How Online Forums Complement Task Documentation in Software Crowdsourcing

Leticia S. Machado
UFPA, Belém, Brazil
leticia.schado@gmail.com

Igor Steinmacher
NAU, Arizona, US
igor.steinmacher@nau.edu

Sabrina Marczak
PUCRS, Porto Alegre, Brazil
sabrina.marczak@pucrs.br

Cleidson R. B de Souza
UFPA, Belém, Brazil
cleidson.dsouza@acm.org

ABSTRACT

An issue in software crowdsourcing is the quality of the task documentation and the high number of registered crowd workers to solve tasks but few submitted solutions only. This happens because uncommunicated or misunderstood requirements can lead crowd workers to deliver a solution that does not meet the customers' requirements or, worse, to give up submitting a solution. In this paper, we present an empirical study in which we analyzed task documentation and online forums messages associated with 25 Software Crowdsourcing (SW CS) challenges. The findings corroborate that weak documentation is a challenge in SW CS. Meanwhile, online forums allow crowd workers to gather additional technical and operational information that is not present in the official task documentation. We provide a stepping stone towards understanding the interplay between requirements and communication, to make it possible to improve SW CS development processes, practices, and tools.

CCS CONCEPTS

• Software and its engineering • Collaboration in software development

KEYWORDS

Software crowdsourcing, communication, documentation tasks.

ACM Reference format:

Leticia Machado, Igor Steinmacher, Sabrina Marczak, Cleidson R. B. de Souza. How Online Forums Extend Task Documentation in Software Crowdsourcing In *Proceedings 13th International Workshop on Cooperative and Human Aspects of Software Engineering, Seoul, South Korea. ACM, New York, NY, USA, 8 pages.*

1. Introduction

Software Crowdsourcing (SW CS) has become an emergent alternative strategy for software development. This strategy, based on the crowd, has been used for companies seeking to increase the speed of their software development efforts [1, 2, 3]. Platforms as Topcoder, uTest, and Passabrain usually explore a competitive approach and offer several software development activities, or challenges, to be performed by individual developers.

Take Topcoder as an example: a requester (or client) submits a task to it with a description and a monetary reward so that crowd members independently create a solution for the given task. Then, developers on the Topcoder platform check the task description and the rewards to decide if it is worthwhile solving the task. Next, interested developers will register for the task and submit their solutions using the platform. A core team on Topcoder and the requester will be responsible for evaluating the developers' solutions. The authors of top-ranked solutions are granted monetary rewards according to the pre-set rules. This process is known as open-call and the central parties involved are the requesters, the crowd and the platform. In this process the productivity and quality of the software solution depends heavily on the participation and efforts of the developers who registered and solved the task.

Several studies discuss SW CS focusing on different aspects of this strategy. For instance, some studies explore how to recommend developers to a given task [2, 3, 4]. These studies investigate how to find developers whose skills can meet the task requirements. Vaz and colleagues [5] focused on the importance of task documentation and pointed out what information is important to the crowd. Meanwhile, Boundreau and colleagues [6] examined how distinct sets of factors such as measures of effort, communication (frequency, timing, content), and participants' motivation shape not only effort choices but the extent of interactions and the collaborative style of work. Two other studies [1, 7] confirmed that interaction within the crowd in SW CS can be a very time-consuming activity.

While previous studies are insightful, it is important to keep in mind that SW CS is a socio-technical activity, i.e., a set of relationships that connect organizations, individuals, technologies and work activities [8]. This means that one aspect (like task documentation) is not isolated from other aspects (e.g., communication). In fact, previous studies [5] have observed that the task documentation in Topcoder often suggests crowd members to participate in the communication activities, i.e., to participate on the online forums. Despite this relationship between communication and documentation, to the best of our knowledge, previous studies have *not* studied these aspects in conjunction in SW CS projects, they have only been studied in isolation as we illustrated in the previous paragraph.

In this work, we address this research issue by studying the relationship between communication among crowd developers and the task documentation provided by task requesters. To be specific, we are answer the following research question: *How online forums extend task documentation in competitive SW CS tasks?*

To answer this research question, we choose Topcoder as our research platform because it is known as one of the earliest and largest crowdsourcing platforms in the market. Specifically, we selected 25 SW CS challenges and associated tasks from Topcoder. For each task, we analyzed the forum messages (communication) and task requirements (documentation). First, by reading each task documentation we identified the tasks that recommended using the online forums. Second, we qualitatively analyzed the messages exchanged in the online forums. We classified the messages to identify which aspects of the documentation were discussed in the forums. Thus, our goal was to understand the role of the forums as an extension to the task documentation to support crowd workers while working on

ICSEW'20, May 23–29, 2020, Seoul, Republic of Korea
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7963-2/20/05…$15.00
https://doi.org/10.1145/3387940.3391497

solutions to SW CS challenges. As part of this work, we analyzed which topics are discussed in the forums and how they relate to the documentation presented as part of the task.

Overall, the main contribution of this paper is the analysis of two related aspects — task documentation and forum communication — of competitive SW CS projects. By triangulating data from the documentation and the content of messages from online forums, it was possible to unveil how communication helps the crowd to build the solutions to the tasks in SW CS projects. For instance, we identified misunderstanding in the requirements, gaps in detailed technical and design specifications, and requests for deadline extensions that were addressed using the online forums. Our results suggest that forum communication extends task documentation understanding in competitive SW CS. In general, communication using forums supports the crowd's work to seek information and helps the crowd to deliver their solutions.

2. Software Crowdsourcing (SW CS)

Software Crowdsourcing (SW CS) refers to the act of externally transferring any software development task from a requester to a potential and undefined large group of online workers—the crowd, in an open call format [9] through a digital platform.

Software development models based on the crowd through competition and microtasking have drawn more attention of companies, autonomous developers, and startups since they display a flexible format to tap IT talent on demand by decomposing tasks into short, self-contained pieces of work that can be independently and quickly performed [10]. Competition is the model in which this study focused on. It consists of having crowd workers independently creating a solution, competing against each other by a monetary reward for task completion [10, 11].

The perception that there is a large crowd of available on SW CS platforms brings implications for requesters/companies who are seeking to speed up their software development process through crowdsourcing [11]. One of the main implications reported by the literature ([1, 3, 12, 13]) is that workers' decisions to participate in a task are highly volatile, making the software development strategy not efficient enough as a problem-solving model for clients that demand tasks. The negative effect on the crowd's interest in competitions (tasks) is an important issue in SW CS since crowd workers might register for a task, but do not submit solutions.

Other SW CS issues have raised the interest of researchers. For instance, several concerns regarding the adoption of SW CS from the requester's perspective using Topcoder are reported in an industry case study by Stol and Fitzgerald [1]. A requester representative stated that SW CS demands effort to prepare specifications, answer crowdsourcing community queries, review submissions, and resolve coordination and quality issues. More broadly, LaToza and colleagues [14] proposed a platform (CrowdCode) to support the decomposition of programming tasks and, in another study [15] they conducted an experiment with a two-phased software design competition, allowing the crowd to "borrow" from initially submitted solutions, which resulted in improved design quality.

2.1. Communication in SW CS

In SW CS, the crowd workers are usually geographically distant and do not have the opportunity for face-to-face interaction. This can inhibit informal communication and reduce trust [16, 17]. The importance of relationship management in outsourcing settings is well documented [18, 19]. Relations among crowd workers and between task requesters are of utter importance to get new information and share knowledge [20]. Tajedin and Nevo [20] argue that due to similarity between SW CS and open source software development methods, crowd workers can expect that communication quality and identification with the project are factors associated with success for the projects outsourced.

Several studies investigated Topcoder [2, 21, 22]. This platform offers services expressed in the form of tasks organized into categories of challenges as follows: Design, Development, Data Science, and Competitive Programming [21]. For each challenge (task) an asynchronous communication channel, an online forum, is created. In this forum, only the competitors who registered in that challenge can access and participate in. Each forum is organized into topics similar to threads. These forums are the place where the crowd can discuss about documents and artifacts and, communicate useful information to perform the task. In fact, some tasks require the crowd to seek for additional information in the forums [7].

Boudreau and colleagues [6] conducted an experiment in Topcoder where participants developed computational algorithms to NASA. They examined how distinct sets of factors such as cash incentives, communication (frequency, timing, content), and participants' motivation shape not only effort choices, but the extent of interactions and the collaborative style of work.

Other studies that mention communication in the context of SW CS are [1, 7]. Stol and Fitzgerald [1] discuss that communication via forums can impose an overload and be time-consuming when responding questions from the crowd and it tends to require a special person (in-house specialist) to answer these questions. The second study, by Machado and colleagues [7], provide a similar argument about the communication and coordination efforts due to latency of time between posed questions and delayed responses.

2.2. Documentation in SW CS

A task in SW CS generally follows a specific flow composed of announcement, selection, completion, delivery, and validation [2]. In SW CS a task represents the problem or part of the problem demanded by the client and is generally defined by the platform. This process of definition and decomposition of a task into micro-tasks is considered one of the major challenges of this development model [1]. When fragmenting the task into micro-tasks, the platform needs to ensure that each micro-task has sufficient information to enable its development. The task documentation cannot be too specific that loses focus nor too broad that challenges its comprehension. In general, task requirements documentation in SW CS turns out to be a crucial factor for the solution of the task given that the quality of the resulting documentation is likely to affect the crowd members' performance and success [4, 5, 20].

There is a fine balance in providing a sufficiently detailed specification for the task on the one hand, and stifling innovation with overly detailed specifications on the other [1, 12, 21]. As Saremi et al. [12] narrate "clients discovered that contest participation decreased if they were unclear about what problems they wanted to solve or presented problems that were too complex in scope". Along the same line, Tajedin and Nevo [20] suggest that projects which can be decomposed into small modules with limited interdependencies and clear requirements are more likely to succeed.

Despite the importance of requirement specification to software development both in a "traditional" strategy (collocated, distributed, outsourcing, etc) or in SW CS, our knowledge about documentation in SW CS tasks is limited. Stol and Fitzerald [1] compare an SW CS project with in-house software development and conclude that the requester "spent significant time and effort on writing specification documentation in an SW CS project, much more than if the software was being developed internally." Meanwhile, Zanatta et al. [24] discuss the barriers faced by newcomers in terms of documentation in SW CS projects including inadequate procedures and other problems related to task documentation.

Machado and colleagues [25] provide a similar argument in a case study on Topcoder challenges: "documentation associated with the accomplishment of the task is fundamental for its effective execution. Difficulties with documentation (lack or incomplete) may be linked, in a way, to many problems of assembly environment setup for the execution of the task."

Both software crowdsourcing and open source software (OSS) development rely on a crowd of developers in diverse locations working on different yet related tasks in a common environment. In this regard, software crowdsourcing is subject to barriers similar to OSS. These include technical problems, poor or no complementary documentation, unclear tasks, and cultural differences [24]. Steinmacher et al. [18] reported a set of barriers faced by newcomers in OSS projects. One of them is documentation and refers to the need of newcomers to learn the project's technical and social aspects to be able to contribute. These authors exemplify some documentation problems including outdated documentation, unclear code comments, information overload, and lack of documentation.

3. Research Method

To answer our research question, we analyzed data from Topcoder [21] task documentation and online forum communication.

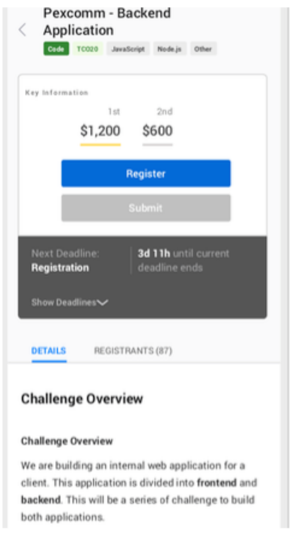
Figure 1 shows the main attributes of a task in the Topcoder platform. Each task is organized as an open contest and it has many attributes containing task name, registration and submission deadlines, challenge overview, prizes value, and required programming languages and technologies.

3.1. Study setup

On Topcoder, each SW CS task is organized as a challenge through an open competition. Designed to enable wide task accessibility and self-selection, Topcoder allows crowd developers to freely choose to engage in tasks based on their personal skills, experience, and interests [3, 12]. This paper focuses on tasks in the Development Challenge category and Code subcategory [21]. Each task has a scope and is composed of technical requirements that define the expected behavior of the solution to the task and, if necessary, the necessary interfaces to integrate with other parts of the system**.**

In each Topcoder's competition, the official communication mechanism is the online forum. Forums are the only channel used for task communication and coordination and they are just visible to the participants who register to the task [7, 21].

On Topcoder, there is a role called copilot. The person playing this role follows up the task development, acting as a mediator between the platform and the crowd. The copilot may be a member of the crowd playing the representative role of the client who requested the task. The copilot's main responsibilities are related to managing questions submitted by crowd members in task-based forums and answering them by interacting with the task requester, updating information, and so on [22]. In short, Topcoder forums allow communication (i) among crowd members who registered for the task and (ii) between crowd members and copilots.



Challenge Overview

Challenge Overview

We are building an internal web application for a client. This application is divided into **frontend** and **backend**. This will be a series of challenge to build both applications.

Figure 1: Topcoder's documentation structure

3.2. Topcoder Data extraction

We collected two types of data from Topcoder:

- Communication messages: In this case, we analyzed 25 challenges (tasks) from Topcoder's Development – Coding category and their associated 25 forums. We collected the forum messages from these 25 tasks during the registration and submission phases. We classified each message to identify meanings associated with them (see Section 3.3), and
- Task documentation: we collected and analyzed the requirements documentation from the same 25 tasks (challenges) described in the previous item. We investigated in the tasks' documentation if there was a recommendation for the registered crowd members to access the associated online forums. Tasks represent the starting point of the SW CS activity. It plays an important role through which crowd developers will know what they need to produce.

As previously mentioned, we selected a sample of challenges for analysis among the several development challenges hosted on Topcoder. The sample selection was based on three main criteria: (i) based on the study by Dubey's and colleagues [23], we focused on the two months with the greatest number of tasks available on Topcoder: July and August; (ii) challenges with a minimum of 15 registered competitors; and (iii) challenges with a financial reward of at least 500 US dollars. These criteria considered a reasonable number of competitors and awards to generate a discussion in the communication forum. Following these criteria, 25 competition challenges (see Table 1) were selected from the pool of tasks in the from the coding subcategory of the development category, and subsequently analyzed.

We analyzed the forum and task documentation associated with each 25 coding challenge. The documentation description was, sometimes, pointing to using the forum communication for additional information. We detail the analysis of task documentation and communication forums in the next sections.

In Table 1, we refer to the tasks that took place in July 2017 as TJ1 to TJ9, and those from August are identified as TA1 to TA16. We analyzed the 1,040 messages from the respective online forums. From these, 493 were from copilots and 547 from crowd members. The last lines of this Table present the descriptive statistics of the analyzed messages.

Table 1: Tasks data overview

Task ID	Registered	Submitted Solutions	Forum Participants	Total Messages	Forum Recommendation
TJ1	**62**	5	6	**102**	yes
TJ2	43	1	6	21	no
TJ3	34	1	15	**60**	yes
TJ4	42	2	4	18	no
TJ5	33	3	5	43	yes
TJ6	52	**7**	9	21	no
TJ7	27	2	4	8	yes
TJ8	43	4	14	**76**	yes
TJ9	48	3	5	19	yes
TA1	47	4	7	34	no
TA2	48	3	8	**115**	yes
TA3	17	1	8	71	yes
TA4	28	2	4	27	yes
TA5	26	2	3	**7**	no
TA6	19	3	8	47	yes
TA7	43	4	6	32	yes
TA8	33	3	4	22	yes
TA9	26	4	5	29	yes
TA10	20	3	7	**21**	no
TA11	50	3	9	36	no
TA12*	24	0	0	2	yes
TA13	52	3	**45**	**122**	yes
TA14	49	2	3	**35**	no
TA15	36	2	13	**53**	yes
TA16	31	3	7	**19**	no
Total	**933**	72	**178**	**1040**	**yes =16 / no =9**
Avg.	**37.32**	**2.88**	**7.12**	**41.16**	
Min.	**17**	**1**	**0**	**2**	
Max.	**62**	**7**	**43**	**122**	

We split the tasks into Group 1 (G1) and Group 2 (G2). We refer to the 16 challenges that recommended the use of the forum in the task documentation as G1, while G2 represents the 9 challenges that did not have such explicit recommendation. Although we split the challenges, it is worth to highlight that all 25 tasks had forums available, since this was one of our task selection criteria.

TA12 is an outlier regarding the forum behavior since no messages were exchanged by the crowd members in it. Only the platform moderator used the forum to share the task documentation.

Table 1 shows that although the number of crowd members who registered to participate is high (Registered column), few solutions are effectively submitted (Submitted Solutions column). In fact, tasks TJ2, TJ3 and TA3 had only one solution submitted. In contrast, 7 active crowd members submitted their solutions to challenge TJ6. This is consistent with previous work by Yang and colleagues [3] who report a high number of quitters in SW CS platforms. Finally, a significant smaller number of crowd members participate in the forum (Forum Participants column).

Data extracted from the forums were: date and time, thread, sender, recipient, and message content. A total of 1,040 messages were extracted, analyzed and classified (see Section 3.3).

Recalling, the typical structure of a general task description consists of: task name, registration and submission deadline, challenge overview, prizes value, required languages and technique (as showed in Figure1).

3.3. Data Analysis

We qualitatively coded each message based on its content aiming to identify meanings associated with the forum messages. The coding process was conducted independently by two authors between December 2017 and February 2018. Then, they reviewed every code together to reach an agreement about the final categories and topics to be used.

Forum messages' content analysis was inspired on Grounded Theory coding procedures [27], i.e., our qualitative coding procedures did not consider any preconceived list of codes nor used any theoretical framework to guide the coding process. We let the codes emerge from the analysis process. More specifically, we used two code types: categories and topics. Categories grouped concepts together under a more abstract high-order concept to explain what was happening when the crowd members exchanged messages via forum. For instance, crowd members reported problems or shared a tip about a particular technology. Topics were used to help understanding which subjects were discussed in each forum message. Examples include requirements, libraries, output or access. By using these two code levels we could identify the purpose of a message and its related discussed subjects.

We also analyzed the task documentation from the 25 challenges. Each challenge had a short general task description, open to all Topcoder competitors, and a more detailed supplementary documentation (e.g., UML diagrams, prototype screens). Similarly to the forums, this detailed documentation was restricted to those who registered to the task.

To analyze the documentation, we classified each piece of information according to the documentation structure model proposed by Vaz and colleagues [5]. For each challenge, we mapped which of the items could be found in the documentation (e.g., environment set up, required tools, acceptance and testing criteria), as well as explicit indications to seek information in the forum. This classification was conducted by one of the authors and reviewed in meetings with 2 others. In cases when the categorization was questioned, the group discussed until reaching consensus.

From the previous analysis on task documentation, we observed two types of challenges: (i) those that explicitly recommend using the task's online forum during the challenges and (ii) those that do not explicitly recommend using the forum. In practice, we have two groups: Group 1 (tasks that explicitly recommend using the forum) and Group 2 (tasks did not use explicitly forum recommendation).

By combining the analysis of the task documentation and the analysis of the forum messages, we were able to identify if the crowd members use the forums to seek information about the topic(s) that was(were) recommended in the task documentation. In addition, we were also able to observe whether crowd members communicated in the forums about *other* topics during the challenges, i.e., topics that were *not* recommended in the task documentation. We discuss our results in the following section.

4. Results

In the following sections, we present the results of our analysis, which answer our research question. We organize them by the two groups of tasks we identified: Group 1 - tasks that explicitly recommended using the forums, and Group 2 - tasks that did not make this explicit recommendation.

4.1. Group 1: Tasks that Explicitly Recommend using the Forum

We identified that 64% of the analyzed challenges (16/25) explicitly recommended the participants to use the online forums during the challenges' development. We refer to these 16 challenges as Group 1 (G1). As we expected, crowd members used the forums to exchange messages about the subjects indicated in the task documentation. Table 2 lists these topics; see Recommended Topics column. For instance, task TJ3 suggested checking the forum to find information about the topic *Project Requirements* and in fact several requirements messages (69) were sent by the crowd (see #of msgs Rec. Topics column). In addition, other topics like *Processing* and *Deadline* were also discussed in the online forum. We called these Extra-Topics, i.e., messages that were not expected to be discussed on the forum. Table 2 summarizes the total messages discussed in Group 1 (828 messages) organized into those topics explicitly recommended (273 messages) and those extra ones (555 messages) exchanged through the forum. These Extra-Topics messages represent the "unexpected" messages posted in the forum: messages that go beyond the topics initially suggested.

As observed in Table 2, the most frequent posts discussed by crowd members on Group 1 for messages about the Recommended Topics were: *Access* (120 messages)*, Requirements, Library, Style, Processing,* and *Connection.* These topics indicate that crowd members send messages in the forums to get technical and functional requirements specification, to request access for code repositories or private keys and other API issues.

The high number of messages about the recommended topic *Access* (120 as per Table 2) during challenges is consistent with the task instructions, i.e., crowd developers seek information via forums about access as provided in the task documentation. According to our qualitative classification, this topic is about requests from crowd members to the copilot to grant access to certain development environments, for accessing code components, containers, files or private keys necessary for developing the task solution. The second mostly communicated topic is *Requirements* (69 messages). This topic refers to discussions related to technical and functional requirements specifications about the task. This may include different artifacts and files used to support the documentation.

In addition, the most discussed Extra-Topics were, in this order: *Requirements* (156 messages)*, Processing, Access, Input, Deadline, Output, Unit, Library, Style,* and *Connection*. There is a certain overlap between the topics since some tasks recommended a specific topic to be discussed and the same topic appears as an extra topic in other tasks.

According to our qualitative coding, the most discussed Extra-Topic *Requirements,* with 156 messages, suggests that crowd members likely see problems in the task requirements, and therefore send several messages via forums about them, even if this topic was not suggested in the task documentation. The second-

Table 2: Topics recommended and discussed in Group 1. The number in between the parenthesis indicate the number (#) of tasks in which the topic was discussed

Recommended Topics	**#of msgs Rec. Topics**	**Extra-Topics**	**# of msgs Extra Topics**	**# of msgs Total**
Access (5)	120	Access (7)	90	210
Requirements (7)	69	Requirements (7)	156	225
Library (3)	39	Library (9)	26	65
Style (4)	33	Style (2)	14	47
Processing (1)	2	Processing (7)	97	99
Connection (1)	10	Connection (2)	5	15
		Deadline (11)	58	58
		Input (7)	60	60
		Output (4)	45	45
		Units (1)	4	**4**
Total	**273**		**555**	**828**

largest discussed Extra-Topic was *Processing* (97 messages as per Table 2) and it refers to sharing coding compilation code or execution errors found in the solution or in associated development servers [22]. Discussion about *Access,* totaling 90 messages, refers to granting access to repositories and tools associated with the task.

More broadly, the additional 1/3 of discussed Extra-Topics (namely *Deadline, Input, Output*, and *Units*; total of 167 messages out of the 555 extra-ones) demonstrate that the crowd members communicate with each other and with the copilots to organize and clarify their understanding about the task. For instance, the topic *Deadline* refers to messages that discuss the date and time set to submit the task. Meanwhile, the topics *Input* and *Output* are about variables and resources related to a task, and *Units* is about the measures of a given input/output variable.

To illustrate the richness of the exchanged messages, we present some excerpt quotes obtained from the analyzed Topcoder tasks. The quotes that illustrate the most mentioned topics about *Requirements* and *Access* from Group 1 (recommend topics forum uses) are presented below. In the quote about *Requirements* a developer – Coder A – is seeking to understand the specification documentation provided in the challenge. She is unsure about a particular section so she asks a question. Before the copilot has the chance to answer the question, another developer (Coder B) suggests something and they engage in a short dialog. Later, the copilot joins the conversations and answers a different question posed by Coder B telling them to follow the specification requirements.

[Coder A] "*All sections of the specification document seem to have a good purpose, except Section 9.3. Should we use the formulas in Section 9.3? How? (It seems the equivalent engineering constants are not needed.)*"

[Coder B] "*If you have a look at summary file, it says that for laminates we need to output engineering constants. Engineering constants (Laminate only)*"

[Coder A] "*Thank you [coder]. Now the "how" question remains: #1. Is t sub k = T sub k ? #2. What is t sub lam? #3. What is C sub {1,1,Lam}? #4. C sub {2,2,Lam} and so on? We have matrices A, B,* *and D; but not C. I checked this fast in the given links, but couldn't this information."*

[Coder B] "*In the specification it is given, the program shall use a standard directory as follows: README.txt Should we need to give txt read me file or Markdown read me file?"*

[Copilot] "*Let's use txt since that's what's required in the spec."*

In the following quote, a coder asks to help for co-pilots by granting them *access* to the required repositories that contain complementary information about the tasks.

[Coder A] "*Dear co-pilot, Please re-check to add my handle = hieplt 404 not found now."*

[Copilot] "*See here – [site] You would have received an invite mail from Github. Please make sure to check your spam folder."*

[Coder A] *"Please resend invite for me. I also check on Spam and All Inbox folder. I see nothing. Sorry!"*

4.2. Group 2: Tasks that did not Explicitly Recommend using the Forum

Group 2 comprises the tasks in which there was *no* explicit recommendation about using the forum in their documentation. This group represents 36% (9/25) of the analyzed tasks. A total of 212 messages were sent about these 10 discussed topics distributed (see Table 3). The largest number of messages were related to the *Requirement* topic (81), followed by *Library* (37), *Deadline* (26), and *Processing* (24) as presented in Table 3.

In addition to being the topic most discussed, *Requirements* was also the topic discussed related to the higher number of tasks—8 of the 9 tasks in Group 2. *Style* and *Input* were the topics discussed by the smaller number of tasks—one each only.

It is important to remember that in Group 2 the task document did not inform the need for the crowd to use the forum to seek information about a certain topic to solve the task. However, the crowd used the online forum to uncover information about requirements, scorecard, library, deadline, processing, connection, output, input, access, and style anyways. The fact that crowd workers seek information about requirements might suggest that the task documentation is not clear or that it does not have enough details to be fully comprehended on its own and later implemented. Further, the higher number of messages related to *Library* in Group 2 might suggest that the documentation does not have enough details about the frameworks or libraries needed to build the solution.

In the following quote, we illustrate an example from Group 2 about the *Library* topic where two coders (Coder A and Coder B) ask the copilot to check the information about an excerpt of programming code from the data API on the functional specification (task documentation). In this case, the copilot agrees with the answer by Coder B.

[Coder A] "*This requirement is listed in previous ldap challenge. Authenticate user via username/password (return user NT ID, and list of groups the user is a member of). So, we should still extract ldap groups for ldap user info for login method?*

[Coder B] *"I think that's unnecessary for CRUD API. I don't see any place in the Functional Spec that makes use of logged in user's LDAP groups"*

[Co-pilot] *"Correct, we don't need user groups any more (requirements have changed a bit since the last challenge)"*

Table 3. Topics Discussed in Group 2.
The number in between the parenthesis indicate the number (#) of tasks in which the topic was discussed

Discussed Topics	# of Messages
Requirements (8)	81
Library (4)	37
Deadline (5)	26
Processing (7)	24
Connection (2)	13
Access (5)	10
Output (2)	9
Style (1)	6
Scorecard (2)	5
Input (1)	1
Total	**212**

Other sets of topics communicated in Group 2 through the forums mostly focused on information seeking about technical, logical and interface aspects such as connection, output and input and, the evaluation criteria for each task (*Scorecard* topic).

4.3. Comparing the Groups

Table 4 presents the number of messages in each group according to the associated topics. Group 1 (G1) represents the explicitly recommended topics, including the extra-topics, i.e., those discussed in parallel. Group 2 (G2) represents the topics that received no recommendation to use the forum. In both groups the *Requirements* topic was the one most discussed (306 messages in total) in the forums during the tasks' development. The *Units* topic appeared in Group 1 only while the *Scorecard* topic appeared only in the tasks from Group 2. As discussed in Section 4.1, *Units* are about measures of a given variable while *Scorecard* (see Section 2) indicates some information about relevant scores for the competition's classification and discussion of evaluation criteria of results.

Table 4. Total of messages and topics from Group 1 and Group 2

Topics	# of msgs G1 (Recommended and Extra)	# of msgs G2 (Discussed)	# of msgs (G1 and G2)
Requirements	225	81	306
Access	210	10	220
Processing	99	24	123
Library	65	37	102
Input	60	1	61
Deadline	58	26	84
Style	47	6	53
Output	45	9	54
Connection	15	13	28
Units	4	0	4
Scorecard	0	5	5
Total	**828**	**212**	**1040**

Our empirical findings indicate that crowd members and copilots engage in both groups of messages to discuss and validate information on different task-related topics. Such information is predominantly associated with task requirements specification and this topic is the most discussed in the forums of both groups (see Table 4). Again, the messages exchanged about requirements evidenced that the crowd members need additional details to clarify the task requirements by interpreting the documentation, discussing with crowd members to find a common understanding about them, and/or validate additional information about the task with copilots. In G1, the following most discussed topics were *Access*, *Processing*, *Library*, and *Input*. Meanwhile, in the forums from G2, the topics frequently discussed were *Library, Deadline, Processing*, and *Connection*. In other words, the common most discussed topics between G1 and G2 are *requirements* and *processing*, which relates to source code compilation and execution errors.

In particular, the quotes about the topic *Processing* illustrate how crowd members engage in the forums to seek for information. Below, two crowd members explicitly ask the copilot to contact the client to confirm the variations they found in their results. The copilot answers them to follow the spec documentation provided while they wait for the customer's feedback.

[Coder A] *"I completed the algorithm but got a very different result. For symmetry material, the B matrix should be all 0.0. But in the summary file, the B matrix is filled with many small values. Could you please ask the client about this? It should be all 0.0 for symmetry material by definition."*

[Coder B] "*I have no problem with the B matrix, but I see minor variations in the values between that in the book and what I have. [@co-pilot] how much variation is acceptable? I believe the variation is unavoidable with floating point arithmetic."*

[Co-pilot] "*I'm waiting for the customer to clarify but for now we'll just have to follow whatever info was provided in the spec to implement it, even if the output values are not the same."*

The quote above can also help explain why processing is one of the most common topics discussed in Groups 1 and 2. *Processing*, as mentioned before, is about source code compilation and execution errors, either in applications or in servers; therefore it is clearly associated with requirements as well.

5. Discussion

To some extent, it is not surprising that in Groups 1 and 2 the most discussed topic in the forums is *Requirements* since this aspect is necessary to understand the problem described in the task and build a solution for it. In fact, our qualitative analysis reinforces that communication among crowd members and between them and co-pilots may solve understanding problems and clarify issues about the requirements during the development of the task.

In short, our results corroborate previous results. For instance, Boudreau and colleagues [6] claim that online forums relieve ambiguity in the task specification. Other studies [1, 5, 7] suggest that task documentation in software crowdsourcing is problematic with missing, unclear, and insufficient instructions. The issue with problematic requirements is that it leads to misconceptions about the task and might also lead to a low number of solution submissions [7, 20]. One of the possible explanations for the problem of task documentation in software crowdsourcing is suggested by [1], who argues: "Organizations may be hesitant to provide too many details on a certain crowd sourced task (i.e., module or component), yet sufficient detail in the specification is necessary for crowd developers understand what the client is requesting".

Stol and colleagues [1] report that a common problem in CS software development is to overwhelm the crowd with unclear and insufficient instructions which might lead to few or a low number of submissions to a task, and also submissions with poor quality. Furthermore, too many constraints placed on the task might also inhibit innovation [1, 20]. This might explain why so many different topics were discussed in Groups 1 and 2: crowd members and copilots engaged in conversations about ten different topics. Even in tasks where there was an explicit recommendation to use the forums in the task documentation (G1), the crowd used these forums to discuss about *four other* different topics, and the messages about these topics were about 30% of the total messages.

In general, customers and platforms must ensure a balance between providing enough information about the requirements and avoid providing non-useful information. This balance is also a problem in collocated or distributed software development. For instance, the lack of documentation about requirements is a constant problem in software development. Problems with understanding code are also common in any development model. However, we argue that striking this balance is even more critical in SW CS projects because this model has 3 main unique characteristics: (i) the extremely distributed nature of work where each crowd member is in a different site, (ii) limited communication, and (iii) a scarce view of a client's project and goals. We briefly expand on each one of them below.

First, crowd members (developers) and customers are geographically distributed. Previous research within the global software engineering literature [17, 19] suggests that communication across geographical barriers might be problematic because of the lack of informal communication necessary to coordinate the work. Second, we argue that SW CS has limited communication because there is *no* direct communication between customers and crowd members [24]: crowd members need to gather information about the problem to be solved through the forums that are mediated by the copilot, i.e., forums are the only communication channel that the Topcoder platform offers to crowd members. And, finally, the crowd has only the information about a particular task; they do not have the overall picture about the larger product under development, as it usually happens in traditional software development. This limited view has already been reported as problematic [1] [7].

In general, our results suggest that task documentation needs to be accompanied by a communication channel to allow crowd members to interact with customer representatives (the copilots), and each other. Even for somewhat small, and atomic tasks with limited complexity, documentation is not enough to allow a crowd member to develop a software module or a piece of a larger software system. Crowd members do need to use communication channels to complement the requirements from the task documentation. As illustrated in the previous section, the engagement of the crowd in the forums was expressive when tasks recommended the forum usage, but still significant when tasks instructions did not explicitly recommend to use the forums.

SW CS platforms' need to include a communication channel raises the question of whether the asynchronous nature of the interactions via forums is, or is not, the best solution for software crowdsourcing. In general, feedback is delayed and interruptions or long pauses in communication often occur in forums [10]. In this sense, synchronous chat tools, like Slack, had shown great benefits as a tool to mediate the communication among software developers. Stray et al. [25] highlight the transparency and the improvements in coordination brought by Slack. The possibility of adding other services (like bots) to support the interaction is also a positive aspect of these environments [26]. Moving towards these tools may improve the communication and the awareness in SW CS context.

6. Conclusion

There are several studies discussing software crowdsourcing, but social and technical works that have been largely unexplored in the context of SW CS. This study presents an empirical study on 25 SW CS challenges where we analyzed their tasks documentation and messages posted on online forums associated with tasks.

The findings suggest that weak documentation is a challenge within the SW CS context. However, when the crowd engages in online forums, they can obtain important additional information about the tasks that are not available in the official task documentation. Most of the tasks analyzed recommend using forums and those tasks that do not recommend use forums still use the forums anyways. In addition, the tasks that recommend using forums use them for more reasons than originally planned. Furthermore, our results suggest there is a need for platforms that offer more communication alternatives to engage the crowd members and to support with copilots during the task execution. Our results, although might not be generalizable to other platforms nor to a larger dataset of tasks in Topcoder, suggest that all the parties need to discuss different aspects of the task in different levels.

We provide a stepping stone for further research into requirements communication and cooperation for improving SW CS development processes, practices, and tools. New insights into that communication via forum and how this communication channel extends task documentation in competitive SW CS challenges have been obtained in this study.

Future work might be investigated to text mining to (semi-)automatically identify topics (requirements, environment, testing cases, etc.) being discussed in the forums so that they are made easily available to other crowd members; identify discussions, alternatives and design decisions regarding the requirements and other aspects so that they can also be easily made available to other crowd members.

ACKNOWLEDGEMENTS

This research has been partially funded by the Brazilian National Council for Research and Development (CNPq), under research grants 420801/2016-2 and 311256/2018-0.

REFERENCES

[1] Stol, Klaas-Jan, and Brian Fitzgerald. Two's company, three's a crowd: a case study of crowdsourcing software development. In ICSE, pp. 187-198. 2014

[2] Ke Mao, Licia Capra, Mark Harman, and Yue Jia. A survey of the use of crowdsourcing in software engineering. Journal of Systems and Software 126 (2017): 57-84.

[3] Ye Yang, Muhammad Rezaul Karim, Razieh Saremi, and Guenther Ruhe. Who should take this task? Dynamic decision support for crowd workers. *In ESEM,* pp. 1-10. 2016.

[4] Ke Mao, Yang, Y.,Wang, Q., Jia, Y.; Harman, M. Developer Recommendation for Crowdsourced Software Development Tasks. In 9th SOSE, 2015, pp. 347-356

[5] Luis Vaz, Igor Steinmacher, and Sabrina Marczak. An empirical study on task documentation in software crowdsourcing on Topcoder. In 14th ICGSE, pp. 48-57, 2019.

[6] Kevin Boudreau, P. Gaulé, K. Group, C. Riedl, and A. Woolley. 2014. From Crowds to Collaborators: Initiating Effort Catalyzing Interactions Among Online Creative Workers. SSRN Electronic Journal (012014).

[7] Leticia Machado, A. Zanatta, S. Marczak, and R. Prikladnicki. 2017. The Good, the Bad and the Ugly: An Onboard Journey in Software Crowdsourcing Competitive Model. https://doi.org/10.1109/CSI-SE.2017.6

[8] Aniket Kittur, J. Nickerson, M. Bernstein, E. Gerber, A Shaw, John Zimmerman, Matt Lease, and John Horton. 2013. The Future of Crowd Work. ACM CSCW, 1301–1318.

[9] Jeff Howe. 2008. Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business (1 ed.). Crown Publishing Group, New York, NY, USA.

[10] Thomas LaToza and Andre van der Hoek. 2016. Crowdsourcing in Software Engineering: Models, Motivations, and Challenges. IEEE Software, 33 (012016),74–80.

[11] Klaas-Jan Stol, Bora Caglayan, and Brian Fitzgerald. 2017. Competition-Based Crowdsourcing Software Development: A Multi-Method Study from a Customer Perspective. IEEE Transactions on Software Engineering PP(112017).

[12] Razieh Saremi and Ye Yang. 2015. Dynamic Simulation of Software Workers and Task Completion. 17–23. https: //doi.org/10.1109/CSI- SE.2015.11

[13] Razieh Saremi, Ye Yang, Guenther Ruhe, and David Messinger. 2017. Leveraging Crowdsourcing For Team Elasticity: An Empirical Evaluation at Topcoder. ICSE 2017 SEIP (05 2017).

[14] Thomas LaToza,, Towne, W., Andre van der Hoek, and James Herbsleb, (2013). Crowd development. 6th CHASE, (pp. 85-88).

[15] Thomas LaToza, M. Chen, L. Jiang, M. Zhao, and A. van der Hoek. 2015. Borrowing from the Crowd: A Study of Recombination in Software Design Competitions. https://doi.org/10.1109/ICSE.2015.72

[16] Erran Carmel and Ritu Agarwal. Tactical approaches for alleviating distance in global software development. IEEE software 18, no. 2 (2001): 22-29

[17] James D. Herbsleb and Rebecca E. Grinter. Architectures, coordination, and distance: Conway's law and beyond. IEEE software 16, no. 5 (1999): 63-70.

[18] Igor Steinmacher, C. Treude, and M. A. Gerosa. Let me in: Guidelines for the successful onboarding of newcomers to open source projects. IEEE Software 36, no. 4 (2018): 41-49.

[19] Julia Kotlarsky and Ilan Oshri. Social ties, knowledge sharing and successful collaboration in globally distributed system development projects. EJIS 14, no. 1 (2005): 37-48.

[20] Tajedin, Hamed, and Dorit Nevo. Determinants of success in crowdsourcing software development. In Conference on Computers and people research, pp. 173-178. 2013.

[21] Topcoder. 2019. Topcoder. Retrieved May 2, 2019 from https://www.topcoder.com 21]

[22] Cleidson R. B. de Souza, L. Machado, and R. R. M. Melo. On Moderating Software Crowdsourcing Challenges. ACM on Human-Computer Interaction 4, no. GROUP (2020): 1-22.

[23] Alpana Dubey, K. Abhinav, S. Taneja, G. Virdi, A Dwarakanath, A Kass, and S Mani. 2016. Dynamics of Software Development Crowdsourcing.

[24] Alexandre Zanatta, I. Steinmacher, L. Machado, Cleidson. R. d. Souza and R. Prikladnicki, "Barriers Faced by Newcomers in Software Crowdsourcing Projects," IEEE Software, vol. 34, no. 2, pp. 37-43, Mar 2017.

[25] Viktoria Stray, Nils Brede Moe, Mehdi Noroozi. Slack me if you can!: using enterprise social networking tools in virtual agile teams. ICGSE 2019: 101-111

[26] Bin Lin, Alexey Zagalsky, Margaret-Anne D. Storey, Alexander Serebrenik. Why Developers Are Slacking Off: Understanding How Software Teams Use Slack. CSCW Companion 2016: 333-336

[27] Corbin, Juliet, and Anselm Strauss. Basics of qualitative research: Techniques and procedures for developing grounded theory. Sage publications, 2014.