The Journal of Systems & Software 192 (2022) 111397

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

A qualitative analysis of themes in instant messaging communication of software developers☆

Camila Costa Silva *, Matthias Galster, Fabian Gilson

University of Canterbury, Christchurch, New Zealand

ARTICLE INFO

Article history:
Received 30 July 2021
Received in revised form 7 June 2022
Accepted 10 June 2022
Available online 16 June 2022

Dataset link: https://zenodo.org/record/6668356

Keywords:
Instant messaging
Developer communication
Reusable knowledge
Software engineering themes
Thematic analysis

ABSTRACT

Software developers use instant messaging (e.g., Slack, Gitter) to collaboratively discuss software engineering problems and solutions. This communication takes place in chat rooms that generally contain a description of the main topic of discussion and the messages exchanged. To analyze whether and how the knowledge accumulated in these chat rooms is relevant to other developers, we first need to understand the themes discussed in these chat rooms. In this paper, we used *thematic analysis* to *manually* identify software engineering themes in the description of 87 chat rooms of Gitter, an instant messaging tool for software developers. Then, we checked whether these themes also occur in 184 public chat rooms of Slack, another instant messaging tool. We identified 47 themes in Gitter chat rooms, and regarding the applicability of themes, we could relate 36 of our themes to 173 Slack chat rooms. Our results indicate that, in the context of our study, chat rooms in developer instant messaging communication are mostly about software development technologies and practices rather than development processes. Furthermore, most chat rooms are topic- rather than project-related (e.g., a chat room used by developers of a particular software development project).

1. Introduction

Software development requires different types of knowledge and expertise (Baltes and Diehl, 2018), for example, about development processes, practices and techniques, design and programming, a particular code base or specific application domains (Robillard, 1999). This knowledge constantly evolves as new technologies, programming languages and practices emerge (Kruchten, 2008). One way for developers to get help when performing development activities is to participate in online discussions with others via *instant messaging* (e.g., Slack[1], Gitter[2], Microsoft Teams[3]) (Storey et al., 2017; Aniche et al., 2018). Instant messaging allows developers to share knowledge in a highly collaborative manner (Storey et al., 2017; Zagalsky et al., 2016; Dittrich and Giuffrida, 2011). In chat rooms (or "channels" in Slack), which are often grouped in "workspaces" (in Slack) or "communities" (in Gitter), developers exchange messages to share experiences (e.g., related to the use of a technology), insights on how to solve practical problems (e.g., how to implement a particular feature) and even concrete software artifacts (e.g., software documentation or code snippets) (Storey et al., 2017). Over the years, the popularity of instant messaging in software engineering has been increasing (Statista, 2018; Chatterjee et al., 2019).

Communication via instant messaging accumulate large amounts of information in form of messages sent between developers (Storey et al., 2017). These messages may contain knowledge potentially useful for other developers, such as code snippets and software documentation (Chatterjee et al., 2019, 2021). On the other hand, this large amount of messages and information makes it difficult to find what one is looking for. Developer communication is unstructured, and sometimes developers may not be experts in the problem domain (Antonino et al., 2016) to perform a meaningful search (Soliman et al., 2021). Hence, simple searches in such communication (e.g., based on keywords) often do not satisfy the information needs of developers since search results are not relevant or not reusable in a particular context (Treude et al., 2015; Soliman et al., 2017, 2021). Furthermore, software engineering concepts are often abstract and the same concept can be described by many keywords, and the same keyword might refer to different concepts (Soliman et al., 2017). For example, searching for "Apache webserver performance" may not return content which does not use the keywords "Apache", "webserver" and "performance", but which still includes relevant

☆ Editor: Nicole Novielli.

* Corresponding author.
E-mail addresses: camila.costasilva@pg.canterbury.ac.nz (C. C. Silva), matthias.galster@canterbury.ac.nz (M. Galster), fabian.gilson@canterbury.ac.nz (F. Gilson).

1 https://slack.com/.

2 https://gitter.im/.

3 https://www.microsoft.com/en-ww/microsoft-teams/group-chat-software.

https://doi.org/10.1016/j.jss.2022.111397

discussions, e.g., about throughput, a measure for performance. Words could also occur with different meanings (e.g., "server" could be related to a software architecture components, to deployment infrastructures or even hardware components).

Previous research (see Section 2) has explored benefits and users of instant messaging in software development (Giuffrida and Dittrich, 2013). For example, previous research found that developers use instant messaging because it is "instant" and real-time (i.e. there are no delays in message delivery), supports peer-to-peer messaging (rather than reaching out to a broad and potentially anonymous audience), and creates presence awareness (Niinimäki and Lassenius, 2008). However, there are few studies to investigate to what extent instant messaging is a suitable source for reusable software development knowledge. In the context of this paper, *reusable* refers to knowledge that can be applied in different contexts or to solve similar problems (Markus, 2001), e.g., personal experiences of developers or software artifacts, such as source code or documentation. Additionally, in contrast to Stack Overflow, a frequently studied Q&A web forum where developers discuss problems, there is a limited number of works on extracting information from developer instant messaging communication (Chatterjee et al., 2019) and on the kind of knowledge that is shared through instant messaging.

To understand the role of developer instant messaging communication as a source of reusable knowledge (e.g., by building "domain-specific" search engines on top of instant messaging), we first need to understand the nature of knowledge in this communication and the main themes discussed. Therefore, our research question is: **what themes represent the main discussions of developers in chat rooms of instant messaging?** Since we are interested in reusable and accessible knowledge, our study focuses on *public* chat rooms of instant messaging, rather than instant messaging within organizations (e.g., Slack channels within companies)[4].

We *manually* identified software engineering themes based on the description of 87 public developer chat rooms of Gitter through thematic analysis (Braun and Clarke, 2006). We organized the themes in a *map of themes*. A *manual* analysis of themes (rather than using automated information retrieval techniques such as topic modeling) allowed us to contextualize themes and to get richer insights about themes. Therefore, our map of themes represents a contextualized description of the chat rooms selected for this study. We identified 47 themes grouped into four first-level themes: (1) Software development (example theme: web development), (2) Software architecture (example theme: microservices), (3) Software quality (example theme: testing), and (4) Professional development (example theme: coding). We also compared themes to:

- A standardized knowledge framework for software engineering, the Software Engineering Body of Knowledge (SWEBOK) (Bourque and Fairley, 2014). We found that most themes are related to concrete implementation problems, rather than higher-level process issues.
- Information needs of developers based on web search queries typically executed by developers (Xia et al., 2017) and on topics discussed in Stack Overflow (Barua et al., 2014). We found that our themes can be associated with most of the high-level categories of search queries proposed by Xia et al. (2017), such as queries related to programming, tools, testing, third party code reuse, and software development practices. In relation to Stack Overflow (Barua et al., 2014), we checked that our themes are in alignment with the topics discussed in Stack Overflow, for example, regarding web-related themes, platform-specific concerns, security and quality assurance.

To verify the applicability of themes, we checked whether our themes from Gitter would also appear in a sample of 184 public chat rooms of Slack. By doing that, we were able to check our themes in other contexts and if they can also represent the discussions in other instant messaging tools used by software developers. We found 36 of the 47 themes from Gitter in 173 of the 184 Slack chat rooms.

In summary, the main contribution of our study is a "portrait" of the main themes discussed in developer instant messaging (i.e. a map of themes) and insights about the characteristics of these themes in the context of the chat rooms. Researchers and practitioners can use our analysis to understand latest trends and challenges emerging in developer communities. Furthermore, the map of themes can be used to develop data-driven software tools to identify knowledge in instant messages or other community-like environments for developer communication. For example, our themes can be used as labels to train automated techniques such as classifiers (e.g., Chatterjee et al. (2021) created an approach based on classifiers that automatically identifies useful information to software developers in public chat rooms). Our map of themes can also be applied in approaches for augmenting software documentation or "semantically tagging" chat rooms (e.g., Souza et al. (2019) created a semi-automatic approach that organizes the knowledge available on Stack Overflow to build cookbooks for APIs). Finally, our study can be used as a guide to apply thematic analysis and expand our map of themes by incorporating new themes identified in chat rooms from other instant messaging tools such as Microsoft Teams[5], Discord[6] or Stack Overflow chats[7]. We discuss detailed implications of our findings for practitioners in Section 5.2 and researchers in Section 5.3.

The rest of the paper is organized as follows. In Section 2, we introduce the background and studies related to our research. We describe the research method in Section 3 and present the results in Section 4. In Section 5, we further discuss the themes identified, and present implications and threats to validity. Finally, in Section 6 we present concluding remarks and future work.

2. Background and related work

Knowledge in software engineering has been explored from different angles. According to Rus et al. (2002) it is difficult to keep track of the knowledge necessary in software development, where that knowledge can be found, and its relevance for certain development tasks. Furthermore, communicating knowledge between software developers is challenging (Treude and Storey, 2011) because most knowledge is poorly and informally captured, difficult to find and often not reusable (Komi-Sirviö et al., 2002).

Previous works have studied what knowledge is shared by developers. For example, Barua et al. (2014) discovered topics and their trends in Stack Overflow (e.g., discussions on tools for version control, technologies like .NET, jobs and experience). Other works have investigated the information needs of developers. For example, Xia et al. (2017) identified what developers search on the web during software development, including the frequency and difficulty of the different search tasks; and Pascarella et al. (2018) identified the information needs of developers in the context of code reviews, such as the need for knowing the uses of methods and variables declared and modified.

[4] Note that in this work we do not assess the quality of knowledge shared in instant messaging.

[5] https://www.microsoft.com/en-nz/microsoft-teams/group-chat-software.

[6] https://discord.com/.

[7] https://chat.stackoverflow.com/.

Moreover, some works explored how to find (and reuse) information from different platforms that developers use as part of their daily work. For example, Soliman et al. (2016) focused on one particular type of knowledge related to software technologies and identified posts from Stack Overflow that provide useful technology-related architecture knowledge. Furthermore, Soliman et al. (2018) developed a domain-specific search engine to search for architecture knowledge in Stack Overflow posts. Treude and Robillard (2016) extracted information from Stack Overflow posts to augment API documentation. Their approach identifies "insight sentences" (sentences that are related to a particular API type and that provide insight not contained in the API documentation of that type) and augments API documentation accordingly.

Regarding developer communication, previous studies focused on understanding how developers use and benefit from means of collaborative communication, or on exploring the information shared in this communication, such as blog posts and public chat rooms. Storey et al. (2014, 2017), for example, reported how different forms of communication and social interactions (e.g., face-to-face, voice calls, instant messaging, web searches and source code repositories) play a critical role in gaining and sharing knowledge in software engineering, and how the choice of communication channels influences developer activities. Chatterjee et al. (2017) explored which kinds of information regarding code snippets are embedded in different software-related documents, such as developer discussions in public chat rooms. Chatterjee et al. (2017) found that in public chat rooms there are more explanatory information about code snippets (i.e. why and how a functionality was implemented and it excepted output) than, for example, about data structure or code efficiency.

Regarding instant messaging, previous studies explored benefits of this mean of communication and found that instant messaging is a cost-effective technology that allow users to keep in touch, ask for spontaneous advice when needed, and helps build relationships (Giuffrida and Dittrich, 2013; Niinimäki and Lassenius, 2008). For example, Alkadhi et al. (2017) investigated the effects of their tool (REACT — RationalE Annotations in ChaT messages) on Slack conversations. REACT aimed to help developers understand each other in instant messaging by including emojis on messages to represent their rationale. Zhu et al. (2021) analyzed the discussions in Stack Overflow rather than the answers given to the questions posted. The authors found that such discussions contain a rich trove of data that is integral to the Q&A processes on Stack Overflow (e.g., questions with a small number of comments are likely to be answered more quickly than questions with no discussion).

Other previous studies focused on global software development and distributed development teams, reported how instant messaging facilitates communication (Giuffrida and Dittrich, 2013; Niinimäki and Lassenius, 2008). Lin et al. (2016), for example, explored the impact of Slack on development team dynamics. Their study found that developers use Slack for personal, team-wide and community-wide purposes, and that developers use and create "bots" to support their work.

There are also previous studies regarding the quality of the content shared in instant messaging. Chatterjee et al. (2019), for example, looked for useful information in public Slack conversations in comparison to Stack Overflow. The authors found that the largest proportion of Slack conversations are "design"-related conversations (which involve discussions of API usage and recommendations), followed by "explanatory"-type of conversations (e.g., developers explaining to each other technologies and capabilities of different programming languages). The authors also identified "structure"-related conversations (e.g., conversations about a specific data or control structure). In another study, Chatterjee et al. (2021) proposed an approach to automatically detect useful information (for mining or reading after the conversation has ended) from developer discussions in public chat rooms.

More recently, Gitter (an instant messaging tool originally used by users of Git repositories) caught the interest of software engineering research. For example, Sahar et al. (2021) studied how developers discuss issue reports and found that issue reports discussed in Gitter take more time to get resolved compared to issue reports that are not discussed. Ehsan et al. (2021) examined the response behavior of developers on Gitter to propose an approach to disentangle the messages in a single conversation thread into topic-specific conversation threads. Shi et al. (2021) explored disentangled Gitter conversations to better understand the collaboration and the data shared by developers in instant messaging communication. Finally, using thematic analysis (like in our study), Mezouar et al. (2021) identified the reasons behind the use of Slack and Gitter, the perceived impact on the associated projects and the quality characteristics of these instant messaging tools. The authors found that developers seek knowledge from instant messaging to obtain timely feedback from experts who, in return, share their expertise with others. This two-way interaction helps build developer communities and increases the reputations of those who contribute in these communities.

However, none of these works identified themes to understand in what context certain themes in instant messaging communication appear (in particular if we only consider "metadata" of instant messaging communication that developers typically check to find relevant chat rooms, e.g, titles and descriptions of chat rooms, rather than full conversations). Additionally, previous works have not explored whether and how themes of chat rooms are related to a standardized knowledge framework or information needs of developers.

3. Research method

We applied *thematic analysis* to obtain software engineering themes from instant messaging communication of developers. Thematic analysis is a "manual" method to identify, analyze and report patterns (themes) in qualitative data through coding (Boyatzis, 1998; Braun and Clarke, 2006). Thematic analysis assumes that "knowledge of reality is gained [...] through social constructions such as language [...], shared meanings, documents, tools [...]", rather than aiming at "evidence of [...] quantifiable measures of variables, hypothesis testing, and [...] inferences about a phenomenon from a representative sample" (Klein and Myers, 1999). Thematic analysis applies analyzes similar to Grounded Theory (e.g., coding and memoing) (Braun et al., 2019). However, thematic analysis (unlike Grounded Theory) is a method, rather than a methodology that does not only involve describing social processes or factors that influence a particular phenomenon, and does not rely on one particular theoretical framework that carries assumptions about the data and what they represent in "reality" (e.g., inductive, deductive, constructionist frameworks) (Braun et al., 2019; Braun and Clarke, 2006).

We applied reflexive thematic analysis (Braun and Clarke, 2019) and adopted a semantic approach. This means that the analysis process progressed from the description of data (i.e. themes as they were identified as implicit and explicit patterns in data) to the interpretation of themes (i.e. the analysis of their significance) (Braun and Clarke, 2006, 2019). In the following we outline the details of the study design. When conducting and reporting the study, we considered the evaluation guidelines for thematic analysis studies as proposed by Braun and Clarke (2021a).

3.1. Data sampling

To select the cases (i.e. chat rooms) for our study, we adopted a purposive sampling approach as mentioned by Baltes and Ralph (2022), i.e. the sample is selected based on the usefulness for achieving the objective of the study. We provide details in the following sections.

3.1.1. Context

The context of our study is an instant messaging tool used by software developers. To select the tool for our study, we used a comparison framework (Costa Silva et al., 2019) to compare Slack (most known tool for team communication) (Statista, 2018), Microsoft Teams (due to its popularity) (Lardinois, 2019), Gitter (identified by other researchers (Storey et al., 2014) and subject of recent studies (Romero et al., 2020; Sahar et al., 2021)) and Spectrum (alternative to Slack). Costa Silva et al. (2019) provide details of the comparison of the four tools. We selected Gitter for the following reasons:

- Discussions on Gitter focus on software development issues. Therefore, we did not need to filter discussions that are not about software development (as they might appear in Slack or Microsoft Teams);
- Gitter (unlike Spectrum, Slack and Microsoft Teams) offers comprehensive APIs to access communication data without paying a fee. Furthermore, Gitter's data use policy and privacy regulations do not restrict the use of data by other users (GitLab, 2020);
- Unlike Slack and Microsoft Teams, Gitter does not restrict access to chat rooms (e.g., no authorization required from chat room administrators).

Gitter is a free open source instant messaging tool released in 2014 with the original goal to support community collaboration amongst developers that use Git repositories for their projects. At the time of conducting this study in 2019 (see Section 3.1.3 for details about when we extracted communication data from Gitter), Gitter had more than 300,000 chat rooms in more than 90,000 communities and more than 800,000 users[2]. This number may have increased since then. In Gitter, a *community* is a group of users. Conversations between users happen in *chat rooms* and a community can have more than one chat room. For example, gitterHQ[8] is a community with several chat rooms such as "developers", "javascript" and "services". Users can join any chat room of a community. All communities and chat rooms in Gitter are public by default (GitLab, 2020).

3.1.2. Units of analysis

Our units of analysis are chat rooms within Gitter communities. Gitter does not provide a complete list of chat rooms, it only allows users to browse communities that are grouped based on tags assigned to those communities. Searching Gitter based on keywords on the other hand provides a *list of chat rooms* as the search results. Since we conducted a manual analysis of chat rooms, we could not use all publicly available chat rooms like other studies that analyzed Gitter data, such as (Ehsan et al., 2021). Others, such as (Shi et al., 2021) selected top-participated communities, and Parra et al. (2020) that selected chat rooms from ten communities. Therefore, to create our own list of chat rooms and to identify chat rooms for our study, we used Gitter's search which searches the tags, names and description of chat rooms. We used 25 generic search terms related to software development activities (e.g., requirements, architecture, maintenance) and generic terms like "software engineering", "development", "management", "quality" and "technology" found in textbooks (Bass et al., 2003). The complete list of search terms and the number of chat rooms for each term is available online[9].

Table 1
Descriptive statistics of Gitter chat rooms (age of 1 includes chat room less than one month old).

	Min	Max	Stdev	Mean	Median
Users	5	16,642	2757	1453	291
Messages	10	365,706	55,409	27,726	4295
Age (months)	1	68	16	36	39
⌀ messages/month	1	9624	1333	658	137
⌀ messages/user	1	597	93	40	12

The search returned 116 unique chat rooms. After removing non-English chat rooms, chat rooms without messages or with messages from bots only, and chat rooms with one-to-one conversations between two users (to ensure that we elicit themes relevant to a broader audience), we selected chat rooms with more than three users, but ended up with chat rooms with five or more users, since at the time of conducting the study, there were no chat rooms that met selection criteria with three or four users. This resulted in 87 chat rooms. Table 1 shows descriptive statistics of these chat rooms. The complete list of chat rooms is available online[9]. Seventy-five chat rooms appeared in separate communities, while twelve chat rooms are from five communities (e.g., chat rooms *spring-security* and *spring-security-oauth* are both from the *spring-projects community*) — see Table 4 for a full list of communities that contribute more than one chat room to the sample. In Gitter, names of chat rooms include the community they belong to. For example, chat room *FreeCodeCamp/Contributors* belongs to community *FreeCodeCamp*.

3.1.3. Data collection

For each Gitter chat room, we extracted the following data (using Gitter's API[10]):

- URL, name, tags (optional);
- Description (optional, up to 80 words);
- Number of users;
- List of messages;
- Age (months from first to last message at time of study).

We collected this data between June 24 and August 7, 2019. All data were stored in spreadsheets and later imported into NVivo 12[11] for data analysis.

3.2. Data analysis

Fig. 1 illustrates the data analysis steps of our study. Two researchers conducted the data analysis and a third researcher reviewed the results. All researchers discussed the results. We followed the thematic analysis steps suggested by Braun and Clarke (2006) (see Fig. 1). We iteratively and incrementally defined, reviewed and refined themes because thematic analysis involves constantly moving back and forth between the data, the coded extracts of data, and the analysis being produced (Braun and Clarke, 2006). Below we discuss each step in detail.

Step 1 — get familiar with data: To get familiar with the data, we created memos to capture the context of chat rooms and conversations. Memoing is the process of writing notes about

8 https://gitter.im/gitterHQ/home.

9 https://zenodo.org/record/6668356

10 https://developer.gitter.im/docs/welcome.

11 https://www.qsrinternational.com/nvivo-qualitative-data-analysis-software/home.

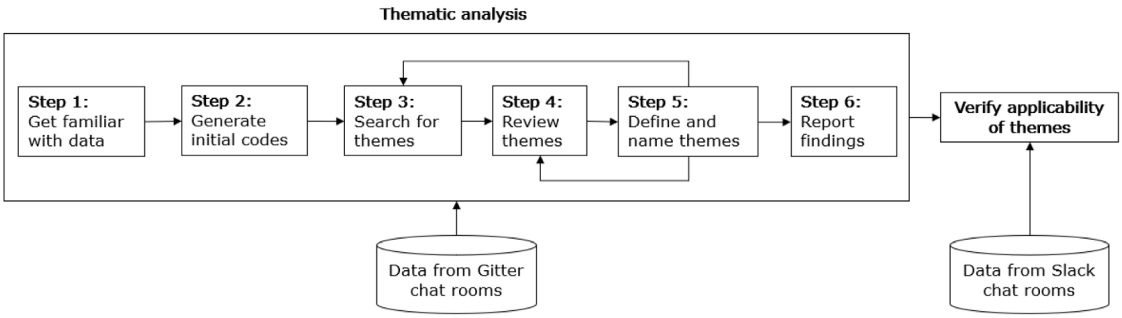

Fig. 1. Overview of data analysis.

the data to be coded. Memoing also helps elaborate categories (themes in our study), preliminary properties and relationships between these categories (Stol et al., 2016). To create the memo for each chat room, one of the researchers screened the messages in the chat rooms and read parts of the conversation. This process of reading the chat rooms' conversations was "ad-hoc" in the sense that we did not have systematic criteria in place that guided the reading (e.g., "read the first 10 messages of every conversation", or "read every second message in a conversation", etc.). We read as many messages as necessary to understand the context of discussions. If available we also consulted external content (e.g., GitHub or GitLab repositories and related websites) to obtain more insights about each chat room.

To add more context-related content to the memos for each chat room, we also automatically created a natural language summary of all messages exchanged in a chat room (El-Kassas et al. (2021) provide an overview of text summarization). To create the summary, we selected BART, an abstractive text summarization technique[12]. We used the implementation of BART from the Transformers Python library[13] (Model *BartForConditionalGeneration.from_pretrained('facebook/bart − large − cnn')*, setting the model to get summaries with 500 to 1000 words. We selected BART because it is a Sequence-to-Sequence (Seq2Seq) technique that usually performs well with short text summarization (El-Kassas et al., 2021). In addition to our own judgement of how fluent summaries sounded, we used ROUGE scores[14] to measure BART performance and adequacy of summaries (i.e. is the length appropriate and does the summary cover the most important information of the text it summarizes). We calculated the recall, precision and F1-score of ROUGE-1 (unigrams), ROUGE-2 (bigrams) and ROUGE-L (longest common sub-sequence) for the summaries of all chat rooms selected (see an overview of these scores in Table C.14 in Appendix C). Scores for each summary were within limits reported in literature (Khatri et al., 2018; Paulus et al., 2018).

Therefore, each memo had the summary of the messages shared in a chat room, notes about our understand of the conversations, URL links to external content (e.g., GitHub repository, websites) and definitions of terms mentioned in the chat rooms' descriptions. An example memo is below:

> Chat room iluwatar/java-design-patterns: *Members discuss the use of design patterns in Java, and help each other with questions and answers. Chat room offers examples of patterns through a GitHub account:* https://github.com/iluwatar/java-design-patterns
>
> Abstractive summary: *Im saying apply Lombok for Builder design pattern only not to whole project Lombok Builder real use case scenario for builder pattern this is for making people aware it is not required to write code manually Hi there is someone able to help me in a problem [...] Hey all question Im looking at contributing to the Iterator pattern Id like to implement an Iterator for a few different data structures instead of just a List as the current implementation does Is the process different if Im contributing to an existing pattern instead of creating a new one Ive read the Wiki and its got instructions for the latter but not the former Should I just fork the repo make my changes squash the commit and submit a pull request as otherwise described in the Wiki Thanks [...]*

Step 2 — generate initial codes: During this step, we inductively identified codes in the name and description of chat rooms and the memos created in Step 1. In our study, codes refer to words or terms found in these data that could (a) represent the potential subjects of discussion (e.g., "JavaScript", "Jobs", "Flutter") and (b) high level classes to characterize each chat room (*type of chat room*, e.g., project- or topic-related) and its object of discussion (*type of discussion*, e.g., tool or process). To identify the types of chat room, we analyzed the terms used in a conversation. For example, when users used terms like "pull request", "merge", "branch" in a conversation where users organized tasks between them. This was an indicator that this chat room is project-related (rather than, for example, product-related). Regarding types of discussion, we searched for definitions related to keywords that were part of the chat room's name or description (e.g., description of Laravel from the chat room *laravel/laravel*[34]) to help us identify if the object of discussion was, for example, a tool or a process. For details about types of chat room and types of discussion, see Section 4.1.

Step 3 — search for themes: In the search for themes, we analyzed all codes and grouped them into themes (the fourth- and third-level themes in our final map of themes), and higher-level themes (the second- and first-level themes in our final map of themes; see Figs. A.4–A.7 in Appendix A). For example, the second-level theme "Types of development" under the first-level theme "Software development" groups all themes related to different types of software development (e.g., web and mobile applications). For each chat room, we derived one, two or (in few cases) three themes.

Step 4 — review themes: In this step, we designed a map of themes to describe the relationship between themes. We show the resulting map of themes (final, after several iterations) in Figs. A.4–A.7 in Appendix A. When creating this map, we reviewed the relationship between themes, i.e. whether higher level themes were grouping lower level themes appropriately. We

[12] Abstractive summarization techniques "abstract" the text to create summaries with words and sentences that differ from the original text sentences, rather than identify "important" sentences as done with extractive summarization techniques. This typically increases the readability and cohesiveness of summaries (El-Kassas et al., 2021).

[13] https://huggingface.co/docs/transformers/model_doc/bart.

[14] ROUGE is a set of metrics that measure the number of matching n-grams (based on words' syntactic) between the summary generated and a reference, which is usually the original text (Lin, 2004).

also looked for inconsistencies (such as duplicated themes) and required changes in the name of themes.

Step 5 — define and name themes: In this step, we described each theme that was assigned to the chat rooms. These descriptions are presented in more detail later in Section 4.2. The authors rechecked: (a) whether themes and their descriptions were appropriately representing the chat rooms. For example, a chat room assigned "Coding" may in fact discuss experiences exchange on coding, which would then better refer to theme "Coding quality"; (b) whether the *types of chat room* and *types of discussion* were describing the context of chat rooms. For example, we checked whether a chat room is about a general topic or the development of a software. Finally, we stopped the process of revisiting steps 3 and 4 to review our themes when, based on our experience and understanding, we could not identify new themes and the themes identified were describing the context of the chat rooms (see more details in Section 5.4).

Step 6 — report findings: When reporting our findings, we also contextualize themes, as suggested by Clarke and Braun (2019), by performing the following:

- *Mapping of themes to existing knowledge framework:* We mapped themes to topics in the SWEBOK (Bourque and Fairley, 2014) as a standardized knowledge framework in software engineering to identify similarities and differences.
- *Comparison of themes with information needs of developers:* We mapped themes to information needs of developers. Here, information needs are based on questions developers typically try to answer by searching the internet (Xia et al., 2017) and on the topics discussed by developers in Stack Overflow (Barua et al., 2014; Beyer et al., 2019).

3.3. Applicability of themes

We checked whether the themes also appear in public chat rooms of Slack (2022), another instant messaging tool. By applying our themes to Slack, rather than to other Gitter chat rooms, we are able to check our themes in other contexts and if they can also represent discussions of other instant messaging communication channels used by software developers.

Slack, similar to Gitter, was released in 2014. In 2019, this instant messaging tool had more than 10 million users, including organizations and individuals in different domains, such as engineering, marketing and project management (Statista, 2018; Slack, 2022). In Slack, a *workspace* (community in Gitter) is a group of users. Conversations happen in *channels* (chat rooms in Gitter). A workspace can have more than one channel. Here, we refer to a Slack workspace as a community and a Slack channel as chat room.

We searched public Slack communities on a website, called Slofile, which hosts a public database of public Slack communities (Slofile, 2020). We filtered communities based on language (English) and category based on tags (programming). From these, we selected communities that had at least one chat room with more than two users (similar to what we did for Gitter). Not all users in the community were users in all chat rooms of the community. Note that the Slofile website was our main source of information about Slack communities and their chat rooms and this website only shows up to ten of the most popular chat rooms of each community as described in Slofile (2021). Slofile does not provide an API to create customized queries for communities).

This resulted in 184 Slack communities. In each of these communities, we chose the chat room with the largest number of users that was (a) not a list of community announcements and news, and (b) not used to introduce new users to the community. Furthermore, we did not chose chat rooms when their description (or the description of the community they belong to) was not comprehensive enough to understand what the chat room was about (e.g., "#investing"). Table 2 shows descriptive statistics for the number of users of these Slack chat rooms. The complete list of communities selected is available online[9].

Table 2
Descriptive statistics related to the number of users of Slack chat rooms.

	Min	Max	Stdev	Mean	Median
Community	5	25,730	3599	1544	264
Chat room	3	25,209	3242	1236	122

Due to access restrictions of Slack, we did not collect data directly from the Slack chat rooms but from the Slofile website. Therefore, we could not access the actual messages or the age of chat rooms. We manually collected the following data for each chat room and the community a chat room belongs to (February 28 and March 1, 2020):

- Community URL and name;
- Community description (optional, up to 80 words);
- Community number of users;
- Chat room name;
- Chat room description (optional, up to 80 words);
- Chat room number of users.

We checked the similarity of Gitter and Slack chat rooms. However, the only common metric that we could obtain for Gitter and Slack was the number of users, so we used a student t-test (unequal variance; significance level 0.05) to check whether the number of users in Gitter chat rooms is significantly different to the number of users in the Slack chat rooms. A p-value of 0.28 indicates that the selected Gitter and Slack chat rooms were not significantly different at least in terms of the number of users.

We followed a semantic approach to associate the themes from Gitter chat rooms to Slack chat rooms. Based on a Slack chat room's description (and sometimes the community's description), we labeled each Slack chat room with one or two themes. For context, we also identified the types of chat rooms (project-related or topic-related) and types of discussion (e.g., products, frameworks, libraries — based on the communities' descriptions) of these Slack chat rooms.

4. Results

4.1. Characterizing chat rooms

As described in Section 3.2, during Step 2 of the thematic analysis we identified the *type of chat room* and the *type of discussion* in each chat room based on its name, description and related memo. Types of chat rooms describe the purpose of the chat room (e.g., to discuss the development or maintenance of a particular project, or to share issues, advice and personal experiences regarding a topic):

- **Project-related:** Chat room is about development-related issues and the planning, distribution and tracking of tasks in a specific software project (e.g., Simple Invoices, an invoice management system for IT professionals[15]). In this type of chat room the users were developers working on that project discussing what tasks to work on.
- **Topic-related:** Chat room is about general topics (e.g., front-end programming using Java libraries[16]) or issues and experiences with using a specific software (e.g., Snipe IT, an IT licensing management system[17]). In this type of chat

15 https://gitter.im/simpleinvoices/simpleinvoices.
16 https://gitter.im/vaadin/web-components.
17 https://gitter.im/snipe/snipe-it.

Table 3
Gitter chat rooms by type of room and type of discussion.

Type of discussion	Type of chat room		*Total*
	Project	Topic	
Frameworks	3	18	21 (24%)
Libraries	6	21	27 (31%)
Principles & practices	3	7	10 (11%)
Processes	0	1	1 (1%)
Products	4	3	7 (8%)
Tools	7	11	18 (21%)
Other	0	3	5 (3%)
Total	23 (26%)	64 (74%)	**87 (100%)**

room, discussions were conducted by developers who were dealing with the topic discussed (e.g., users of the product discussed).

Types of discussions describe what is the object of discussion in that chat room:

- **Frameworks**: Users discuss issues and experiences with frameworks (e.g., Flutter[18], which is an open source framework for the development of multi-platform applications);
- **Libraries**: Users discuss issues and experiences with software libraries (e.g., Scikit-learn[19], which is a library with machine learning models);
- **Principles & practices**: Users discuss fundamental concepts of software engineering and programming paradigms and/or practices for software development (e.g., best practices for smart contracts[20]);
- **Processes**: Users discuss processes for developing and maintaining software (e.g., processes for building solutions for data science problems[21]);
- **Products**: Users discuss issues and experiences with a particular software product or suite (e.g., Snipe IT[17], which is an open source Information Technology (IT) asset management system);
- **Tools**: Users discuss issues, use experiences or the development of tools to create, debug, test or maintain software (e.g., Semantic-release[22], which is an automated tool for version management and package publishing);
- **Other**: Users discuss anything not related to the previous types (e.g., job opportunities for developers[23]).

Table 3 shows the number of Gitter chat rooms by type of chat room and type of discussion. We found that most chat rooms are related to topics (64 out of 87 chat rooms) rather than projects. Regarding the type of discussion, most chat rooms are about libraries (e.g., *scikit-learn/scikit-learn*[19]), frameworks (e.g., *ConsenSys/truffle*[24]) and tools (e.g., *ngrx/ store*[25]).

4.2. Overview of themes

The resulting map of themes, after several iterations, includes types of nodes:

- **Third- and fourth-level**: Themes identified in chat rooms that derived from codes. Third- and fourth-level themes are not shown in the overview map (Fig. 2), but only in the full version of the map (see Figs. A.4–A.7 in Appendix A).
- **Second-level**: Themes which aggregate third- and fourth-level themes.
- **First-level**: Themes that describe the four main branches in the map of themes and aggregate second-level themes. These themes are:
  - *SD — Software development*: Themes related to software development and maintenance activities;
  - *SA — Software architecture*: Themes related to the structure of systems and technologies;
  - *SQ — Software quality*: Themes related to quality of functionality and correctness;
  - *PD — Professional development*: Themes related to software engineering learning and training.

Fig. 2 shows only the first- and second-level themes of our complete map of themes. The complete version of this map, including third- and fourth-level themes is in Fig. A.4, Fig. A.5, Fig. A.6 and Fig. A.7 in Appendix A.

In Appendix B we included Tables B.10–B.13 where we describe second-, third- and fourth-level themes in detail, organized by second-level themes. For each theme we also included a chat room description as an example. SQ — Software Quality (a first-level theme) do not have third- and fourth-level themes; therefore, the descriptions and examples in Table B.10 refer to the second-level themes. The complete list of chat rooms with assigned themes can be found online[9].

In thirty-five (out of 87) chat rooms we identified two or three themes. The assignment of multiple themes added details to the characterization of the main discussion of these chat rooms. For example, chat rooms *hopelessoptimism/data-engineering-101*[26] and *webuildsg/live*[27] are both about *PD1.2 Workshops* but the former refer to workshops on *SD5.1.1 Big data* and the latter, workshops on *SD6.1 Graphical user interface*.

4.2.1. Themes in communities

Table 4 shows the themes of chat rooms that came from the same community[28]. As shown in Table 4, in some chat rooms from the same community we identified the same theme (e.g., chat rooms of the *spring-projects* community), but not all chat rooms of a community discuss the same theme. For example, chat rooms of the *ConsenSys* community were labeled with *SA1.2 Blockchain* in addition to other themes. Therefore, we cannot assume that all chat rooms of a community always address the same theme.

4.2.2. Similar themes in different contexts

We identified similar themes in different contexts. We show these themes in Fig. 3. The themes *PD2.1 Coding* and *SQ1 Code quality* are both referring to programming activities but in different contexts. *PD2.1 Coding* appeared in the context of users helping each other to deal with coding issues, while *SQ1 Code quality* appeared in chat rooms where users discuss resources to create quality code. Similarly, *SQ2 Software testing* appears in the context of discussions about methods and resources for testing and *PD2.2 Testing* in the context of discussing testing practices.

18 https://gitter.im/flutter/flutter.
19 https://gitter.im/scikit-learn/scikit-learn.
20 https://gitter.im/ConsenSys/smart-contract-best-practices.
21 https://gitter.im/FreeCodeCamp/DataScience.
22 https://gitter.im/semantic-release/semantic-release.
23 https://gitter.im/lnug/london-node-jobs.
24 https://gitter.im/ConsenSys/truffle.
25 https://gitter.im/ngrx/store.
26 https://gitter.im/hopelessoptimism/data-engineering-101.
27 https://gitter.im/webuildsg/live.
28 Communities may have more chat rooms than the ones selected for our study.

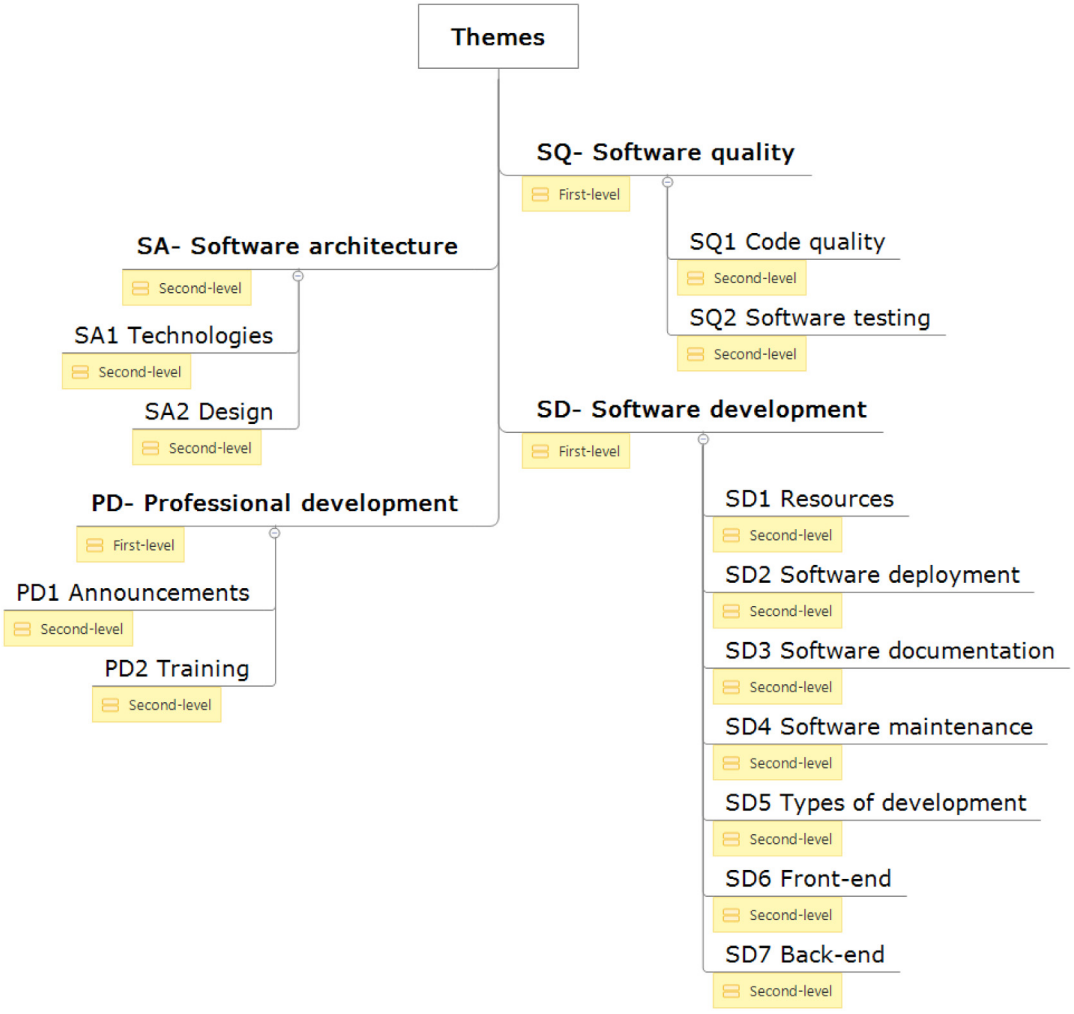

Fig. 2. Software engineering themes in Gitter chat rooms.

Table 4
Themes in Gitter chat rooms from the same community.

Community	Chat room	Themes
FreeCodeCamp	Contributors	PD2.1 Coding
	DataScience	SD1.2.1 Libraries SD1.2.3 Machine learning [Systems]
	HelpBackEnd	SD7 Back-end PD2.1 Coding
	testable-projects-fcc	PD2.1 Coding
ConsenSys	smart-contract-best-practices	SA2.1 Patterns SA1.2 Blockchain
	truffle	SD1.3 Integrated development environment SA1.2 Blockchain
ManageIQ	manageiq	SA1.1 Cloud computing
	manageiq/ui	SA1.1 Cloud computing SD6.1 Graphical user interface
Spring-projects	spring-security spring-security-oauth	SA2.2.1 Security SA2.2.1 Security
Webpack	docs	SD2.1.1 Build automation SD3 Software documentation
	webpack	SD2.1.1 Build automation SQ1 Code quality

4.3. Themes and the SWEBOK

To check how the themes identified in our study align with a standardized knowledge framework for software engineering, we mapped them to the SWEBOK (Bourque and Fairley, 2014). The SWEBOK is organized into 15 chapters which have *sections*. For example, chapter "1 Software Requirements" consists of the

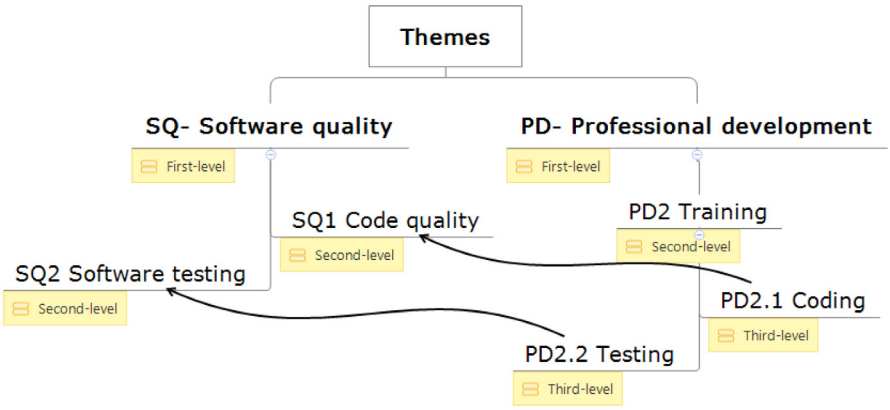

Fig. 3. Themes that appeared in different contexts.

sections "1.1 Software Requirements Fundamentals", "1.2 Requirements Process", etc. Each section has *subsections* (e.g., under "1.1 Software Requirements Fundamentals", there are subsections "1.1.1 Definition of a Software Requirement", "1.1.2 Product and Process Requirements", etc.). We were able to relate all our themes to SWEBOK (sub)sections based on their conceptual similarities and considering the context of the themes, i.e. the chat rooms in which they occur (see Table 5). The mapping was conducted, reviewed and discussed by three researchers. When mapping themes to (sub)sections, we followed a bottom-up approach, i.e. we first tried to map themes to subsections and if that was not possible (e.g., if themes were more general than subsections) tried to map them to a section. In Table 5, we also record how we mapped themes and SWEBOK (sub)sections following these mapping rules:

- *Explicit:* Theme is referenced in the SWEBOK (sub)section (e.g., *SD1.1 Application programming interface* is explicitly covered by subsection "3.4.1 API Design and Use");
- *Implicit:* Theme is not directly referenced in SWEBOK (sub)section, but it is related to the content of the SWEBOK (sub)section due to the context of chat rooms under that theme (e.g., *SD2 Software deployment* is implicitly covered by subsection "6.6.1 Software Building");
- *Example:* Theme can be considered as an example of what is described in a SWEBOK (sub)section (e.g., *SD2.1.1 Build automation* is a concrete example for the concepts discussed in subsection "3.3.8 Integration").

We could not map themes to (sub)sections in the following chapters:

- 9 Software Engineering Models and Methods
- 12 Software Engineering Economics
- 14 Mathematical Foundations
- 15 Engineering Foundations

4.4. Themes and developer knowledge needs

In this section, we discuss themes and how they relate to knowledge needs of developers. We identify knowledge needs based on what developers search on the web (see Section 4.4.1) as well as what developers discuss on Stack Overflow (see Section 4.4.2).

4.4.1. Web search queries

Xia et al. (2017) identified knowledge needs of developers based on what they search on the web. The authors described 34 search tasks, grouped into seven search categories:

1. General searches (e.g., to understand terminology, software development practices);
2. Debugging and bug fixing (e.g., to understand exceptions/ errors and security or performance bugs);
3. Programming (e.g., to learn about design patterns, code examples, standards);
4. Third party code reuse (e.g., to find reusable code snippets, libraries, HTML/ CSS templates);
5. Tools (e.g., to learn how to use and/or customize IDEs, version control systems, issue management systems);
6. Database (e.g., to learn about SQL, no-SQL database issues and optimization);
7. Testing (e.g., to find guidelines on testing methods, how to use a testing tool, data sets for testing).

We related the search categories to themes by comparing the description of each category (including its list of search tasks) to the scope of each first-level theme:

- SD — Software Development, SA — Software Architecture and PD — Professional Development can be related to search categories "Programming", "General searches" and "Tools".
- SQ — Software Quality can be related to search category "Testing".
- SD — Software Development can also be related to search category "Third party code reuse".

Search categories "Database", and "Debugging and bug fixing" were not clearly related to our themes. Even though we could not associate these search tasks to a particular theme, developers may discuss questions related to "Debugging and bug fixing" in individual messages. However, we did not find it as the theme (i.e. the main subject of discussion in a chat room).

Although the first-level themes Software quality and Professional development were related to some of the search categories, we could not associate the themes *SQ1 Code quality*, *PD1.1 Jobs* and *PD1.2 Workshops* to the search tasks within such categories. For example, there were no search tasks for job opportunities under "General searches" or any other search category.

In summary, since web searches of developers can be related to themes in developer communication, practitioners may also search developer communication to satisfy information needs captured in web searches, in particular information needs regarding new technologies, programming, tools and third party code reuse.

4.4.2. Stack overflow

Barua et al. (2014) analyzed what topics were discussed by developers in Stack Overflow. Furthermore, Beyer et al. (2019)

Table 5
Mapping of themes to SWEBOK sections and subsections.

First-level theme	Second-, third- and fourth-level theme	SWEBOK chapter	SWEBOK (sub)section	Mapping
SD	1.1 Application programming interface	3 Software Construction	3.4.1 API Design and Use	Explicit
SD	1.2.1 Configuration	6 Software Configuration Management	6.4.1 Software Configuration Status Information	Explicit
		8 Software Engineering Process	8.2 Software Life Cycles	Implicit
SD	1.2.2 Release	6 Software Configuration Management	6.6.2 Software Release Management	Explicit
SD	1.3 Integrated development environment	3 Software Construction	3.5.1 Development Environments	Explicit
SD	1.4 Reverse engineering	5 Software Maintenance	5.4.3 Reverse Engineering 5.5 Software Maintenance Tools	Explicit
SD	1.5 Version control	6 Software Configuration Management	6.7 Software Configuration Management Tools	Explicit
SD	2 Software deployment	6 Software Configuration Management	6.6.1 Software Building	Implicit
SD	2.1 Continuous integration	3 Software Construction 6 Software Configuration Management	3.2.1 Construction in Life Cycle Models 6.6.2 Software Release Management	Implicit
SD	SD2.1.1 Build automation	3 Software Construction	3.3.8 Integration	Example
SD	2.2 Distributed software	3 Software Construction	3.4.12 Construction Methods for Distributed Software	Explicit
SD	3 Software documentation	1 Software Requirements	1.5.1 System Definition Document	Explicit
		7 Software Engineering Management	7.4 Review and Evaluation	Implicit
SD	4 Software maintenance	5 Software Maintenance	5.1.5 Evolution of Software	Explicit
		6 Software Configuration Management	6.3.1 Requesting, Evaluating, and Approving Software Changes	Implicit
SD	5.1.1 Big data	13 Computing Foundations	13.12.6 Data Mining	Implicit
SD	[Machine learning] 5.1.2.1 Libraries	13 Computing Foundations	13.12.6 Data Mining	Example
SD	[Machine learning] 5.1.2.2 Research	13 Computing Foundations	13.12.6 Data Mining	Example
SD	[Machine learning] 5.1.2.3 Systems	13 Computing Foundations	13.12.6 Data Mining	Example
SD	5.2 Games	3 Software Construction	3.4 Construction Technologies	Implicit
SD	5.3 Mobile development	3 Software Construction	3.3.2 Construction Languages	Implicit
SD	5.4 Web development	3 Software Construction	3.3.2 Construction Languages	Implicit
SD	5.5 Multi-platform development	3 Software Construction	3.5 Software Construction Tools	Example
SD	6 Front-end	2 Software Design	2.4 User Interface Design	Implicit
SD	6.1 Graphical user interface	2 Software Design	2.4.2 User Interface Design Issues	Explicit
SD	7 Back-end	2 Software Design	2.3.1 Architectural Structures and Viewpoints	Implicit
PD	1.1 Jobs	11 Software Engineering Professional Practice	11.1 Professionalism	Implicit
PD	1.2 Workshops	11 Software Engineering Professional Practice	11.1 Professionalism	Implicit
PD	2.1 Coding	3 Software Construction 13 Computing Foundations	3.3.3 Coding 13.1.5 Problem Solving Using Programs	Explicit
PD	2.2 Testing	3 Software Construction 4 Software Testing	3.3.4 Construction Testing 4.5.1 Practical Considerations	Explicit
		10 Software Quality	3.3.2 Testing	Implicit
SA	1.1 Cloud computing	2 Software Design	2.3 Software Structure and Architecture	Example
SA	1.2 Blockchain	13 Computing Foundations	13.6.2 Types of Data Structure	Example
SA	2.1 Patterns	2 Software Design	2.3.3 Design Patterns	Explicit
SA	2.2.1 Security	2 Software Design	2.3.4 Architecture Design Decisions	Example
SA	2.3.1 Service-oriented architecture	2 Software Design	2.3.2 Architectural Styles	Example
SA	2.3.1.2 Containers	2 Software Design	2.3.2 Architectural Styles	Example
SA	2.3.1.1 Microservices	2 Software Design	2.3.2 Architectural Styles	Example
SQ	1 Code quality	2 Software Design	2.5.2 Quality Analysis and Evaluation Techniques	Example
		10 Software Quality	3.3.1 Static Techniques	Implicit
SQ	2 Software testing	4 Software Testing	4.5.2 Test Activities 4.6.1 Testing Tool Support	Implicit
		10 Software Quality	4 Software Quality Tools	Implicit

Table 6
Comparing Stack Overflow Topics to Themes (the first two columns refer to findings from Barua et al. (2014), the last column refers to the themes identified in our study).

Main topic	Stack Overflow Topic	Theme
Web-related	Website design/CSS	SD6 Front-end SD6.1 Graphical user interface
	Web development	SD5.4 Web development
	Web service/application	SD5.4 Web development SD1.1 Application programming interface
Data management	Database platform	–
	SQL	–
	XML	–
	Object-relational mapping	–
Platform-specific	.NET Framework	–
	Java	SD5.4 Web development SD5.5 Multi-platform development
	Windows/Visual Studio	–
	Web-related	SD2.1.1 Build automation SD1.3 Integrated development environment
	Mobile App Development	SD5.3 Mobile development
Security	Authentication/security	SA2.2.1 Security
	Cryptography	–
Quality assurance	Version control	SD1.5 Version control
	Testing	SQ2 Software testing
Knowledge/Experience	Job/experience	PD1.1 Jobs
	Learning	PD2 Training PD1.2 Workshops
General	Coding style/practice	PD2.1 Coding
	Problem/solution	–
	QA and links	–

analyzed what types of questions developers ask on Stack Overflow. Both studies describe knowledge needs of developers that they try to satisfy by consulting when using a Q&A website. Table 6 shows the comparison between the main topics identified by Barua et al. (2014) in Stack Overflow and our themes. Note that we could not associate any of our themes to the main topic "Data management" found by Barua et al. (2014). As can be seen in Table 6, the granularity of topics from Barua et al. (2014) and the themes from our study differ, i.e. a mapping is not always straightforward.

In Table 6 we did not associate the Stack Overflow topics "Problem/solution" and "QA and links" of Barua et al. (2014) with our themes. However, these topics are expressed in the chat rooms' discussions as we checked in the summaries of the chat rooms' conversations (see Step 1 of Section 3.2), rather than to a particular theme. In some degree, all conversations in our selection of Gitter chat rooms revolved around sharing experiences, code snippets and external links on how to solve issues.

Beyer et al. (2019) found that in Stack Overflow, users ask questions about the following (referred to as "types of question"):

- API usage: how to use an API;
- Conceptual: finding ways to solve an issue or implement a feature;
- Discrepancy: tentative solutions to solve an issue or implement features;
- Errors: report of errors and exceptions;
- Review: rechecking suggestions to find better ways to solve an issue or implement a feature;
- API change: differences in features between versions of an API;
- Learning: suggesting and asking for tutorial material.

Except for the theme *SD1.1 Application programming interface*, which is related to both types of question "API usage" and "API change" in the classification of Beyer et al. (2019), our themes cannot be clearly associated with these types of question. This is because the types of question are represented at a different level of granularity compared to our themes. This means, these types of questions are more about the content of actual questions asked on Stack Overflow, which are more focused and smaller in scope, compared to themes, which are about the main subject discussed by developers in a chat room. This also means that question types such as "Learning" and "Errors" (for example, in chat rooms under second-level theme *PD2 Training*) may be answered in chat rooms' conversation threads, but this would be another dimension of analysis.

4.5. Patterns and insights from themes

After identifying themes, types of chat rooms and types of discussions, we explored how they related to each other to look for insights about the discussions in the chat rooms. We found that:

- Developers discuss mostly in chat rooms that were related to general topics rather than in chat rooms of specific software projects (see Section 4.1).
- Chat rooms tend to discuss themes related to first-level theme SD — Software Development, regardless of whether a chatroom is project- or topic-related. This can be related to the nature of Gitter, which was initially created for software developers using GitHub or GitLab;
- Developers very often discuss web development (SD5.4) and in particular libraries for web development (e.g., the library Xcodeproj from the chat room *CocoaPods/Xcodeproj*[29]).

[29] https://gitter.im/CocoaPods/Xcodeproj.

Other re-occurring themes that are discussed in the selected Gitter chat rooms were *PD2.1 Coding* (nine occurrences), *SD6.1 Graphical user interface* (eight occurrences), *SA2.1 Patterns* (seven occurrences), *SD4 Software maintenance* (seven occurrences) and *SQ2 Software testing* (six occurrences);

- We confirmed that developers do indeed share knowledge related to Principles & practices, such as practices for coding (PD2.1) or design patterns (SA2.1) principles. Other themes related to this type of discussion were back-end (SD7), mobile development (SD 5.3), Blockchain (SA1.2), and testing (PD2.2). On the other hand, many of our themes were not identified in chat rooms we associated to Principles & practices. For example, software testing (SQ2), themes under design style (SA2.3) and themes under data intensive systems (SD5).

4.5.1. Trends in themes

Two of the chat rooms in our sample were created in 2013, ten in 2014, 39 in 2015, 24 in 2016, six in 2017 and six in 2018. However, considering that the total number of chat rooms created in each of these years is larger than our sample, it would be misleading to analyze trends in themes over time (e.g., by analyzing the themes of chat rooms created in a certain year). On the other hand, even in our limited and unbalanced set of chat rooms regarding the years when they were created, we observe similar developments of themes as in other discussion channels. For example, Barua et al. (2014) observed (a) an increasing trend of discussions on Stack Overflow around mobile development and web applications between 2008 and 2010, and (b) that discussions on Stack Overflow around jobs and version control were decreasing. In our sample, we found more chat rooms related to web development (including six new chat rooms on that topic between 2014 and 2015), but we did not find that the number of job- or version control-related chat rooms was decreasing.

4.5.2. Popularity, activity and engagement of themes

To expand the contextualization of themes found it our sample, we analyzed themes regarding their popularity, activity and engagement based on the number of messages, number of users and age of chat rooms related to that theme:

- *Popularity* of a theme is the interest of people in it, using the number of users and chat rooms related to gauge this interest (it does not consider the number of messages exchanged regarding a theme, but simply how many people are interested in a theme).
- *Activity* of a theme is the frequency of messages exchanged about it over time, using the number of messages in related chat rooms to measure this frequency (this metric does not consider the number of users, but rather the "traffic" related to a theme which can be from many or few users).
- *Engagement* of a theme is how constantly people discuss it over time, which was measured based on the number of messages exchanged by users during a certain period of time (i.e. this metric considers messages, time as well as users).

Therefore, in the context of the chat rooms selected for our study, we considered that popular themes are the most interesting themes in Gitter; active themes are the most frequently discussed; and engaging themes are the themes which kept more users participating in discussions over time. Table 7 shows the results of these metrics for each third- and fourth-level theme. The higher these metrics' score the better.

Popularity considers (a) the number of chat rooms per theme (denoted as r), (b) the number of users of all chat rooms related to a theme (denoted as u), and (c) the average number of users per chat room related to a theme (denoted as $size = u/r$). Regarding the popularity in terms of the number of rooms (r), the most popular themes are *SD5.4 Web development* (19 chat rooms) and *PD2.1 Coding* (nine chat rooms). In terms of the total number of users (u), the most popular themes are *SD5.4 Web development* (almost 22,000 users) and *PD2.1 Coding* (around 19,000 users). Finally, regarding popularity in terms of size, the most popular themes are *SQ1 Code quality* (8503 users on average per related chat room) and *SD2.1.1 Build automation* (8406 users on average per related chat room).

Activity refers to how active a theme was over time. Here, we first calculated the activity of each chat room as m/a (with m as the number of messages of a chat room and a as the chat room's age in months since the first message in the chat room was sent). Then, the activity of a theme is the average activity of all chat rooms related to that theme. The most active themes are *SD5.5 Multi-platform development* (9624 messages), *SD7 Back-end* (1890 messages), *SQ1 Code quality* (1639 messages) and *SD2.1.1 Build automation* (1621 messages).

Engagement refers to how engaged users were in discussing a theme over time. Here, we first calculated the engagement of each chat room as $m/(u * a)$ with m as the number of messages of a chat room related to a theme, u as the number of users of that chat room and a as its age in months (as above). Then, the engagement of a theme is the average engagement of all chat rooms related to that theme. Note that we multiplied the number of users by the age of the chat room because we aimed at checking the impact of users on the number of messages shared in a specific time frame. For example, we can compare the engagement of two themes T1 and T2 where T1 is represented in a 2-month old chat room with 1000 messages and 10 users, and T2 is represented by a 10-month old chat room with 1000 messages and 20 users. By calculating engagement of these two themes, we would check that T1 was more engaging than T2 because the T1 had fewer users (10 versus 20) that contributed with more messages in a shorter time frame. The most engaging themes are *PD2.2 Testing* (16 messages per user) and *SA2.2.1 Security* (9 messages per user).

We summarize our insights regarding popularity, activity and engagement of first-level themes below:

- SD — Software Development is the most popular theme in terms of the number of chat rooms and users, it is also the most active theme;
- SQ — Software Quality is the most popular theme in terms of average number of users in related chat rooms (e.g., rooms where we identified *SQ1 Code quality*);
- PD — Professional Development is the most engaging theme by considering the theme *PD2.2 Testing*;
- Although SA — Software Architecture did not stand out in terms of popularity, activity or engagement, themes under this theme are amongst the most engaging themes (e.g., *SA2.2.1 Security*).

4.6. Applicability of themes in slack chat rooms

Since thematic analysis cannot show generalization or transferability (Braun and Clarke, 2021a; Stol and Fitzgerald, 2018), we checked whether the themes identified in Gitter also appear in 184 chat rooms of public Slack communities (as described in Section 3.3). By checking the applicability of our themes, we could analyze whether and how our themes describe discussions of other chat rooms.

Regarding the characterization of Slack chat rooms (see Table 8), we found that similarly to Gitter chat rooms, most Slack chat rooms are related to topics (143 chat rooms) rather than

Table 7
Popularity (r: number of chat rooms per theme; u: number of users of all chat rooms related to a theme; $size = u/r$), activity and engagement by theme; numbers rounded to closest integer (e.g., engagement of 0 means that the average engagement was close to 0, i.e. very low).

First-level theme	Second-, third- and fourth-level theme	Popularity (r)	Popularity (u)	Popularity (*size*)	Activity	Engagement
PD	1.1 Jobs	2	272	136	3	0
PD	1.2 Workshops	2	123	62	72	1
PD	2.1 Coding	9	18,527	2059	807	4
PD	2.2 Testing	1	24	24	379	16
SA	1.1 Cloud computing	5	1304	261	519	2
SA	1.2 Blockchain	4	13,606	3402	920	0
SA	2.1 Patterns	7	2457	351	70	0
SA	2.2.1 Security	5	2308	462	1132	9
SA	2.3.1 Service-oriented architecture	1	5	5	2	0
SA	2.3.1.2 Containers	2	1519	760	542	0
SA	2.3.1.1 Microservices	1	54	54	3	0
SD	1.1 Application programming interface	3	5046	1682	1250	1
SD	1.2.1 Configuration	2	608	304	15	0
SD	1.2.2 Release	2	213	107	68	1
SD	1.3 Integrated development environment	1	4821	4821	1334	0
SD	1.4 Reverse engineering	1	215	215	44	0
SD	1.5 Version control	1	2984	2984	271	0
SD	2 Software deployment	2	345	173	72	0
SD	2.1 Continuous integration	1	291	291	588	2
SD	SD2.1.1 Build automation	2	16,813	8407	1621	0
SD	2.2 Distributed software	2	4650	2325	1394	0
SD	3 Software documentation	5	506	101	97	4
SD	4 Software maintenance	7	4073	582	169	4
SD	5.1.1 Big data	2	98	49	64	1
SD	[Machine learning] 5.1.2.1 Libraries	3	5857	1952	1197	3
SD	[Machine learning] 5.1.2.2 Research	1	556	556	551	1
SD	[Machine learning] 5.1.2.3 Systems	5	4483	897	112	0
SD	5.2 Games	3	1518	506	1014	4
SD	5.3 Mobile development	3	9457	3152	139	0
SD	5.4 Web development	19	21,555	1134	312	1
SD	5.5 Multi-platform development	1	7456	7456	9624	1
SD	6 Front-end	1	420	420	137	0
SD	6.1 Graphical user interface	8	14,286	1786	229	2
SD	7 Back-end	2	5436	2718	1890	1
SQ	1 Code quality	2	17,006	8503	1639	0
SQ	2 Software testing	6	7735	1289	928	1

projects (41 chat rooms). On the other hand, in terms of types of discussion, most Slack chat rooms are related to Other (55 chat rooms, e.g., *events* in community *codeCommunity*[30] and *main* of the *PIGSquad* community[31]), and Principles & practices (54 chat rooms, e.g., *beginners* in the *Elm* community[32]).

When applying our themes to Slack chat rooms, we realized that our themes could not describe some of their discussions fully. For example, the Slack chat room *lounge*[33] from the community "Space Coast Tech" was associated to the theme *PD1.2 Workshops* but we could not associate this room to other of our themes, which would describe the topic of such workshops. Therefore, for some chat rooms we only got a single perspective over what might be their main discussions. The complete list of Slack chat rooms with related themes is available online[9].

We labeled 173 Slack chat rooms with 36 themes identified in Gitter. We could not assign 11 themes to Slack chat rooms (see Table 9). The most frequent Gitter themes found in Slack chat rooms were *SD5.4 Web development* (40 occurrences), *PD2.1 Coding* (29 occurrences), *PD1.2 Workshops* (22 occurrences) and *PD1.1 Jobs* (22 occurrences). Although some themes did not apply to our selection of Slack chat rooms, they may be found in other Slack chat rooms which were not selected for this study. The 11 chat rooms that we could not assign to any theme seem to be related, based on their name and description, to bots development, issue management tools or system administration (servers, operational systems and virtual environments).

30 https://slofile.com/slack/codecommunity.
31 https://slofile.com/slack/pigsquad.
32 https://slofile.com/slack/elmlang.
33 https://slofile.com/slack/spacecoasttech.

Table 8
Slack chat rooms by type of room and discussion.

Type of discussion	Type of chat room		*Total*
	Project	Topic	
Frameworks	3	8	11 (6%)
Libraries	1	3	4 (2%)
Principles & practices	3	51	54 (29%)
Processes	0	12	12 (7%)
Products	13	18	31 (17%)
Tools	4	13	17 (9%)
Other	17	38	55 (30%)
Total	41 (22%)	143 (78%)	**184 (100%)**

Even though we cannot determine popularity, activity and engagement of themes for Slack as we did for Gitter (see the data collection for Slack chat rooms in Section 3.3), we checked whether popular, active and engaging themes in Gitter at least appeared in Slack:

- **Popular themes:** Frequent themes, such as *SD5.4 Web development* and *PD2.1 Coding* were also frequently associated to Slack chat rooms. Some popular themes in Gitter less frequent in Slack are: *SA1.2 Blockchain* (four occurrences) and *SQ1 Code quality* (one occurrence).
- **Active themes:** Active themes (high average number of messages per month), such as *SD5.5 Multi-platform development*, were associated to Slack chat rooms.
- **Engaging themes:** Engaging themes (high average number of messages exchanged by users over time), such as *SA2.2.1 Security* and *PD2.2 Testing*, were associated to Slack chat rooms.

Regarding the eleven themes that were not associated with Slack, since we did not intend to generalize themes (not a goal in thematic analysis Braun and Clarke, 2006, 2021b), these themes are as important for Gitter as the themes that were associated with Slack. The themes identified in Gitter reflect our selection of chat rooms at the time the study was conducted and are based on the analysis of descriptions of chat rooms and summaries of discussions. Hence, the themes not associated with Slack chat rooms are still valid for Gitter.

Out of the eleven Gitter themes that we could not associate with Slack chat rooms, seven themes (*SA2.2 Tactics*, *SA2.3 Style*, *SA2.3.1 Service-oriented architecture*, *SD1.2.2 Release*, *SD2.1.1 Build automation*, *1.4 Reverse engineering* and *SD2.1 Continuous integration*) are third- and fourth-level themes, which represent more specialized themes in comparison to second- and first-level themes. This may indicate that our selection of Gitter chat rooms presented some specific themes that were not applicable to the context of the selected of Slack chat rooms.

On the other hand, there were four second-level themes (*SA1 Technologies*, *SA2 Design*, *SD1 Resources*, and *SD5 Types of development*) that, unlike other second-level themes in our map, such as *PD1 Announcements* and *PD2 Training*; we could not associate to Slack chat rooms. In the development of our map of themes, some second-level themes (including these four themes) were created to *group* themes identified in Gitter chat rooms, as described in Section 3.2. This means, they were not directly identified in Gitter, but derived from commonalities of themes we did find in Gitter. However, most of the third- and fourth-level themes under these themes (e.g., *SA1.1 Cloud computing* and *SA1.2 Blockchain* under *SA1 Technologies*) were associated to Slack. This indicates that some of the second-level themes under *SA — Software architecture* and *SD — Software development* may not be easily associated to other contexts because they alone are less descriptive than the themes grouped by them (third- and fourth-level themes).

Table 9
Mapping of Gitter themes to Slack chat rooms.

First-level theme	Second-, third- and fourth-level theme	Frequency
PD	1 Announcements	1
PD	1.1 Jobs	22
PD	1.2 Workshops	22
PD	2 Training	11
PD	2.1 Coding	29
PD	2.2 Testing	1
SA	1 Technologies	0
SA	1.1 Cloud computing	5
SA	1.2 Blockchain	4
SA	2 Design	0
SA	2.1 Patterns	3
SA	2.2 Tactics	0
SA	2.2.1 Security	3
SA	2.3 Style	0
SA	2.3.1 Service-oriented architecture	0
SA	2.3.1.2 Containers	2
SA	2.3.1.1 Microservices	2
SD	1 Resources	0
SD	1.1 Application programming interface	5
SD	1.2 Management	2
SD	1.2.1 Configuration	4
SD	1.2.2 Release	0
SD	1.3 Tools	0
SD	1.3 Integrated development environment	1
SD	1.4 Reverse engineering	0
SD	1.5 Version control	1
SD	2 Software deployment	2
SD	2.1 Continuous integration	0
SD	2.1.1 Build automation	0
SD	2.2 Distributed software	4
SD	3 Software documentation	1
SD	4 Software maintenance	6
SD	5 Types of development	0
SD	5.1 Data intensive systems	1
SD	5.1.1 Big data	2
SD	5.1.2 Machine learning	4
SD	5.1.2.1 Libraries	1
SD	5.1.2.2 Research	1
SD	5.1.2.3 Systems	1
SD	5.2 Games	11
SD	5.3 Mobile development	15
SD	5.4 Web development	40
SD	5.5 Multi-platform development	1
SD	6 Front-end	7
SD	6.1 Graphical user interface	9
SD	7 Back-end	3
SQ	1 Code quality	1
SQ	2 Software testing	4

5. Discussion

5.1. Comparison to related work

In Section 4, we already compared our findings with the findings of other studies that identified themes and topics relevant to developers. In this section, we provide a discussion of our work in the broader context of developer instant communication on Gitter.

Our study aimed at identifying themes that would represent the main discussions of developers in chat rooms of instant messaging communication. By applying thematic analysis to the data of 87 Gitter chat rooms, we identified 47 themes (shown in Figs. A.4–A.7 in Appendix A) contextualized with two types of chat room and seven types of discussion. We organized these themes in a map to represent their context.

Related studies have also analyzed the content of instant messaging communication and Gitter. Each study focused on a different approach of data analysis. The results of our study could provide an additional layer and new angle of analysis of their findings:

- Sahar et al. (2021) focused their analysis on issue reports discussed in Gitter chat rooms; they analyzed who reports issues, what issues are discussed, the resolution time of issues, and how discussing issue reports in Gitter impact their resolution time. Our map of themes could complement this study, for example, by classifying their selection of chat rooms and contextualizing the issue reports analyzed.
- Ehsan et al. (2021) disentangled conversation threads in Gitter chat rooms and identified what makes a question getting responses and which features of a conversation are associated with the resolution outcome (e.g., URL, code snippets, active GitHub contributor). For this study, our map of themes could be used to classify each conversation thread and to contextualize the features associated to the resolution of questions.
- The study of Shi et al. (2021) could be also benefit from our map of themes to contextualize the conversation threads disentangled and analyzed by them. Shi et al. (2021) identified the communication profile, the community structure, the topics discussed and the interaction pattern of developers on Gitter. The topics identified by Shi et al. (2021) were based on the types of question organized by Beyer et al. (2019). As already discussed earlier in Section 4.4.2, even though these types of question are represented at a different level of granularity compared to our themes, describing more of the content of each message, our themes could be used to classify the conversation threads at a higher level of abstraction.

- Similarly to our study, Mezouar et al. (2021) applied thematic analysis. However, in that study, thematic analysis was applied to responses to a survey of users of Slack or Gitter chat rooms. By using this method, Mezouar et al. (2021) explored why developers use chat rooms, what is their perceived impact in software development processes, and what defines the quality of chat rooms and their related chat service (e.g., the available features). Our themes could be used to complement the study of Mezouar et al. (2021) to describe what are the themes developers have interest on discussing when selecting a chat room to participate.

5.2. Implications for practitioners

Based on our study, practitioners can get an overview of software engineering discussions and the interest of developers. For example, we found that most of the Gitter chat rooms cover topics related to libraries, frameworks and tools; this could be because Gitter is an instant messaging tool mostly associated to GitHub and GitLab projects. Among the chat rooms related to libraries, there are several about web application development. For instance, the first chat room created in Gitter about web development is Laravel, a PHP framework[34].

Based on the year of creation of chat rooms, practitioners can check which themes of discussion appeared and changed in developer communication over time. In our study, we observed the following appearance of themes:

- Themes regarding jobs, coding, machine learning, graphical user interface, Blockchain, service-oriented architecture, design patterns, software maintenance, software documentation, version control and web development started to appear in 2014, the same year as Gitter was released.
- In 2015, some of the themes were games, testing, workshops, software testing, security, back-end, big data, cloud computing, management resources, software deployment, integrated development environment (IDE), mobile development, application programming interface (API), and service-oriented architecture.
- Between the years 2016 and 2018, few new themes appeared, such as code quality, multi-platform development and reverse engineering in 2016.

Most of the chat rooms that we analyzed are topic- rather than project-related. However, there are chat rooms (related to Tools, Libraries and Products types of discussion) about the development or maintenance of particular projects. Therefore, newcomers interested in joining and contributing to these projects could use Gitter to familiarize themselves with problems and issues arising in those projects. This might be particularly useful for open-source projects where newcomers face barriers, such as a lack of orientation during on-boarding and poor project documentation (Steinmacher et al., 2019; Ford et al., 2018).

Practitioners interested in discussions about software development may access some popular Gitter chat rooms. At the time this research was conducted, the most popular chat rooms (number of messages or number of users) were *flutter/flutter* (toolkit for development of user interfaces)[18], *webpack/webpack* (a build automation tool)[35], *cypress-io/cypress* (a software testing framework)[36] and *akka/akka* (toolkit for building reactive applications in Java and Scala)[37].

Furthermore, the map of themes covered by developer chat rooms in Gitter represents a contextualized description of instant messaging communication of developer communities. This map of themes can be used to improve the management of the knowledge within such communities, a type of knowledge that is normally unstructured and hard to retrieve. For example, these themes could be used to tag communities, making them easier to be found in web searches.

Practitioners may also use our study to check how the selected chat rooms were aligned with standardized software engineering knowledge (i.e. SWEBOK Bourque and Fairley, 2014). The SWEBOK captures different broad knowledge areas that software engineering professionals should be aware of and which describe the different types of information needs. As we have shown before in Section 4.3, not all of these knowledge areas are covered by our themes. For example, we did not map themes to "9 Software Engineering Models and Methods", which are generally relevant in daily tasks of software development projects. To check whether these themes are really not covered in Gitter chat rooms or if we missed them because of our sampling, we searched Gitter chat rooms using explicit keywords such as "process", "model" and "method" (i.e. keywords practitioners may use to find chat rooms related to these themes). The chat rooms we found were not describing software engineering processes, models and methods as defined in SWEBOK chapters. For example, rooms we found about "process" were about software processes, threads, etc., rather than development processes; similarly, chat rooms found with keyword "method" were about Java methods or methods to implement a certain feature, rather than software development methods.

As we could not map our themes to some sections of SWEBOK (see details in Section 4.3), further studies may investigate why software engineering themes such as models and methods are not commonly discussed by developers in instant messaging communication. For example, we can explore if (and why) developers discuss less about higher-level process-related aspects of software development, and if this is because they are interested in more concrete (and implementation-related) questions and problems. For example, Mezouar et al. (2021) found that developers use Gitter chat rooms to support development activities such as "issue resolution" and "project tracking". Additionally, given that the analyzed chat rooms involve open source systems, we could explore if there is a lack of awareness related to processes or a lack of interest in models and methods.

5.3. Implications for researchers

In general, researchers may use Gitter to engage and identify the latest trends and challenges emerging in developer communities. The insights and map of themes obtained in this study can be used for further analysis (e.g., topics discussed in communities and structure of discussions) of developer communication, focusing on the characterization of the knowledge shared. Furthermore, Chatterjee et al. (2019) suggest linking information in Stack Overflow posts to information in Slack communities, since information on both platforms address similar concerns. Therefore, the map of themes identified in our study could be used to semantically link the content from instant messaging and online discussion forums. Additionally, we can use the map of themes in supervised information retrieval techniques applied to developer communication (e.g., to develop a machine learning classifier to identify relevant themes in communities).

By associating our themes with public Slack chat rooms, we demonstrated how applicable are the themes to other chat rooms (173 chat rooms out of 184 were associated to our themes). Researchers may identify further software engineering themes

34 https://gitter.im/laravel/laravel.
35 https://gitter.im/webpack/webpack.
36 https://gitter.im/cypress-io/cypress.
37 https://gitter.im/akka/akka.

in a different sample of chat rooms of Gitter or in chat rooms of different instant messaging tools (e.g., Microsoft Teams, Spectrum, Discord or chat rooms from Stack Overflow) using different research methods, e.g., Design Science Research (Hevner et al., 2004) to develop a theoretical framework of themes in instant messaging. However, researchers may need to adjust the data analysis process and may not be able to strictly replicate our study. For example, even though chat rooms on Stack Overflow (used in previous studies such as Ford et al., 2018) allow software developers to share knowledge on specific topics (e.g., RUST programming language[38]), an analysis based on users and activity may be difficult since number of users per chat room is not available. In Stack Overflow chat rooms everyone (logged in or not) can read discussions, but only users with 20 reputation points[39] can send messages. Further studies may, for example, explore how the themes in our map appear in Stack Overflow chat rooms considering the reputation of users (i.e. that users have to contribute to the Stack Overflow web forum first before contributing to discussions in chat rooms).

Additionally, researchers may associate and expand our themes using other data sets. For example, Chatterjee et al. (2020) created a data set with disentangled messages of three public Slack communities; Parra et al. (2020) created a data set with the messages of ten Gitter communities, where messages received a label identifying their purpose on the conversation.

5.4. Limitations and validity

Since we followed a qualitative approach in our study, typical strategies for generalizing findings (such as lab-to-field, sample-based or case-based generalization strategies (Wieringa and Daneva, 2015)) are not applicable. Therefore, the potential threats to validity of our study are related to trustworthiness. Trustworthiness typically refers to four aspects: credibility, transferability, dependability and confirmability (Boyatzis, 1998; Korstjens and Moser, 2018). Credibility is about the confidence in the truth-value of the research findings; transferability is about the applicability of the study results and to what extent the findings interest other researchers and contexts; dependability and confirmability are about whether methods were applied correctly and if the results are grounded in data, not in assumptions (Korstjens and Moser, 2018). In general, dependability and confirmability require that researchers check procedures and the data analyzed with study participants (Korstjens and Moser, 2018). Since we did not involve any participants, we only discuss credibility and transferability.

Credibility: Due to the nature of the data analyzed (chat room names and descriptions are short text with fewer than 100 words) and the importance of contextualizing themes and understanding relationships and meanings of themes, we did not use an automated approach (e.g., topic modeling), but opted for a "manual" reflexive thematic analysis. This analysis also included the manual analysis of automatically generated summaries of chat rooms. We therefore analyzed a limited number of chat rooms. However, we recognize that by analyzing more Gitter chat rooms, we may have identified different themes. At the same time we acknowledge that (Braun and Clarke, 2021b) argue that *data saturation* may not be a useful concept in thematic analysis studies because the idea of saturation may be "incompatible" with the idea of reflexive thematic analysis. Braun and Clarke (2021b) describe that in thematic analysis codes are not fixed, over time and during analysis they can evolve, expand, contract, be collapsed with other codes or be abandoned. This is because the coding process is often more interpretive and conceptual across an analysis and its developments and refinements reflect the researchers' deepening engagement with their data. In general, data saturation is typically achieved when "nothing new" can be identified in data, which is not necessarily needed for reflexive thematic analysis studies (Braun and Clarke, 2021b). During Step 5 of our thematic analysis, we rechecked our map of themes many times until we could not find other opportunities for changing it. This does not necessarily mean that we observed saturation, but it means that we reached a stage where we could not develop more of our map of themes.

To ensure credibility we used *triangulation* and *persistent observation*. For triangulation we applied data and investigator triangulation. In data triangulation, we collected data from different time periods, different chat rooms and consulted additional sources, e.g., related GitHub/Gitlab repositories. In investigator triangulation, two researchers analyzed the data and three researchers reviewed the results. For persistent observation, which is the constant review of the data collection and analysis (Korstjens and Moser, 2018), we constantly reviewed the description of themes and reassigned them as necessary.

We acknowledge that our map of themes is a snapshot of themes covered by chat rooms of Gitter at the time of the study, influenced by our understanding and interpretation of the data (as described by Braun and Clarke (2021b)). Since new chat rooms may appear and existing chat rooms might disappear, as much as other researchers may engage with the data and identify themes differently, we recognize that replications of our study may find different themes. Similarly, the description of chat rooms, etc. may not always be up-to-date or may change over time. We checked the Gitter API documentation[40] and unfortunately there is no tracking for the date of a chat room's creation and description update to get a "time stamp" of the analyzed data.

Similarly, we acknowledge that the results of our study may have been influenced by our selection of chat rooms. By using different keywords in the Gitter search (e.g., "agile") to obtain a list of chat rooms (as discussed in Section 3.1.2, Gitter does not provide a list of chat rooms but only allows browsing rooms), we may have found more chat rooms related to software development processes or to themes that we did not include in our map of themes. However, running a search with "agile" as search keyword led to only one chat room that met our selection criteria for Gitter chat rooms (see Section 3.1). Regarding Gitter chat rooms, we recognize that users may be members of more than one chat room. Therefore, popularity and engagement, i.e. metrics based on the number of users, consider the overall number of users per chat room instead of the number of unique users. Hence, a user of multiple chat rooms related to a theme contributed to higher popularity and engagement of themes related to these chat rooms.

Transferability: Although we checked the applicability of themes identified in Gitter with another instant messaging tool (Slack), we cannot transfer our findings to other chat rooms or to other developer communication channels. Besides the different contexts between the chat rooms in these instant messaging tools, we did not try to identify themes in Slack because we could not access as much information about chat rooms as in Gitter (e.g., number of users of chat rooms, etc.) due to data access restrictions of Slack (Costa Silva et al., 2019). This may have impacted the association of themes to Slack chat rooms. To clarify the context of chat rooms in these tools, we described the chat rooms of Gitter and Slack in detail, considering the metadata available, the procedures used to collect and analyze data and the limitations related to our findings.

38 https://chat.stackoverflow.com/rooms/62927/rust.

39 https://stackoverflow.com/help/whats-reputation.

40 https://developer.gitter.im/docs/rooms-resource.

We acknowledge bias and the influence of the authors' experience with software engineering in the identification of themes (i.e. researchers with different experiences may identify different themes). One of the authors who identified the themes has less experience with software engineering than the other authors. On the other hand, reflexive thematic analysis assumes that the background of researchers may affect the identification of themes (Braun and Clarke, 2006). Furthermore, as mentioned in Section 3, we considered the evaluation of thematic analysis studies proposed by Braun and Clarke (2021a) as a guideline for reporting our study.

Regarding the abstractive summaries used with the memos for the thematic analysis, we acknowledge that BART generates better summaries (in terms of readability and cohesiveness) with different words and sentences that do not belong to the original text by using more flexible expressions based on paraphrasing, compression or fusion (El-Kassas et al., 2021). However, abstractive techniques can generate summaries with repeated words (El-Kassas et al., 2021). Regarding the quality of summaries measured with ROUGE scores and considering the scores from the study of Paulus et al. (2018), our ROUGE scores were at the lower end of the recommended spectrum (e.g., some below 0.4 — see an overview of these scores in Table C.14 in Appendix C). This is particularly true for recall scores because there were few overlapping n-grams between our summaries and the original text (i.e. the messages of chat rooms). Still, given the precision the ROUGE scores gave us a "sense" of the quality of our summaries. Additionally, by reading the summaries we were able to judge that they were comprehensible enough to supporting our data analysis (more about the evaluation of abstractive summaries can be found in Gupta and Gupta (2019) and Schluter (2017)). In our study we used the summaries as a complementary source of data, rather than the sole source of data.

6. Conclusions

In this study we applied thematic analysis to 87 Gitter chat rooms to identify the themes they cover. Our insights offer a first step towards comprehending the characteristics of the knowledge shared in developer instant messaging communication. We noticed that in general, the themes were consistent with topics in the SWEBOK, a standardized knowledge framework for software engineering (e.g., software design and construction) and could be related to developer knowledge needs, such as software development practices, version control systems and design patterns. The map of themes we present in this paper represents a contextualized description of instant messaging communication of developers in the context of the selected chat rooms. Considering that the knowledge presented in discussions of developer communications is unstructured and hard to retrieve, this map of themes can be used to improve utilizing the knowledge within such communities, for example, by semantically tagging chat rooms.

Our results indicate that, in the context of our study, chat rooms in developer instant messaging communication are mostly about software development technologies and practices rather than development processes. However, as mentioned in Section 5.4, this finding maybe have been influenced by our selection of chat rooms. When comparing themes to SWEBOK chapters and subsections (see Section 4.3), we noticed that themes were mostly represented by topics in (sub)sections under the chapters related to software construction and design. Regarding developer information needs (see Section 4.4), our themes could be related to knowledge needs related to programming and testing. We also found that popular, active and engaging themes were related to lower-level implementation-related concerns (e.g., libraries for web development and integrated development environment) and practices (e.g., coding and testing).

By associating themes from Gitter to Slack chat rooms, we also verified the occurrence of themes related to software development technologies and practices. For example, *PD2.1 Coding* and *SD5.4 Web development* were also themes in Slack chat rooms. On the other hand, we observed a difference between Gitter and Slack chat rooms: In Slack, themes related to professional development appeared frequently (*PD1.1 Jobs* and *PD1.2 Workshops*). This may indicate that instant messaging is important for developers to keep informed about events and new job opportunities. Few chat rooms in Gitter are related to processes for software development, as their main topic of discussion. In Gitter we had one chat room for processes, while in Slack chat rooms there were twelve.

In future work we will disentangle discussions to identify the *structure* of the conversations threads within chat rooms (e.g., questions asked and related answers), similar to Chatterjee et al. (2020, 2021) and Ehsan et al. (2021). Based on the types of questions asked (e.g., such as the types of question used by Beyer et al. (2019) and Shi et al. (2021)), we aim at classifying the threads into types of conversations (see Section 2) to describe and understand how the knowledge is shared and built. As part of this, we will also analyze the actual *messages* of chat rooms in more detail and beyond abstractive summarization to identify more concrete topics discussed within the themes of the chat rooms. We will then be able to compare the topics within conversations to the themes and check if they are consistent. For instance, if a chat room's theme is about using a machine learning library, but the actual discussions are about general data mining concerns, then there is a mismatch between themes and topics. This would impact how practitioners find relevant chat rooms (e.g., they may not rely on the description and title of chat rooms). This will also enable us to expand our map of themes towards a more detailed framework of understanding synchronous messaging platforms for developers following systematic taxonomy development methods (Usman et al., 2017). Finally, we will characterize the users that engage in these chat rooms. We will analyze, for example, how many newcomers to software development are in these chat rooms in comparison to experts, and how users' background may influence the knowledge shared.

CRediT authorship contribution statement

Camila Costa Silva: Conceptualization, Investigation, Visualization, Data curation, Writing – original draft. **Matthias Galster:** Conceptualization, Supervision, Investigation, Writing – review & editing. **Fabian Gilson:** Validation, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Availability of data and material

The data used and generated for this paper is available online at https://zenodo.org/record/6668356.

Acknowledgments

We would like to thank the editor and the anonymous reviewers for their insightful and detailed feedback that helped us to significantly improve the manuscript.

Appendix A. Map of themes (complete)

In brackets we included the number of times a theme was identified in a chat room. Since we identified more than one theme in rooms (see Section 3.2), the total number exceeds 87 (see Figs. A.4–A.7).

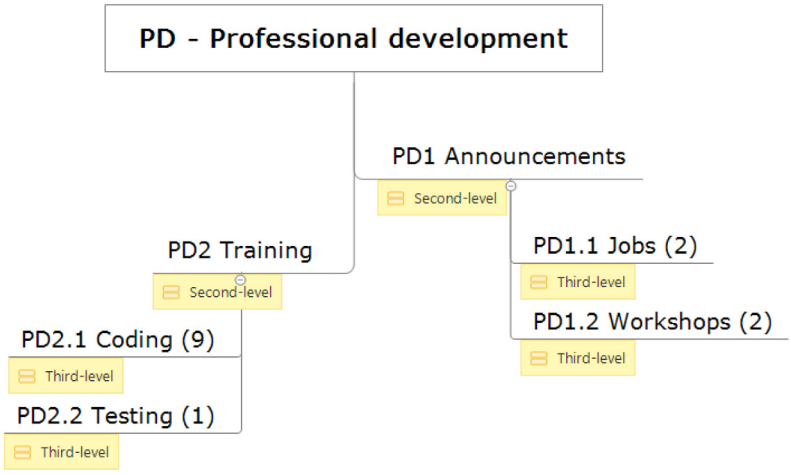

Fig. A.4. Professional development.

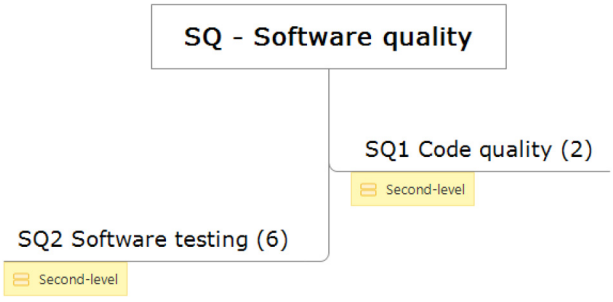

Fig. A.5. Software quality.

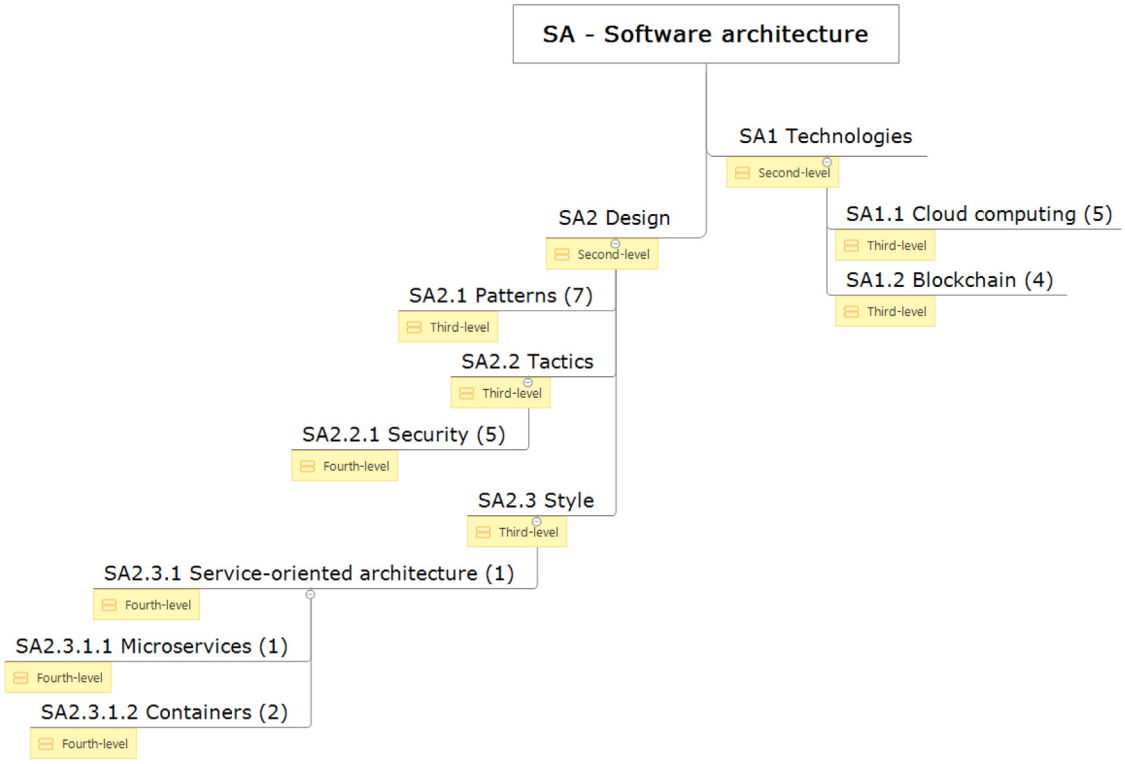

Fig. A.6. Software architecture.

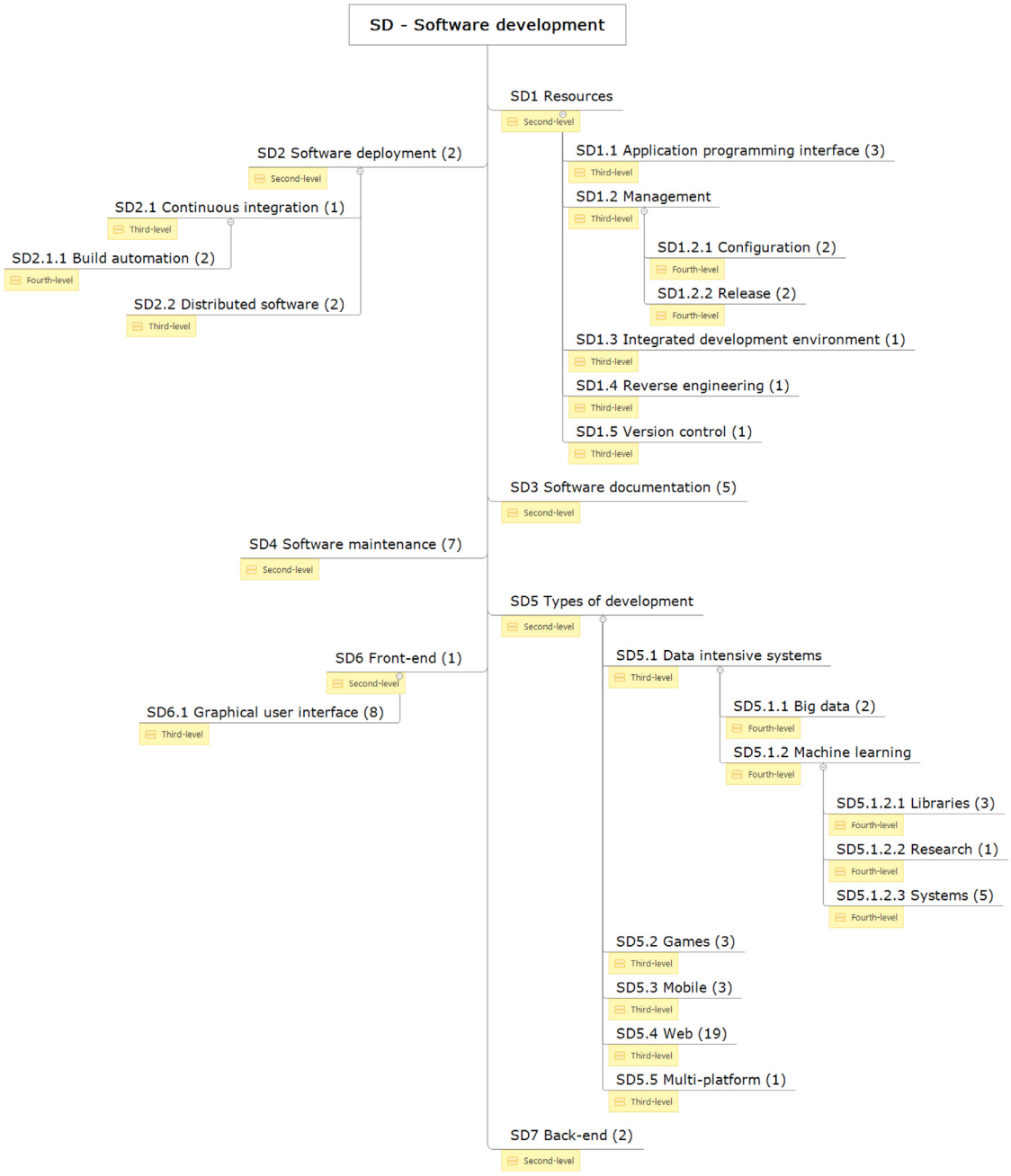

Fig. A.7. Software development.

Appendix B. Description of themes

See Tables B.10–B.13

Table B.10
Description of Themes under SQ — Software Quality[a].

SQ1 Code quality			
Third-level	Fourth-level	Description	Example
–	–	Experiences and advice on solving issues related to the use of tools for assuring source code quality	"Adds static typing to JavaScript to improve developer productivity and code quality." [Chat room: https://gitter.im/facebook/flow]
SQ2 Software testing			
Third-level	Fourth-level	Description	Example
–	–	Experiences, advice and issue solving related to the use of tools and frameworks for software testing	"A flexible lightweight multi-language test report tool with the possibility to add steps, attachments, parameters and so on." [Chat room: https://gitter.im/allure-framework/allure-core]

[a]Since first-level theme SQ has only second-level themes but no third- and fourth-level themes, the description refers to the second-level themes SQ1 and SQ2

Table B.11
Description of Themes under SA — Software Architecture.

SA1 Technologies Content related to technologies			
Third-level	Fourth-level	Description	Example
Blockchain	–	Experiences, issues and advice related to the use and development of solutions for Blockchain	Part of the memo (chat room has no description): Go Ethereum is an implementation of the protocol Ethereum for blockchains [Chat room: https://gitter.im/ethereum/go-ethereum]
Cloud computing	–	Experiences, issues and advice related to the use and development of solutions for Cloud computing	"Configuration library for JVM languages". Issues related to the configurations of Lightbend, a platform for Cloud computing [Chat room: https://gitter.im/lightbend/config]
SA2 Design Content related to software architecture design (frameworks, practices, requirements, plans)			
Third-level	Fourth-level	Description	Example
Patterns	–	Experiences and advice in principles and practices for design patterns	"A collection of samples to discuss and showcase different architectural tools and patterns for Android apps" [Chat room: https://gitter.im/googlesamples/android-architecture]
Style	Service-oriented architecture (SoA)	Experiences and advice related to a library for implementing Service-oriented Architecture style	"Tactician integration with the Bernard queueing library" [Chat room: https://gitter.im/thephpleague/tactician-bernard]
	SoA - Containers	Experiences and advice regarding the use of tools for containers	"RxJS powered state management for Angular applications, inspired by Redux" [Chat room: https://gitter.im/ngrx/store]
	SoA - Microservices	Experiences and advice regarding the use of microservices	"Seneca.js (http://senecajs.org/) NodeSchool workshop". Part of the memo: How to use, hints and problems related to Seneca that is a microservices toolkit for Node.js - http://senecajs.org/ [Chat room: https://gitter.im/senecajs/seneca-in-practice]
Tactics	Security	Experiences and advice related to software authentication issues, such as security protocols	"Welcome. Ask away! Unless otherwise specified we assume you're using the latest 5.x version of Spring Security" [Chat room: https://gitter.im/spring-projects/spring-security]

Table B.12
Description of Themes under SD — Software Development.

SD1 Resources Content related to technologies to support software development processes			
Third-level	Fourth-level	Description	Example
Application programming interface	–	Experiences and issues regarding the use of application programming interfaces provided by a software solution (not about documenting or developing APIs)	"Discussion for developers using Mixer's API" [Chat room: https://gitter.im/Mixer/developers]
Management	Configuration	Experiences, issues and advice related to managing software configurations	"A gradle release plugin". This plugin provides a Maven-like release process for projects using Gradle [Chat room: https://gitter.im/researchgate/gradle-release]
	Release	Feedback on the releases of new versions of a software	"Channel for release announcements and feedback" [Chat room: https://gitter.im/rchain/Releases]
Integrated development environment	–	Experiences, issues and advice related to the use of integrated development environments	"Ganache v7 is out! Read more about it here: https://trufflesuite.com/blog/introducing-ganache-7/". Part of the memo: Truffle is a development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine [Chat room: https://gitter.im/ConsenSys/truffle]
Reverse engineering	–	Experiences, issues and advice related to the use of a tool for reverse engineering	"A tool for reverse engineering Android apk files" [Chat room: https://gitter.im/iBotPeaches/Apktool]
Version control	–	Experiences, issues and advice related to the use of a version control resources	"Version Control on your Server. Visit https://gitlab.com/gitlab-org/gitlab-ce for more information" [Chat room: https://gitter.im/gitlabhq/gitlabhq]
SD2 Software deployment Content related to activities or technologies that make a software available for use			
Third-level	Fourth-level	Description	Example
–	–	Experiences, issues and advice related to the use of solutions for software deployment	"Fully automated version management and package publishing". Part of the memo: Semantic-release automates the whole package release workflow including: determining the next version number, generating the release notes and publishing the package (Gitbook.io) [Chat room: https://gitter.im/semantic-release/semantic-release]
Continuous integration	–	Experiences and issues related to the implementation of resources for continuous integration	"Jenkins Configuration-as-Code plugin". Part of the memo: Jenkins is open source automation server. [Chat room: https://gitter.im/jenkinsci/configuration-as-code-plugin]
	Build automation	Experiences, issues and advice related to the use of build automation platforms	"For questions please post on Stack Overflow and use the 'webpack' tag (http://stackoverflow.com/tags/webpack)." [Chat room: https://gitter.im/webpack/webpack]
Distributed software	–	Experiences, issues and advice related to solutions for distributed software	"This channel is available for all Akka enthusiasts—newbies as well as gurus—for the exchange of knowledge and the coordination of efforts around Akka; it is a community effort and resource and not backed by Typesafe. For more structured discussions please refer to the akka-user mailing list. Instead of a long Code of Conduct we rely upon common sense: be kind and respectful to those who are already here and to those who come after you, harassment of any kind will not be tolerated; in case of trouble contact akka.official. Please also note that this is not an official Akka support channel, for commercial support please contact Typesafe or visit Typesafe.com. Source: http://bit.ly/akka-gitter" [Chat room: https://gitter.im/akka/akka]
SD3 Software documentation			
Third-level	Fourth-level	Description	Example
–	–	Discussions about the documentation of a software	"Reference and documentation for Perfect" [Chat room: https://gitter.im/PerfectlySoft/PerfectDocs]

(*continued on next page*)

Table B.12 (*continued*).

SD4 Software maintenance			
Third-level	Fourth-level	Description	Example
–	–	Experiences, issues that appear during the maintenance of a particular software	Part of the memo (chat room has no description): Babel is a Python library that provides an integrated collection of utilities that assist with internationalizing and localizing Python applications in particular web-based applications (https://github.com/python-babel/babel) [Chat room: https://gitter.im/python-babel/babel/maintenance-corner]
SD5 Types of development Content related to activities or technologies for the development of different software applications			
Third-level	Fourth-level	Description	Example
Data intensive systems	Big data	Experiences, issues and advice related to the use of solutions for developing big data applications	"Talk to Vespa (http://vespa.ai) developers and users". Part of the memo: Vespa is a text search engine for big data [Chat room: https://gitter.im/vespa-engine/Lobby]
	Machine learning [Libraries]	Experiences, issues and advice related to the use of libraries for machine learning	"scikit-learn: machine learning in Python. Please feel free to ask specific questions about scikit-learn. Please try to keep the discussion focused on scikit-learn usage and immediately related open source projects from the Python ecosystem" [Chat room: https://gitter.im/scikit-learn/scikit-learn]
	Machine learning [Research]	Experiences, issues and advice related to the implementation of research projects applying machine learning	"Home of biologically-constrained machine intelligence research and technology." [Chat room: https://gitter.im/numenta/public]
	Machine learning [Systems]	Experiences, issues and advice related to machine learning-based systems	"Ruby Neural Evolution of Augmenting Topologies (NEAT)." [Chat room: https://gitter.im/flajann2/rubyneat]
Games	–	Experiences, issues and advice related to game development	"Help us test the develop branch for the MonoGame 3.7 release!" [Chat room: https://gitter.im/MonoGame/MonoGame]
Mobile development	–	Experiences, issues and advice related to the development of mobile applications	"The most popular HTML, CSS, and JavaScript framework for developing responsive, mobile first projects on the web." [Chat room: https://gitter.im/twbs/bootstrap]
Web development	–	Experiences, issues and advice related to web development	"A JavaScript utility library delivering consistency, modularity, performance & extras." [Chat room: https://gitter.im/lodash/lodash]
Multi-platform development	–	Experiences, issues and advice related to the use of a multi-platform development framework	"Flutter makes it easy and fast to build beautiful apps for mobile and beyond." [Chat room: https://gitter.im/flutter/flutter]
SD6 Front-end			
Third-level	Fourth-level	Description	Example
–	–	Experiences, issues and advice related to front-end development	"Vaadin has a high quality component set for building mobile and desktop web applications in modern browsers" [Chat room: https://gitter.im/vaadin/web-components]
Graphical user interface	–	Experiences, issues and advice related to the use of resources for the development of graphical user interfaces	"A collection of CSS3 powered hover effects to be applied to links, buttons, logos, SVG, featured images and so on. Easily apply to your own elements, modify or just use for inspiration. Available in CSS, Sass, and LESS." [Chat room: https://gitter.im/IanLunn/Hover]
SD7 Back-end			
Third-level	Fourth-level	Description	Example
–	–	Experiences, issues and advice related to back-end development	"shapeless: Generic programming for Scala — Latest stable release 2.3.3 — Code of conduct https://www.scala-lang.org/conduct/" [Chat room: https://gitter.im/milessabin/shapeless]

Table B.13
Description of Themes under PD — Professional Development.

PD1 Announcements Content related to advertisements			
Third-level	Fourth-level	Description	Example
Jobs	–	Job advertisements and job application tips	"Everything you need to kick ass on your coding interview" [Chat room: https://gitter.im/andreis/interview]
Workshops	–	Advertisement of software development-related workshops and developers professional networking	"Ep53 on CSS in large web apps with Chris Lienert on 6 Jan 2018, Saturday 11am. Join the live stream at https://live.webuild.sg/. All live episodes are released later for download. Subscribe to all the episodes via RSS (https://live.webuild.sg/feed.xml) or iTunes (https://itunes.apple.com/us/podcast/we-build-sg-live/id713804010)" [Chat room: https://gitter.im/webuildsg/live]
PD2 Training Content related to learning activities or experiences exchange between developers			
Third-level	Fourth-level	Description	Example
Coding	–	Experiences and advice on coding practices	"A general chat for developers. Don't ask to ask, just ask." [Chat room: https://gitter.im/gitterHQ/developers]
Testing	–	Experiences and advice on testing practices	"Python dev and test discussions. Not just pytest, but it's a common topic." [Chat room: https://gitter.im/TestAndCode/Lobby]

Appendix C. ROUGE scores for BART summaries

See Table C.14.

Table C.14
Descriptive analysis of ROUGE scores.

	ROUGE-1			ROUGE-2			ROUGE-L		
	r	p	f1	r	p	f1	r	p	f1
Min	0.0004	0.4649	0.0008	0.0001	0.1509	0.0003	0.0004	0.4649	0.0008
Max	0.8632	0.9683	0.7398	0.7814	0.8700	0.6207	0.8632	0.9643	0.7358
Mean	0.1038	0.8402	0.1400	0.0746	0.6842	0.0966	0.1027	0.8322	0.1385
Median	0.0194	0.8617	0.0379	0.0068	0.6978	0.0134	0.0193	0.8504	0.0376
Std Dev	0.1822	0.0923	0.2043	0.1545	0.1214	0.1692	0.1813	0.0930	0.2028

r: Recall; p: Precision; and f1: F1-score.

References

Alkadhi, R., Johanssen, J.O., Guzman, E., Bruegge, B., 2017. REACT: AN approach for capturing rationale in chat messages. In: International Symposium on Empirical Software Engineering and Measurement, Vol. 2017-Novem. pp. 175–180. http://dx.doi.org/10.1109/ESEM.2017.26.

Aniche, M., Treude, C., Steinmacher, I., Wiese, I., Pinto, G., Storey, M.-A., Gerosa, M.A., 2018. How modern news aggregators help development communities shape and share knowledge. In: Proceedings of the 40th International Conference on Software Engineering. ACM, Gothenburg, pp. 1–12. http://dx.doi.org/10.1145/3180155.3180180.

Antonino, P.O., Morgenstern, A., Kuhn, T., 2016. Embedded-software architects: It's not only about the software. IEEE Softw 33 (6), 56–62. http://dx.doi.org/10.1109/MS.2016.142.

Baltes, S., Diehl, S., 2018. Towards a theory of software development expertise. In: Proceedings of the 2018 26th Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, Lake Buena Vista, pp. 1–14. http://dx.doi.org/10.1145/3236024.3236061.

Baltes, S., Ralph, P., 2022. Sampling in software engineering research: a critical review and guidelines. Empir. Softw. Eng. 27 (4), 94. http://dx.doi.org/10.1007/s10664-021-10072-8, URL: http://arxiv.org/abs/2002.07764https://link.springer.com/10.1007/s10664-021-10072-8.

Barua, A., Thomas, S.W., Hassan, A.E., 2014. What are developers talking about? An analysis of topics and trends in stack overflow. Empir. Softw. Eng. 19 (3), 619–654. http://dx.doi.org/10.1007/s10664-012-9231-y.

Bass, L., Clements, P., Kazman, R., 2003. Software Architecture in Practice, second ed. Addison-Wesley Professional, Boston, p. 560.

Beyer, S., Macho, C., Di Penta, M., Pinzger, M., 2019. What kind of questions do developers ask on stack overflow? A comparison of automated approaches to classify posts into question categories. Empir. Softw. Eng. 25 (3), 2258–2301. http://dx.doi.org/10.1007/s10664-019-09758-x.

Bourque, P., Fairley, R., 2014. Guide to the Software Engineering Body of Knowledge, Technical Report, third ed. IEEE Computer Society, Washington D.C., p. 335, URL: www.swebok.org.

Boyatzis, R.E., 1998. Transforming Qualitative Information: Thematic Analysis and Code Development. Sage Publications, Thousand Oaks, p. 200.

Braun, V., Clarke, V., 2006. Using thematic analysis in psychology. Qual. Res. Psychol. 3 (2), 77–101. http://dx.doi.org/10.1017/CBO9781107415324.004.

Braun, V., Clarke, V., 2019. Reflecting on reflexive thematic analysis. Qual. Res. Sport Exerc. Health 11 (4), 589–597. http://dx.doi.org/10.1080/2159676X.2019.1628806, URL: https://www.tandfonline.com/action/journalInformation?journalCode=rqrs21.

Braun, V., Clarke, V., 2021a. One size fits all? What counts as quality practice in (reflexive) thematic analysis? Qual. Res. Psychol. 18 (3), 328–352. http://dx.doi.org/10.1080/14780887.2020.1769238, URL: https://www.tandfonline.com/action/journalInformation?journalCode=uqrp20https://www.tandfonline.com/doi/full/10.1080/14780887.2020.1769238.

Braun, V., Clarke, V., 2021b. To saturate or not to saturate? Questioning data saturation as a useful concept for thematic analysis and sample-size rationales. http://dx.doi.org/10.1080/2159676X.2019.1704846, URL: https://www.tandfonline.com/action/journalInformation?journalCode=rqrs21.

Braun, V., Clarke, V., Hayfield, N., Terry, G., 2019. Answers to Frequently Asked Questions About Thematic Analysis. Technical Report, The University of Auckland, Auckland, pp. 1–23.

Chatterjee, P., Damevski, K., Kraft, N.A., Pollock, L., 2020. Software-related slack chats with disentangled conversations. In: Proceedings of the 17th International Conference on Mining Software Repositories. IEEE/ACM, Seoul, pp. 588–592. http://dx.doi.org/10.1145/3379597.3387493.

Chatterjee, P., Damevski, K., Kraft, N.A., Pollock, L., 2021. Automatically identifying the quality of developer chats for post hoc use. ACM Trans. Softw. Eng. Methodol. 30 (4), 1–28. http://dx.doi.org/10.1145/3450503, URL: https://dl.acm.org/doi/10.1145/3450503.

Chatterjee, P., Damevski, K., Pollock, L., 2019. Exploratory study of slack q&a chats as a mining source for software engineering tools. In: Proceedings of the 16th International Conference on Mining Software Repositories. IEEE, Montreal, pp. 1–12.

Chatterjee, P., Nishi, M.A., Damevski, K., Augustine, V., Pollock, L., Kraft, N.A., 2017. What information about code snippets is available in different software-related documents? An exploratory study. In: Proceedings of the 24th International Conference on Software Analysis, Evolution, and Reengineering. IEEE, Klagenfurt, pp. 382–386. http://dx.doi.org/10.1109/SANER.2017.7884638.

Clarke, V., Braun, V., 2019. Guidelines for Reviewers and Editors Evaluating Thematic Analysis Manuscripts. Technical Report, The University of Auckland, Auckland, pp. 1–2, URL: https://cdn.auckland.ac.nz/assets/psych/about/our-https://cdn.auckland.ac.nz/assets/psych/about/our-research/documents/TAwebsiteupdate10.8.17reviewchecklist.pdf.

Costa Silva, C., Gilson, F., Galster, M., 2019. In: Franch, X., Männistö, T., Martínez-Fernández, S. (Eds.), Comparison Framework for Team-Based Communication Channels. In: Lecture Notes in Computer Science, vol. 11915, Springer, Catalunia, pp. 315–322. http://dx.doi.org/10.1007/978-3-030-35333-9{_}22, (chapter PROFES 201). URL: http://link.springer.com/10.1007/978-3-030-35333-9_22.

Dittrich, Y., Giuffrida, R., 2011. Exploring the role of instant messaging in a global software development project. In: Proceedings of the Sixth International Conference on Global Software Engineering. IEEE, Helsinki, pp. 103–112. http://dx.doi.org/10.1109/ICGSE.2011.21, URL: http://ieeexplore.ieee.org/document/6063155/.

Ehsan, O., Hassan, S., Mezouar, M.E., Zou, Y., 2021. An empirical study of developer discussions in the gitter platform. ACM Trans. Softw. Eng. Methodol. 30 (1), 1–39. http://dx.doi.org/10.1145/3412378, URL: https://dl.acm.org/doi/10.1145/3412378.

El-Kassas, W.S., Salama, C.R., Rafea, A.A., Mohamed, H.K., 2021. Automatic text summarization: A comprehensive survey. Expert Syst. Appl. 165, 113679. http://dx.doi.org/10.1016/j.eswa.2020.113679, URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417420305030.

Ford, D., Lustig, K., Banks, J., Parnin, C., 2018. "We don't do that here": How collaborative editing with mentors improves engagement in social Q&A communities. In: Proceedings of the 2018 Conference on Human Factors in Computing Systems. ACM, New York, pp. 1–12. http://dx.doi.org/10.1145/3173574.3174182, URL https://doi.org/10.1145/3173574.3174182https://dl.acm.org/doi/10.1145/3173574.3174182.

GitLab, 2020. Terms - GitLab. URL: https://about.gitlab.com/terms/.

Giuffrida, R., Dittrich, Y., 2013. Empirical studies on the use of social software in global software development - a systematic mapping study. Inf. Softw. Technol. 55 (7), 1143–1164. http://dx.doi.org/10.1016/j.infsof.2013.01.004.

Gupta, S., Gupta, S.K., 2019. Abstractive summarization: An overview of the state of the art. Expert Syst. Appl. 121, 49–65. http://dx.doi.org/10.1016/j.eswa.2018.12.011, URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417418307735.

Hevner, A.R., March, S.T., Park, J., Ram, S., 2004. Design science in information systems research. MIS Q. 28 (1), 75–105. http://dx.doi.org/10.2307/25148625, URL: http://dblp.uni-trier.de/rec/bibtex/journals/misq/HevnerMPR04.

Khatri, C., Singh, G., Parikh, N., 2018. Abstractive and extractive text summarization using document context vector and recurrent neural networks. In: Proceedings of the 24th Conference on Knowledge Discovery and Data Mining - Deep Learning Day. ACM, London, pp. 1–10, URL: https://arxiv.org/abs/1807.08000v2http://arxiv.org/abs/1807.08000.

Klein, H.K., Myers, M.D., 1999. A set of principles for conducting and evaluating interpretive field studies in information systems. MIS Q. 23 (1), 67–93.

Komi-Sirviö, S., Mäntyniemi, A., Seppänen, V., 2002. Toward a practical solution for capturing knowledge for software projects. IEEE Softw. 19 (3), 60–62. http://dx.doi.org/10.1109/MS.2002.1003457.

Korstjens, I., Moser, A., 2018. Series: Practical guidance to qualitative research. Part 4: Trustworthiness and publishing. Eur. J. Gen. Pract. 24 (1), 120–124. http://dx.doi.org/10.1080/13814788.2017.1375092.

Kruchten, P., 2008. The biological half-life of software engineering ideas. IEEE Softw. 25 (5), 10–11. http://dx.doi.org/10.1109/MS.2008.127, URL: http://ieeexplore.ieee.org/document/4602666/.

Lardinois, F., 2019. Microsoft says teams now has 13M daily active users. URL: https://tcrn.ch/3a1MG6t.

Lin, C.-Y., 2004. ROUGE: A Package for automatic evaluation of summaries. In: Proceedings of the Workshop on Text Summarization Branches Out. Association for Computational Linguistics, Barcelona, pp. 1–8.

Lin, B., Zagalsky, A., Storey, M.-A., Serebrenik, A., 2016. Why developers are slacking off: Understanding how software teams use slack. In: Proceedings of the 19th Conference on Computer Supported Cooperative Work and Social Computing Companion. ACM, San Francisco, pp. 333–336. http://dx.doi.org/10.1145/2818052.2869117, URL: http://dl.acm.org/citation.cfm?doid=2818052.2869117.

Markus, L.M., 2001. Toward a theory of knowledge reuse: Types of knowledge reuse situations and factors in reuse success. J. Manage. Inf. Syst. 18 (1), 57–93. http://dx.doi.org/10.1080/07421222.2001.11045671.

Mezouar, M.E., Alencar da Costa, D., German, D., Zou, Y., 2021. Exploring the use of chatrooms by developers: An empirical study on slack and gitter. IEEE Trans. Softw. Eng. 1. http://dx.doi.org/10.1109/TSE.2021.3109617, URL: https://goo.gl/forms/oX4UqWUDRcykBP372https://ieeexplore.ieee.org/document/9528018/.

Niinimäki, T., Lassenius, C., 2008. Experiences of instant messaging in global software development projects: A multiple case study. In: Proceedings of the 3rd International Conference Global Software Engineering. IEEE, Bangalore, pp. 55–64. http://dx.doi.org/10.1109/ICGSE.2008.27.

Parra, E., Ellis, A., Haiduc, S., 2020. GitterCom - A dataset of open source developer communications in gitter. In: Proceedings of the 17th International Conference on Mining Software Repositories, Vol. 5. ACM, New York, NY, USA, pp. 563–567. http://dx.doi.org/10.1145/3379597.3387494, https://dl.acm.org/doi/10.1145/3379597.3387494.

Pascarella, L., Spadini, D., Palomba, F., Bruntink, M., Bacchelli, A., 2018. Information needs in contemporary code review. In: Proceedings of the ACM on Human-Computer Interaction, Vol. 2. ACM, pp. 1–27. http://dx.doi.org/10.1145/3274404, https://dl.acm.org/doi/10.1145/3274404.

Paulus, R., Xiong, C., Socher, R., 2018. A deep reinforced model for abstractive summarization. In: Proceedings of the 6th International Conference on Learning Representations. International Society of the Learning Sciences, Vancouver, pp. 1–12, URL: http://arxiv.org/abs/1705.04304.

Robillard, P.N., 1999. The role of knowledge in software development. Commun. ACM 42 (1), 87–92. http://dx.doi.org/10.1145/291469.291476.

Romero, R., Parra, E., Haiduc, S., 2020. Experiences building an answer bot for gitter. In: Proceedings of the 42nd International Conference on Software Engineering Workshops. ACM, Seoul, pp. 66–70. http://dx.doi.org/10.1145/3387940.3391505.

Rus, I., Lindvall, M., Sinha, S.S., 2002. Knowledge management in software engineering: A state of the art report. In: IEEE Softw.. DoD Data & Analysis Center for Software (DACS), p. 53. http://dx.doi.org/10.1109/MS.2002.1003450.

Sahar, H., Hindle, A., Bezemer, C.-P., 2021. How are issue reports discussed in gitter chat rooms? J. Syst. Softw. 172, 110852. http://dx.doi.org/10.1016/j.jss.2020.110852, https://linkinghub.elsevier.com/retrieve/pii/S0164121220302429.

Schluter, N., 2017. The limits of automatic summarisation according to ROUGE. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Vol. 2. Association for Computational Linguistics, Stroudsburg, pp. 41–45. http://dx.doi.org/10.18653/v1/E17-2007, URL: http://aclweb.org/anthology/E17-2007.

Shi, L., Chen, X., Yang, Y., Jiang, H., Jiang, Z., Niu, N., Wang, Q., 2021. A first look at developers' live chat on gitter. In: Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, New York, NY, USA, pp. 391–403. http://dx.doi.org/10.1145/3468264.3468562, URL: https://2021.esec-fse.org/details/fse-2021-papers/31/A-First-Look-at-Developers-Live-Chat-on-Gitterhttps://dl.acm.org/doi/10.1145/3468264.3468562.

Slack, 2022. Features | slack. URL: https://slack.com/features.

Slofile, 2020. Slofile - discover slack communities. URL: https://slofile.com/.

Slofile, 2021. How it works. URL: https://slofile.com/how-it-works.

Soliman, M., Galster, M., Riebisch, M., 2017. Developing an ontology for architecture knowledge from developer communities. In: Proceedings of the International Conference on Software Architecture. IEEE, Gothenburg, pp. 89–92. http://dx.doi.org/10.1109/ICSA.2017.31.

Soliman, M., Galster, M., Salama, A.R., Riebisch, M., 2016. Architectural knowledge for technology decisions in developer communities: An exploratory study with stack overflow. In: Proceedings of the 13th Working Conference on Software Architecture. IEEE, Venice, pp. 128–133. http://dx.doi.org/10.1109/WICSA.2016.13.

Soliman, M., Salama, A.R., Galster, M., Zimmermann, O., Riebisch, M., 2018. Improving the search for architecture knowledge in online developer communities. In: Proceedings of the 15th International Conference on Software Architecture. IEEE, Seattle, pp. 186–195. http://dx.doi.org/10.1109/ICSA.2018.00028.

Soliman, M., Wiese, M., Li, Y., Riebisch, M., Avgeriou, P., 2021. Exploring web search engines to find architectural knowledge. In: Proceedings of the 18th International Conference on Software Architecture. IEEE, Stuttgart, pp. 162–172. http://dx.doi.org/10.1109/ICSA51549.2021.00023, URL: https://visdom-project.github.io/website.

Souza, L.B., Campos, E.C., Madeiral, F., Paixão, K., Rocha, A.M., Maia, M.d.A., 2019. Bootstrapping cookbooks for APIs from crowd knowledge on stack overflow. Inf. Softw. Technol. 111 (March 2018), 37–49. http://dx.doi.org/10.1016/j.infsof.2019.03.009.

Statista, 2018. Slack total and paying user count 2018: Statistic. URL: https://bit.ly/34mdXyY.

Steinmacher, I., Treude, C., Gerosa, M.A., 2019. Let me in: Guidelines for the successful onboarding of newcomers to open source projects. IEEE Softw. 36 (4), 41–49. http://dx.doi.org/10.1109/MS.2018.110162131.

Stol, K.-J., Fitzgerald, B., 2018. The ABC of software engineering research. ACM Trans. Softw. Eng. Methodol. 27 (3), 11–51. http://dx.doi.org/10.1145/3241743.

Stol, K.-J., Ralph, P., Fitzgerald, B., 2016. Grounded theory in software engineering research: a critical review and guidelines. In: Proceedings of the 38th International Conference on Software Engineering. IEEE/ACM, Austin, pp. 120–131. http://dx.doi.org/10.1145/2884781.2884833.

Storey, M.-A., Singer, L., Cleary, B., Figueira Filho, F., Zagalsky, A., 2014. The (R)evolution of social media in software engineering. In: Proceedings of the Conference on Future of Software Engineering. ACM, Hyderabad, pp. 100–116. http://dx.doi.org/10.1145/2593882.2593887, URL: http://dl.acm.org/citation.cfm?doid=2593882.2593887.

Storey, M.-A., Zagalsky, A., Filho, F.F., Singer, L., German, D.M., 2017. How social and communication channels shape and challenge a participatory culture in software development. IEEE Trans. Softw. Eng. 43 (2), 185–204. http://dx.doi.org/10.1109/TSE.2016.2584053.

Treude, C., Robillard, M.P., 2016. Augmenting API documentation with insights from stack overflow. In: Proceedings of the International Conference on Software Engineering. ACM, Austin, pp. 33–42. http://dx.doi.org/10.1145/2973839.2973847.

Treude, C., Robillard, M.P., Dagenais, B., 2015. Extracting development tasks to navigate software documentation. IEEE Trans. Softw. Eng. 41 (6), 565–581. http://dx.doi.org/10.1109/TSE.2014.2387172.

Treude, C., Storey, M.-A., 2011. Effective communication of software development knowledge through community portals. In: Proceedings of the 19th Symposium and the 13th European Conference on Foundations of Software Engineering. ACM, Szeged, pp. 91–101. http://dx.doi.org/10.1145/2025113.2025129, URL: http://dl.acm.org/citation.cfm?doid=2025113.2025129.

Usman, M., Britto, R., Börstler, J., Mendes, E., 2017. Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method. Inf. Softw. Technol. 85, 43–59. http://dx.doi.org/10.1016/j.infsof.2017.01.006, URL: https://linkinghub.elsevier.com/retrieve/pii/S0950584917300472.

Wieringa, R., Daneva, M., 2015. Six strategies for generalizing software engineering theories. Sci. Comput. Program. 101, 136–152. http://dx.doi.org/10.1016/j.scico.2014.11.013.

Xia, X., Bao, L., Lo, D., Kochhar, P.S., Hassan, A.E., Xing, Z., 2017. What do developers search for on the web? Empir. Softw. Eng. 22 (6), 3149–3185. http://dx.doi.org/10.1007/s10664-017-9514-4.

Zagalsky, A., German, D.M., Storey, M.-A., Teshima, C., Poo-Caamaño, G., 2016. How the r community creates and curates knowledge: an extended study of stack overflow and mailing lists. In: Proceedings of the 13th Working Conference on Mining Software Repositories. IEEE/ACM, Austin, pp. 441–451. http://dx.doi.org/10.1007/s10664-017-9536-y.

Zhu, W., Zhang, H., Hassan, A.E., Godfrey, M.W., 2021. An empirical study of question discussions on stack overflow. pp. 1–27, URL: http://arxiv.org/abs/2109.13172.

Camila Costa Silva is a Ph.D. student at the University of Canterbury, New Zealand.

Matthias Galster is an Associate Professor at the University of Canterbury, New Zealand.

Fabian Gilson is a Senior Lecturer at the University of Canterbury, New Zealand.