gaining trust becomes much more difficult, as stated by one of the participants:

> If the visualisation tool is newly developed, developers do not just start using it, and you say it is very good, but you have to show them and they have to see the value by themselves

Unfortunately, developers and testers prefer to trust their gut instincts over the tool's output when making decisions. Most of the time, team members talk to each other to reach a decision. Another participant shared,

> Developers have a hard time in trusting the tooling developed by other team members. It is easy for them to talk to people about specific problems and reach a conclusion […\enleadertwodots] If the tool answers it, it will be faster but I am not sure if developers trust it. This is my opinion, coming from quality and CI department

Another participant shared a similar experience:

> Even if they see some visualisation of results, it is still talking to other people and gut feelings before releasing the product. We need to trust the visualisation system

7.3 | Tool creators versus tool users

Participants agreed that building a bridge between tool users and tool developers is important. This lack of communication between two types of users affects the feedback loop, as mentioned by one of the participants:

> When we configure a specific visualisation for a specific team, our job is done as long as they need more. We do not ask for feedback and they do not provide it

A similar experience was shared by another participant,

> It is difficult because I am a provider of information that someone else looked at. I do not get feedback from them whether it was a smooth ride or there were issues. Perhaps they use the system occasionally or they are not using it at all

The lack of a feedback loop and communication differences between tool designers and tool users, according to participants, have an effect on the expectations from the tools. It is a challenging task to understand what developers want, as mentioned by one of the participants:

> Sometimes, we do not have clear requirements for a tool to be developed. For example, we need to provide specific confidence values about some

> artefacts. In this case, we need to talk to lots of people. It is like a pyramid. You start talking to some people and as you go down, you talk to more and more people to understand how they work, how they decide confidence. Then we make a tool

Another participant shared,

> Developer do not know how the visualisation tool works [what are data sources or algorithms] that we provided. I think we should work together. This way, we can develop an effective tool

7.4 | Effort versus worth

Since we develop different visualisation tools for different teams, the effort spent should be worthwhile, as stated by one of the participants:

> We put effort [in developing tools] in what type of testing/branch it is about. For unit testing done at the developer side, we do not need a visualisation for it, similarly nothing for manual system testing somewhere. We prefer to provide visualisation on automatic regression testing. We need to evaluate if the efforts put in tooling worth it

Teams working on a range of products, environments, and hardware have varying needs for visualisation software. Allocating or prioritising CI team resources for teams' requests is challenging. One of the participants shared,

> 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

7.5 | Confidentiality versus transparency

One of the challenges stated by the CI architects is maintaining a balance of confidentiality and transparency. When providing answers to some questions, we need to verify if users are authorised for it, as stated by one of the participants:

> Developers might not be aware of what they are coding for. They have some perspective it cannot go all way up to the abstraction layer. They think that they improving an algorithm but they do not know that we are deploying this product to specific customers who have specific and confidential requirements

Many developers want to know which release or customer their code will be included in. Sometimes, we are unable to have a visual representation of their commits in relation to the final product, as stated by one of the participants:

> We do not want them to know about the actual use case that we are supporting. This is why developers cannot see where their feature [code changes] end up actually

This challenge is associated with all software activities such as requirements, development, testing, issue management, and release management, as shared by another participant:

> The teams cannot be aware of a change request for some of the clients or products so developing a visualisation tool to keep these types of information confidential is a challenge. These scenarios also limit the use of tools because developers will not use them because their perspective is narrow

7.6 | Different CI work flows

Different teams inside large organisations work on various items. Each team has the right to select their own CI pipeline from a variety of activities. Configuring visualisation software for various CI workflows is difficult. One of the participants stated,

> Each team has a different workflow of CI for their product. Different activities in their workflows that might require different UI. The challenge is to provide them visualisation according to their workflows and needs

Configuring the visualisation tool for software teams takes time, and we have to redo the work when they adjust their CI pipeline, as said by one of the participants:

> It is very difficult to configure the visualisation tool. It takes a lot of time. The configuration is very technical and requires technical and previous knowledge about the CI pipeline. Teams who struggle to develop a stable pipeline increase our work

The visualisation tools should be able to dynamically customise depending on the needs of the team, according to the participants. Furthermore, linking numerous data sources to visualisation is a difficult job, as stated by one of the participants:

> Where the data is coming from is also a major challenge, some information came from issue management system which should be linked with requirement which is another database. We also

> need information from GitHub […] so most of the times, the input source is distributed and combining them and getting faster results is a challenge

7.7 | Result interpretations

When the tool was built by another team, the main difficulty for the developers was interpreting the results they saw on the screen. If a result cannot be traced back to its source, it can be interpreted in a number of ways. Another explanation may be that you are looking at findings without any meaning or background details. One of the participants reported,

> Sometimes, the tool just provides the results but it is a team's responsibility to interpret them. The quality information team provides only the information but they do not interpret results. They do not say much about it

Another participant shared his experiences regarding result traceability:

> The tool we have today provides a good overview like a final status but they are not good for traceability. The current tool requires some additional knowledge that is not there. There is no traceability in the current tool such as what was executed before this stage

The tool should provide sufficient information for interpretation, as mentioned by one of the participants:

> I would like to have a tool wherever I look for results, I would like to have a button or link that will take me to a description of the environment, about the machine, the configuration of the machine or the hardware under test, so I do not do the detective work

7.8 | User interface complexities

Project managers, during release, want to access information about the product/software such as what is included, how many test cases were failed, what is the confidence level of release, what is a baseline code for this release etc. Choosing what information can appear on the screen and how much flexibility we should have in tracking it back to its origins is a challenging job, as mentioned by one of the participants:

> It is hard to determine how much information should be provided on the screen. You can have higher-level information and when the user clicks, you go deep and deep. How many levels should be there to view the information? We are afraid that it can make a tool very complex to use and read […]

Another experience was shared by one of the participants about the information on-screen and the types of users (i.e. technical and non-technical users):

> Since the data produced by CI pipeline is enormous, it is difficult to develop a simple visualisation tool for a non-technical person such as high-level mangers and complex tool for technical person. […] What information should be updated and what can be old is a major decision. Most users do not know what they want

8 | MAPPING OF IDENTIFIED CHALLENGES WITH IDENTIFIED INFORMATION NEEDS

The first step towards data visualisation is to define the needs. It is also vital to comprehend the difficulties that practitioners encountered in creating and sustaining the visualisation software. With the help of practitioners, we mapped the identified challenges (i.e. Section 7) with the identified information needs (Section 4). Table 8 presents the mapping between identified information needs and identified challenges. Only five challenges (i.e. CH2, 3, 5, 7, 8) were mapped to information needs because the other three (i.e. CH 1, 4, 6) were too broad to be related to specific information needs. Practitioners stated that these three challenges might apply to all of the information needs or none of them, as shown in Table 8.

We observed two outliers in Table 8. First, all applicable challenges were mapped to six information needs (i.e. $C_{1–3}$ and $B_{1–3}$). These information needs are related to the confidence level in artefacts and bugs. On the other hand, there were only two information needs (i.e. $CC_{7,9}$) that were not mapped with any of the applicable challenges. This is why these two information needs were given such low priority and frequency, as shown in Table 6.

As can be seen in Table 8, three identified challenges (i.e. $CH_{1,\ 2,\ 8}$) were mapped to approximately all the information needs. However, challenges such as *confidentiality* versus *transparency* and *result interpretation* were mapped to 11 and 15 information needs, respectively.

9 | RECOMMENDATIONS

In this section, we list the recommendations from practitioners who develop and maintain visualisation tools for software teams concerning the challenges identified in RQ3.

R1 Developers have a reason to trust the tool if it offers indications about (1) the source of information, (2) how old the information is, and other information quality properties.

TABLE 8 A mapping of identified information needs with identified challenges. The symbol '✓' represents that the identified challenge applies to the identified information need. The symbol '×' represents that the identified challenge does not apply to the identified information need. The symbol '-' represents that the identified challenge is very general and cannot be linked to specific identified information needs

		Challenges							
ID	**Information need**	**Visualisation tools development & maintenance (CH1)**	**Information quality & trust in tooling (CH2)**	**Tool creators versus tool users (CH3)**	**Effort versus worth (CH4)**	**Confidentiality versus transparency (CH5)**	**Different CI work flows (CH6)**	**Result interpretations (CH7)**	**User interface complexities (CH8)**
T1	Which test cases/suites have been run on which product?	-	✓	✓	-	×	-	✓	×
T2	Which test cases/suites have been run on which branch?	-	✓	✓	-	×	-	✓	×
T3	In which environment/machine do specific test cases fail?	-	✓	✓	-	×	-	✓	×
T4	Which test suites' execution times have increased recently?	-	✓	✓	-	×	-	×	×
T5	What are the build/test results of my commits?	-	✓	✓	-	×	-	×	×
T6	What are the unstable areas of the code that require more testing/attention?	-	✓	✓	-	×	-	×	✓
T7	Which of the test cases are flaky?	-	✓	✓	-	×	-	×	×
T8	What is the test execution history of a specific test case?	-	×	×	-	×	-	×	✓
CC1	Does the final release to customers include my code?	-	×	×	-	✓	-	×	×
CC2	What is the status/health of new code changes?	-	✓	✓	-	×	-	×	✓
CC3	Which requirement does the specific commit implement?	-	×	✓	-	✓	-	×	×
CC4	Which change request does the specific commit implement?	-	×	✓	-	✓	-	×	×
CC5	Is the given new feature implemented?	-	✓	✓	-	✓	-	×	✓
	CC6	Is the given feature ready to release to customers?	-	✓	✓	-	✓	-	×
✓									
CC7	How often does a specific employee deliver new code to the system?	-	×	×	-	×	-	×	×
CC8	How has my code affected non-functional properties of the product?	-	✓	×	-	×	-	✓	✓

(Continues)

TABLE 8 (Continued)

		Challenges							
ID	**Information need**	**Visualisation tools development & maintenance (CH1)**	**Information quality & trust in tooling (CH2)**	**Tool creators versus tool users (CH3)**	**Effort versus worth (CH4)**	**Confidentiality versus transparency (CH5)**	**Different CI work flows (CH6)**	**Result interpretations (CH7)**	**User interface complexities (CH8)**
CC9	Which internal release notes have my comments/code?	-	×	×	-	×	-	×	×
C1	How much confidence do we have in the release to deploy to the customers?	-	✓	✓	-	✓	-	✓	✓
C2	How much confidence do we have in the test suite?	-	✓	✓	-	✓	-	✓	✓
C3	How much confidence do we have in stand-alone projects to be merged into the master branch/baseline?	-	✓	✓	-	✓	-	✓	✓
B1	Which bugs have been fixed in the specific release?	-	✓	✓	-	✓	-	✓	✓
B2	How many bugs are still open with the specific release?	-	✓	✓	-	✓	-	✓	✓
B3	Is the bug fix ready to release to customers?	-	✓	✓	-	✓	-	✓	✓
B4	Who broke the build?	-	×	✓	-	×	-	×	×
A1	What tasks are pending in the pipeline for a long time?	-	✓	✓	-	×	-	×	✓
A2	When and why was this artefact created/modified?	-	✓	✓	-	×	-	×	✓
A3	Who created this artefact?	-	✓	✓	-	×	-	×	✓

R2 To avoid being a legacy system, the visualisation tool should include details such as requirements' specifications, design documents, and so on, much like conventional product development activities.
R3 It is recommended to develop a base architecture upon which different visualisation tools can be built. Teams used varieties of open-source software (e.g. GitHub, Gerrit, Jenkins etc.); thus, developing a base protocol similar to Eiffel [8] is encouraged.
R4 The credibility of a tool grows over time. They [developers] will trust it if they use it regularly.
R5 It is preferred for software teams to develop similar and systematic CI pipelines (e.g. following the company's guidelines) to reduce the work of configuring visualisation systems.
R6 Maintain a software visualisation repertoire to cut down on redundancies and improve re-usability.
R7 Bespoke development of visualisation tools should be discouraged to reduce the risk of superfluous tools.
R8 Any proposal for a change in visualisation requirements should go through the required change management teams for real-time assessment.
R9 Resources such as CI architects or maintainers should be increased, depending on the company's needs and structure. The balance of a workload between those who develop the business products and those who provide services to help develop a quality product should be maintained.
R10 The tool users should provide continuous feedback to visualisation tool developers for effective and efficient future decisions. In addition, these continuous communications would reduce the risk of misinterpretation of results.

10 | DISCUSSION

10.1 | Information needs

The activity of identifying and cataloguing information needs should serve as an opportunity to discuss each need critically among team members. Information needs such as C_{1-3} 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_{5-6} 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.

Continuous integration and delivery focusses 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 the complete task at hand as well as a means for optimising 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 6. 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 releases 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 unquantifiable.

10.2 | In-house/external tool & challenges

We observed that the identified information needs were known to practitioners. However, information about effective tools that can address those identified needs was scarce. Practitioners actively looked for tools such as visualisation solutions that they expected to 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_{1-3}, $CC_{2,6}$, and B_3) were marked as 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 the 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 and 20 min to address under the current setup.

There are many challenges associated with 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 the needs of multiple types of practitioners. In addition to this, each CI tool prefers a specific input and output format resulting in increased difficulty in addressing information needs because one answer may require information from different CI logs or output. Identifying the needs of the teams is the first step towards the repertoire of stable tools.

We noticed that company A utilises an in-house visualisation tool to fully address T_4, CC_{1-4}, $C_{1,3}$ 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 visualisation tools. Unfortunately, only case A has dedicated resources for building a few in-house visualisation tools, which underscores that companies do not pay attention to this area.

Whether it is an external or in-house tool development, we noticed that all the challenges are applicable in both scenarios. For example, practitioners still need to maintain the internal/external visualisation tool (i.e. *'Visualisation Tools' Development & Maintenance [CH1]'*) which may or may not be a hard task. Given an external tool, the maintenance task can be harder due to a lack of expertise to maintain it. No matter where the tool is developed, in-house or external, developers will have a hard time trusting the information provided by the tool (i.e. *'Information Quality & Trust in Tooling [CH2]'*). The trust is gradually built over time when the interaction frequency between the tool and tool users is increased. Practitioners procrastinate whether exploring the options of third-party tools/plugins or developing in-house tools is worth their time or not (i.e. *'Effort vs Worth [CH4]'*). Companies that are new to open-source software might find this software confusing, which can be an obstacle to the technology's adoption [20]. External tools are developed for a broader audience, thus requiring additional attention and information related to confidentiality of information being presented on the tool (i.e. *'Confidentiality vs Transparency [CH5]'*). The challenges *'Different CI Work Flows'*, *'Result Interpretations'*, and *'User Interface Complexities'* are applicable to internal or external tools.

11 | IMPLICATIONS

We expect our research to provide 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 cataloguing information needs and possible ways to address it, the redundancy of similar information being sought can be reduced, thus leading the efforts to be directed towards real tasks at hand. We also provide challenges and recommendations to practitioners in case they are struggling with data visualisation in CI/CD. We expect that practitioners can consider these recommendations in their companies. 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. In addition, the validation of our challenges and recommendations in other contexts are highly welcome.

12 | VALIDITY THREATS

12.1 | Internal validity

An internal validity threat could be that participants did not understand the scope of the study, such as how to state their information needs, challenges, or recommendations. 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.

12.2 | Construct validity

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

12.3 | External validity

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

13 | CONCLUSION

The data produced in the continuous integration and delivery can provide significant insights such as a team's performance, possible bottlenecks, or areas for improvements etc. Providing an efficient and effective tool to visualise this data is a challenge. This challenge becomes more difficult when we do not know what information software professionals seek. In answering RQ1, we catalogued 27 software professionals' information needs and categorised them as *testing*, *code & commit*, *confidence level*, *bug*, and *artefacts*. 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). We identified 8 challenges that were faced by the practitioners in developing or maintaining the visualisation tools. Ten recommendations of practitioners were enlisted in this study for those who are struggling with visualisation issues.

ACKNOWLEDGMENT

The authors would like to thank the participants in our multiple case study for their availability and help in data collection and data validation, in asking clarifying questions about the data as well as in engaging and insightful discussions. This work was supported by Linkoping University and project 18 on Data Visualisation in CI/CD of Software Centre.

ORCID

Azeem Ahmad https://orcid.org/0000-0003-3049-1261

REFERENCES

1. Josyula, J., et al.: Software practitioners information needs and sources: a survey study. In: Proceedings of the 2018 9th International Workshop on Empirical Software Engineering in Practice (IWESEP), pp. 1–6. ISSN: 2333-519X. Nara (2018). https://doi.org/10.1109/IWESEP.2018.00009
2. Ko, A.J., DeLine, R., Gina, V.: Information needs in collocated software development teams. In: Proceedings of the 29th International Conference on Software Engineering (ICSE'07), pp. 344–353. IEEE Computer Society, USA (2007). https://doi.org/10.1109/ICSE.2007.45
3. LaToza, T.D., Myers, B.A.: Developers ask reachability questions. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE'10), vol. 1. pp. 185–194. Association for Computing Machinery, Cape Town, (2010). https://doi.org/10.1145/1806799.1806829
4. Sillito, J., Murphy, G.C., De Volder, K.: 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), pp. 23–34. Association for Computing Machinery, Portland (2006). https://doi.org/10.1145/1181775.1181779
5. Feather, M.S., Menzies, T., Connelly, J.R.: Matching software practitioner needs to researcher activities. In: Proceedings of the Tenth Asia-Pacific Software Engineering Conference Software Engineering Conference (APSEC'03), vol. 6. IEEE Computer Society, USA (2003)
6. Rahman, A., et al.: What questions do programmers ask about configuration as code? In: Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering (RCoSE'18), pp. 16–22. Association for Computing Machinery, Gothenburg (2018). https://doi.org/10.1145/3194760.3194769
7. Fritz, T., Gail, C.M.: Using information fragments to answer the questions developers ask. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE'10), vol. 1, pp. 175–184. Association for Computing Machinery, Cape Town (2010). https://doi.org/10.1145/1806799.1806828
8. Ståhl, D., Hallén, K., Bosch, J.: Achieving traceability in large scale continuous integration and delivery deployment, usage and validation of the eiffel framework. Empir. Software Eng. 22, 967–995 (2017). https://doi.org/10.1007/s10664-016-9457-1
9. Ahmad, A., Leifler, O., Sandahl, K.: The 36th ACM/SIGAPP Symposium on Applied Computing, March 22–26, 2021 (Virtual Event). Republic of Korea (2021). https://doi.org/10.1145/3412841.3442026
10. Cota, M.P., et al.: Massive data visualization analysis analysis of current visualization techniques and main challenges for the future. In: Proceedings of the 2017 12th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–6. Lisbon (2017). https://doi.org/10.23919/CISTI.2017.7975704
11. Sharif, K.Y., Buckley, J.: Observation of Open Source programmers' information seeking. In: Proceedings of the 2009 IEEE 17th International Conference on Program Comprehension, pp. 307–308. ISSN: 1092-8138. Vancouver (2009). https://doi.org/10.1109/ICPC.2009.5090071
12. Ebert, F., et al.: Communicative intention in code review questions. In: Proceedings of the 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 519–523. ISSN: 2576-3148. Madrid (2018). https://doi.org/10.1109/ICSME.2018.00061
13. LaToza, T.D., Myers, B.A.: Hard-to-answer questions about code. In: Evaluation and Usability of Programing Languages and Tools (PLATEAU'10), pp. 1–6. Association for Computing Machinery, Reno (2010). https://doi.org/10.1145/1937117.1937125
14. Sharma, V.S., Mehra, R., Kaulgud, V.: What do developers want? An advisor approach for developer priorities. In: Proceedings of the 2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), pp. 78–81. Buenos Aires. (2017). https://doi.org/10.1109/CHASE.2017.14
15. Yin, R.K.: Case Study Research Design and Methods, 4th edn. Sage Publications, Thousand Oaks. (2009). https://trove.nla.gov.au/work/11329910
16. Strauss, A., Corbin, J.: Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, 2nd edn. Sage Publications, Inc, Thousand Oaks (1998)
17. Luo, Q., et al.: An empirical analysis of flaky tests. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014), pp. 643–653. Association for Computing Machinery, New York (2014). https://doi-org.e.bibl.liu.se/10.1145/2635868.2635920
18. jenkinsci/eiffel-broadcaster-plugin. original-date: 2019-01-22T15:04:13Z. (2019). https://github.com/jenkinsci/eiffel-broadcaster-plugin
19. McNab, A.L., Ladd, D.A.: Information quality: the importance of context and trade-offs. In: Proceedings of the 2014 47th Hawaii International Conference on System Sciences, pp. 3525–3532. ISSN: 1530-1605. Waikoloa (2014). https://doi.org/10.1109/HICSS.2014.439
20. Ven, K., Verelst, J., Mannaert, H.: Should you adopt open source software? IEEE Software 25(3), 54–59 (2008). https://doi.org/10.1109/MS.2008.73

How to cite this article: Ahmad A, Leifler O, Sandahl K. Data visualisation in continuous integration and delivery: Information needs, challenges, and recommendations. *IET Soft*. 2021;1–19. https://doi.org/10.1049/sfw2.12030