Collaborative Behavior and Winning Challenges in Competitive Software Crowdsourcing

LETICIA S. MACHADO, Universidade Federal do Pará, Brazil
RICARDO RODRIGO M. MELO, Universidade Federal do Pará, Brazil
CLEIDSON R. B. DE SOUZA, Universidade Federal do Pará, Brazil
RAFAEL PRIKLADNICKI, Pontifícia Universidade Católica do Rio Grande do Sul

Software Crowdsourcing (SW CS) allows a requester to increase the speed of its software development efforts by submitting a task to be performed by the crowd. SW CS is usually structured around software platforms, which are used by crowd members to identify a task suited for them, gather information about this task, and finally, submit a solution for it. In competitive software crowdsourcing, members of the crowd independently create solutions while competing against each other by monetary rewards for task completion. While competition usually reduces collaboration, in this paper, we investigated how crowd members create a collaborative behavior during programming challenges using online forums to help each other, share useful information, and discuss important documents and artifacts. We also investigated different collaborative behaviours by crowd members and and how this collaboration is associated with crowd members' improved outcome in the challenges. These results are based on analysis of the online forums from Topcoder, one of the largest competitive SW CS platforms.

CCS Concepts: • **Human-centered computing** → *Human computer interaction (HCI)*; *Empirical studies in HCI*; **Empirical studies in collaborative and social computing**.

Additional Key Words and Phrases: Software Crowdsourcing, Topcoder, Collaboration, Communication

ACM Reference Format:
Leticia S. Machado, Ricardo Rodrigo M. Melo, Cleidson R. B. de Souza, and Rafael Prikladnicki. 2021. Collaborative Behavior and Winning Challenges in Competitive Software Crowdsourcing. *Proc. ACM Hum.-Comput. Interact.* 5, GROUP, Article 220 (July 2021), 25 pages. https://doi.org/10.1145/3463932

1 INTRODUCTION

Crowdsourcing (CS) is defined as the act of an organization to make its work available to an undefined, potentially large network of people – a crowd using an open call for participation [21]. CS has been adopted for several purposes such as innovative design [24],[25], data science [46], and software development [26],[53].

Software Crowdsourcing, or simply SW CS, is an instantiation of crowdsourcing in the context of software development. According to Stol and Fitzgerald [45], it is a particular way of designing and creating software through the engagement of a pool of online members who can be tapped

Authors' addresses: Leticia S. Machado, Universidade Federal do Pará, Brazil, leticia.smachado@gmail.com; Ricardo Rodrigo M. Melo, Universidade Federal do Pará, Brazil, ricardo.mello@gmail.com; Cleidson R. B. de Souza, Universidade Federal do Pará, Brazil, cleidson.desouza@acm.org; Rafael Prikladnicki, Pontifícia Universidade Católica do Rio Grande do Sul, rafael.prikladnicki@pucrs.br.

2573-0142/2021/7-ART220 $15.00
https://doi.org/10.1145/3463932

on-demand to contribute to various types of software development tasks (e.g., requirements, design, coding, testing, evaluation, and maintenance).

The first studies about software crowdsourcing were published by Archak [2], Lakhani et al.[26], Begel et al.[3], and LaToza et al[29]. After these studies, this topic has raised interest of different researchers. For instance, Stol and Fitzgerald [45] describe a series of issues in software crowdsourcing projects, including communication and coordination, and quality assurance. Other authors have built prototype tools to study software crowdsourcing. LaToza et al.[29], for example, developed an application called CrowdCode for decomposing programming work into micro-tasks that can be implemented by crowd members.

Similarly to traditional crowdsourcing, SW CS is usually structured around software platforms. These are marketplaces allow requesters to seek crowd members to perform their tasks and, at the same time, support crowd members in finding tasks to work on. Examples of SW CS platforms include TopCoder [48], uTest, and Passbrain. In general, these platforms explore a competitive approach in which crowd members independently create solutions competing among them motivated by a monetary reward after completing a task [28]. At the end of the process, the requester validates the submitted solutions and rewards crowd members based on their solutions. Oftentimes, just one crowd member wins the task and gets the monetary reward.

The CSCW community has long recognized that competition reduces collaboration [38]. Despite that, recent studies [36] [4] [16] have indicated that there is collaboration in competitive crowdsourcing, i.e., the work conducted by a crowd is not as independent, autonomous, and isolated as it was assumed to be [16]. This is important since collaboration can improve the quality and quantity of crowdsourced task submissions [27] [54] [46]. In summary, while the competitive aspects of CS platforms have been studied in detail [1], our knowledge about collaboration in competitive crowdsourcing is still limited. In particular, to the best of our knowledge, collaboration in competitive software crowdsourcing has only been studied in controlled settings [36] [5][27]. Therefore, the goal of this paper is to investigate collaborative aspects of crowd members in competitive SW CS projects. This is done by answering the following research questions:

- **RQ1** Do software crowd members collaborate in natural settings?
- **RQ2** Do different types of crowd members engage in similar collaborative behaviour?
- **RQ3** Does collaboration among crowd members impact their individual performance?

To answer these research questions this paper describes an empirical study aimed to understand the collaboration among crowd members in the Topcoder platform. In this paper we adopt an "expansive definition" of collaboration (similar to Gray's [16]), one where crowd members do *not* work towards a shared goal (e.g., managing trains [19], developing software [17], designing and constructing a large building [9], managing a crisis [31]), but instead help each other, share information, and discuss documents. Furthermore, crowd members are all involved in the same, although temporary, online community, which Tausczik and Wang [46] call community-wide collaboration.

From a methodological point of view, we adopted a mixed-methods design [12] where we studied collaboration through the messages exchanged in Topcoder's forums. These forums are created for each specific task to support the communication of crowd members, among them and a customer representative, called co-pilot. They "enable participants to ask questions and discuss the requirements with the architects, clients and each other, to add additional detail or relieve ambiguity in the contest specification" [36]. We analyzed 25 forums associated with challenges from the 2 busiest months on TopCoder [15] in a total of 1,053 messages. We give particular focus on the analysis of 557 messages sent by crowd members only, i.e., excluding the co-pilot. Our analysis included the types of messages sent, who sent these messages, the number of messages they sent and, also the patterns of communication among crowd members.

In summary, this paper has three main contributions. First, it describes the collaborative activities of crowd members involved in competitive software crowdsourcing in a non-controlled, natural setting (RQ1). Second, it provides an initial explanation based on an analysis of the content of the messages that different crowd members [54] use the forums to collaborate about different subjects (RQ2). And, finally, it identifies an initial association between collaboration and individual performance in competitive software development (RQ3), which, until now, has only been identified in non-competitive scenarios [8].

The rest of this paper is organized as follows. Section 2 introduces the concepts of crowdsourcing and SW crowdsourcing and reviews related literature. Section 3 describes the methodology. The following sections, 4 and 5, present the results and the discussion. In Section 6, we present the design implications of our results, as well as, the limitations of our work in Section 7. Finally, section 8 describes our conclusions.

2 BACKGROUND

In a seminal paper Orlikowski[38] suggested that rewards need to be aligned with people's individuals goals so that collaboration can take place. Her conclusion was based on a field study in a company implementing a collaborative tool: while collaboration was expected to happen among the employees, organizational rewards that incentivized competition did *not* encourage collaboration. After that, research on collaboration in the context of competitions has not been explored by CSCW researchers with a few exceptions [10]. With the rise of crowdsourcing, and specifically competitive crowdsourcing where only one winner gets a financial reward, this has changed. In this section, we will define the concepts of crowdsourcing and present previous studies on collaboration and competition in crowdsourcing.

2.1 Crowdsourcing

In 2006, Howe [21] coined the term crowdsourcing (CS) when discussing how businesses were using the Internet to outsource work to many individuals. Crowdsourcing is basically a distributed problem-solving model [6]. A widely accepted definition argues that crowdsourcing is the act of an organization to outsource its work to an undefined networked labor using an open call for participation [21]. Open calls are divided in atomized tasks that can be completed and paid for in small increments [45] [23].

In general, there are 3 key components in a CS project: (i) the *requester* who creates a task and submits a request describing its main requirements, including instructions, constraints, acceptance criteria, and goals. The requester also defines the task's duration and the target audience, taking into account the abilities of the crowd. This results in a requirements document that goes through the platform; (ii) the *platform* makes the task requirements available to the crowd, assigns the task to the crowd, and intermediates the communication between two parties; and, finally, (iii) the *crowd members* who perform the requested task according to the requirements.

The advantage of using global and heterogeneous resources by assigning a task to the public rather than passing it on to a single company seems to be indisputable to most authors who study crowdsourcing [14] [43] [25] [23] [42]. Additional advantages include: cost reduction [33], faster time to market [45], large scale work [23], creative and open innovation [6], and higher quality [2].

Meanwhile, critics to crowdsourcing [25] [14] argue that the per-task payment structure used tend to abuse of the globally distributed crowd members since there is an extremely low payment for their labor, i.e., CS might be a form of labor exploitation. This is worse in the *competitive* CS model where usually only the author of the winning solution gets paid [25]. Crowdsourcing may also displace current crowd members and has the potential to replace some forms of skilled labor with unskilled labor as tasks are decomposed into smaller pieces recursively [25].

2.2 Software Crowdsourcing

Software engineering also seeks to gain the benefits of open collaboration from crowdsourcing [25] [36] [39]. Then, software crowdsourcing (SW CS) is defined as the act of engaging a global set of online crowd members, who provide software solutions or services on demand [3] [45].

Studying SW CS is particularly important because software development tasks are "often interdependent, complex, heterogeneous, and can require significant periods of time, cognitive effort and various types of expertise" [25]. This is clearly different from typical micro-tasks used in most CS platforms "that are characterized as self-contained, simple, repetitive, short, requiring little time, cognitive effort and specialized skills" [45]. Examples of platforms for SW CS include TopCoder, Upwork, 99design, uTest, and Passbrain [26] [56] [34].

SW CS usually adopts a *competitive* model in which the requester prescribes the work to be done by the crowd and later receives the different solutions proposed by the crowd members who worked individually competing against each other. Competitive SW CS is often anchored in a monetary reward based on task completion in which only the first ranking competitor receives the monetary rewards. The competitive approach to SW CS receives much attention in software development because one of the pioneers platforms, Topcoder, adopts this model. Other SW CS platforms – Upwork, uTest, and Passbrain – also explore a competitive approach[1].

2.3 Collaboration in Competitive Crowdsourcing

In competitive CS, crowd members become rivals and compete against each other. According to Hutter and colleagues [22], the level of collaboration and mutual support drastically decreases: crowd members are concerned about their solutions being stolen and, reluctant to exchange information, and share knowledge. However, Hutter et al. also observed how competition and collaboration, as extreme poles of individual behavior, might occur at the same time. They show that people not only compete in the challenges to win the prize, but also to socially interact and collaborate with other users, e.g., by commenting, giving feedback, and exchanging ideas. In fact, they found a positive correlation between crowd members competitive *and* collaborative behavior and contest outcome, i.e., there are quality improvements in the submitted ideas [2].

Gray and colleagues [16] describe a large study of Amazon Mechanical Turk (AMT) to explore whether crowdworkers collaborate and, if so, why, how and on what. They adopted an "expansive definition of collaboration" and show how crowd members collaborate to: (i) manage the administrative overhead associated with crowdsourcing tasks (ii) find lucrative tasks and reputable employees and (iii) recreate social connections and support often associated with brick and mortar-work environments. These connections allow crowd members to socialize as well as help each other complete the actual work itself. These 3 forms of collaboration took place *off-platform* since AMT does not provide any feature to support collaboration among crowd workers.

Finally, Tausczik and Wang [46], studied collaboration in competitive crowdsourcing on the Kaggle platform, a platform that allows either individual or team participation. Kaggle also allows participants to openly share their solutions with the community during a contest. Tausczik and Wang [46] suggest that 10% of the studied users shared solutions during the contests, and "that

[1]The competitive model is not unique to SW CS. Several non-software crowdsourcing platforms also adopt a competitive approach including 99design, Innocentive, Kaggle, OpenIdeo, and Threadless.
[2]The contest Hutter and colleagues [22] studied was conducted in two phases, and the crowd chose the best solutions of the 1st phase. This scenario hardly happens in SW CS: customers are the ones who select the winning solution.

participants doing medium well in the contest were the most likely to share code, and that sharing code improved individual, but not collective performance".

2.4 Collaboration in Competitive Software Crowdsourcing

In SW CS, only a few studies have focused on collaboration and competitive behavior. Specifically, Nag et. al[36], Boudreau and Lakhani [5], and LaToza et al. [27] all suggest that when crowd members collaborate, this has the potential to improve their comprehension about the task goals and, consequently, the quality of their solutions.

Nag et. al [36] demonstrate the successful applicability of CS for spaceflight software through an open tournament in the Topcoder platform. In this study, collaboration through online forum discussions was mandatory to enable participants ask questions about requirements with each other, add additional detail or reduce ambiguity in the contest specification. In addition to forums, external collaborative tools were used including source control tools [18], wikis, etc.

LaToza et al. [27] conducted a study where competing designers were given access to the designs of others and encouraged to use them to revise their own designs. Two separate but parallel competitions were conducted: one for software architecture design, and one for user experience design. The competitions were conducted entirely through electronic communication (email and Dropbox), i.e., without support of a CS platform. Their findings demonstrate the benefits of sharing solutions in software design to enable participants to borrow ideas and improve their designs.

Finally, Boudreau and Lakhani [5] conducted a field experiment in Topcoder designed to compare two different situations: (i) intermediate solutions were made available for inspection and reuse by other crowd members, and (ii) intermediate solutions were not disclosed. The results suggest that intermediate disclosure of solutions led to 70% fewer solution submissions because choices of solutions approaches were less independent, i.e., intermediate disclosure led to convergence of solutions with 30% fewer solution approaches tried. On the other hand, intermediate disclosure of solutions resulted in efficient reuse, lower costs, and higher performance. Final disclosure of solutions reduced "groupthink" resulting in independent experimentation with diverse approaches However, Boudreau and Lakhani[5] warn: "this led to considerable effort devoted to sub-optimal approaches and overall lesser learning and performance achieved". These results suggest that different problems and domains might need different approaches: contexts more adequate for wide experimentation might use final disclosure of solutions, while contexts that require higher performance might adopt intermediate disclosure of solutions.

2.5 Comparison with Previous Work

Previous work about collaboration in competitive (software) crowdsourcing draws attention to two aspects. First, most papers conducted studies in controlled, non-realistic scenarios. For instance, Hutter et al. [22] and LaToza et al. [27] used their own digital solutions ("platforms") to support the collaboration among crowd members. Meanwhile, other authors such as Nag et al. [36], Stol and Fitzgerald [45], Yang et al. [54] adopted the Topcoder platform. However, they either enforced collaboration and combined Topcoder with external tools or made changes to the platform to include additional collaborative features: Boudreau and Lakhani [5], for instance, reported that they had to work "closely with Topcoder executives and technologists to modify the platform".

Second, only two papers studied collaboration in competitive CS in *realistic* scenarios , but none of them studied *software* crowdsourcing. Gray et al. [16] studied AMT, which does not provide collaborative features. Meanwhile, Tausczik and Wang [46] studied the Kaggle platform, which has the ability to share solutions at the team and community levels. We argue that AMT and Kaggle can be thought of as extremes of a collaborative-support continuum: AMT provides no support at all, while Kaggle provides the ability to share solutions. In this paper, we study Topcoder, which is located in an intermediate position along this continuum since it only supports online forums for communication among crowd members. This makes Topcoder an interesting platform to study collaboration in competitive SW CS.

In summary, our work differs from these previous studies in two ways. First, it describes an empirical study of software crowdsourcing in a *natural* scenario instead of an experimental one. And, second, this scenario is based on a platform located in an intermediate position along the collaboration support continuum. This platform is described in the next section.

2.6 The Topcoder platform

Topcoder is one of the main platforms for competitive SW CS worldwide [2] [26]. It covers all phases of the software development lifecycle. Each development task is organized as a challenge through open competitions. Topcoder allows crowd members to choose to engage in software development tasks based on their personal skills, experience, and interests [40] [54].

The Topcoder platform has an online community of 1.2 million members from over 190 countries [48]. The perception of a large crowd with necessary expertise to develop solutions brings implications for requesters who expect faster time-to-market and high quality [44]. There is also a negative effect on the crowd's interest in competitions: a large number of crowd members register for tasks, but do not submit solutions [45] [55] [54] [41].

In this platform, there is a person who plays an important role in each challenge, (s)he is called co-pilot. The co-pilot is a customer representative whose role is to help crowd members: through the online forums, they provide guidance and support for the crowd members removing misconceptions, getting the registered members aware of deadlines, clarifying requirements with the requesters, and so on [13]. Co-pilots are experienced Topcoder's community members who manage the technical aspects of crafting and running competitions through to successful delivery [45]. In some challenges, crowd members are required to use the forums: for instance, additional information about the tasks is made available to the crowd members solely through the forums [49].

3 METHODOLOGY

To answer our research questions we adopted a mixed-methods approach combining both quantitative and qualitative data. Specifically, we used an embedded design [12] focusing on the messages exchanged in Topcoder's online forums and the crowd members who authored these messages. Topcoder online forums are created for each specific challenge to support the communication among the challenge participants: crowd members and the co-pilot(s). We collected and analyzed a total of 557 messages sent by crowd members in 25 forums associated with challenges from the 2 busiest months on TopCoder [15].

The qualitative analysis of the data allowed us to classify the messages sent by the crowd members in different subjects associated with the challenges. This information was combined with a classification of each crowd member based on his/her performance in the challenge or on the number of messages (s)he sent in the forums. Then, we conducted a quantitative analysis, using chi-square tests, to compare the collaborative behavior of crowd members according to their performance. After that, a new qualitative analysis assessed in more details the messages sent by crowd members. The following sections will details our approaches for data collection and analysis.

3.1 Data collection

3.1.1 Criteria for selecting Coding Challenges. From several development challenges, or tasks, hosted on Topcoder, we initially selected a group of 25 challenges for analysis. Our selection was based on the criteria described on Table 1.

Table 1. Summary of Challenge Selection Criteria

Category	Metrics	Description
(i) Active period of analysis	Between July and August 2017	Tasks available and open for registration in the period.
(ii) Number of registrants	15	Number of crowd workers who applied for a task.
(iii) Financial reward	$500	Task budge. The value the task requester is willing to pay
(iv) Task phase	Registration and submission	Indicates the task status (registration, submission, review)
(v) Task challenge	Development Challenges	Indicates the challenge area (design, development, and data science)
(vi) Task category	Code	Captures the different task categories (conceptualization, design, coding, etc.)
Total	25 Challenges	Total of challenges during the period (July - 9 tasks, and August - 16 tasks)

The first criterion was based on Dubey's et al [15] study, in which July and August were reported as the two months with the highest number of tasks posted on Topcoder. We have also restricted our sample to tasks with at least fifteen registrants in the challenges and with a minimum financial reward of USD 500. Criteria (ii) and (iii) were used to make sure the analyzed challenges attracted developers and, therefore, had discussions in their forums.

Topcoder has different categories and subcategories for participation in SW CS challenges, including algorithm marathons, design, data science, and development. In this study, we focused on "Development" challenges (item v) and the "Code" task category (item vi) during the registration and submission phases (item iv). This allowed us to focus specifically on programming challenges, arguably, the most important ones in TopCoder.

3.1.2 Selected Coding Challenges. After applying our criteria, we selected 25 coding challenges. The list of analyzed challenges is presented on Table 2. We refer to the 9 challenges that took place in July 2017 as DJ1 to DJ9, while the 16 challenges from August 2017 as DA1 to DA16.

We colleted 1,053 messages from these 25 challenges: 496 sent by co-pilots and messages sent by crowd members. These messages were sent by 120 different people: 11 co-pilots and 109 crowd members. We restricted our analysis to messages sent by crowd members since we wanted to understand how they collaborate.

Table 2 shows that a high number of crowd members register to have access to the information about the challenge [3]. However, a significant smaller number of crowd members do send messages in the online forums. At the end of the challenge, a smaller number of crowd members submit solutions. In fact, in a few cases (challenges DJ2, DJ3 and DA3), only one solution was submitted. This will be discussed in the next section. In contrast, seven crowd members submitted their solutions to challenge DJ6. This result is consistent with previous studies indicating a high number of quitters in SW CS platforms [54]. Finally, it is possible to notice that for all 25 challenges, a first and second prizes were awarded, and in three cases (DJ6, DJ9, and DA11) a third place prize was also awarded.

[3]Including the first author who had to register to have access to the forums.

Table 2. Message and crowd members information about the challenges

Challenge	Members in the forums	# of submitted Solutions	# of Messages	C1	C2	C3	W1	W2	W3
DJ1	5	5	102	21^{*}	16^{**}	16^{*}	16	10	-
DJ2	5	1	21	4	3	3	1	-	-
DJ3	14	1	60	15	6^{**}	4	6	-	-
DJ4	3	2	18	6	3^{**}	2^{**}	2	3	-
DJ5	4	3	43	7	7^{**}	6^{*}	0	7	-
DJ6	8	7	23	11^{*}	7	4	0	0	0
DJ7	3	2	8	1^{**}	1^{**}	1^{*}	1	1	-
DJ8	13	4	76	7^{**}	7	5^{*}	7	2	-
DJ9	3	3	19	4^{**}	2^{**}	2	2	4	0
DA1	6	4	35	10	5^{**}	1^{*}	0	5	-
DA2	6	3	115	41^{**}	9^{*}	4^{**}	41	4	-
DA3	7	1	71	12^{**}	11	3	12	-	-
DA4	3	2	27	10^{**}	2^{**}	1	2	10	-
DA5	2	2	7	2	1	-	0	0	-
DA6	7	3	47	10^{**}	5	3	10	0	-
DA7	5	4	32	8^{*}	4^{*}	3	1	-	-
DA8	3	3	22	5^{**}	4^{**}	4	4	5	-
DA9	4	4	29	10^{**}	4^{**}	-	10	-	-
DA10	6	3	22	8^{**}	1^{**}	1^{*}	8	1	-
DA11	7	3	36	7^{**}	5^{**}	3^{**}	3	5	7
DA12	0	2	2	-	-	-	0	0	-
DA13	43	3	122	21^{**}	5^{**}	4^{*}	21	5	-
DA14	2	2	35	12^{**}	7^{**}	-	7	12	-
DA15	13	2	53	15^{*}	2^{**}	1	2	-	-
DA16	6	3	28	5^{**}	4^{**}	1	5	4	-
TOTAL	178	72	1053	252	121	72	161	68	7
AVE.	7.12	2.88	42.12	10.05	5.04	3.42	6.44	3.77	2.33
MIN.	0	1	2	1	1	1	0	0	0
MAX.	43	7	122	41	16	16	41	12	7

To answer RQ3, our initial plan was to analyze all 25 challenges. However, given the fact that some challenges had a very low number of crowd members who sent messages in the forum (e.g., DA5 with 2 crowd members only), we restricted our analysis to forums where at least 5 different members posted in the forums. This resulted in the analysis of 15 challenges only for RQ3.

Finally, we developed a tool just to extract the data from the forums. Using this tool, we extracted: (i) date and time of each message, (ii) thread information, (iii) message sender, (iv) message recipient, and (v) the message itself. All the data was exported as a spreadsheet for later analysis.

3.2 Data Analysis

3.2.1 Content Analysis of the Messages. Initially, we read the content of each message and based on the interpretation of this content, we assigned codes to them [11]. Coding was conducted between December 2017 and February 2018. We decided to use codes that emerged from the analysis of the messages in the forums, i.e., we did not have any preconceived notion of the content of the messages nor used any theoretical framework during the coding process. For each forum's message the first and second authors read and coded it separately. Then, they reviewed together every single message and code to reach an agreement about the final categories and topics to be used. The other authors then reviewed the resulting coding scheme.

We used two types of codes: 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, a co-pilot was making a public announcement, crowd members were reporting problems, a crowd member was sharing a tip, etc. Definitions and examples of all categories are presented on Table 3. Meanwhile, topics were defined at a level of detail that could allow us to understand which subjects crowd member were communicating about during the challenges including, requirements, libraries, output, access, etc. Table 4 presents the definitions and examples of all topics.

Using categories and topics, we could identify the purpose of each message. For instance, that it was reporting a problem in the requirements document, or that it was an invitation request to gain access to a repository. In total, 10 categories and 11 topics emerged from the analysis of the messages, and each message was classified according to a topic and category.

Table 3. Definition and Examples of Categories

Category	Definition	Example
Public Announcement	Informative aspect about the task addressed to all registered competitors	"[...] The purpose of this document should be to train the ML system to identify data in the schedule."
Tips	Spontaneous suggestion to solve a certain problem	"But just a heads up, the numerical problems aren't free from errors, you better have a calculator nearby"
Help Request and Answer	Request for someone to help the crowd member to deal with something	"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?"
Confirmation Request and Response	Question to check and ensure the understanding of something related to a given topic	"It should cover entities and their fields. Also it should cover restrictions and data types for fields, right?"
Invitation Request and Response	Request to be invited to have access to a resource: a certain file, a repository, or a private key	"add me to [...] doesn't work now – [...] gitlab handle: [...] please Invite me to [...] project username: [...] email: [...]"
Identified and Response Problem	Informative message that characterizes a problem or mistake associate with the task	"[...] I'm experiencing the same problem, and I don't remember having that problem when I ran the code then. Don't really know what it could be though..."

Table 4. Definition and Examples of Topics

Topic	Definition	Example
i - Access	Access to code repositories, platform, data, or private key	"Is it possible to get access to the ios repository? trying to figure out the authorization workflow"
ii - Library	Technical aspect of libraries, frameworks, APIs, code components, plug-ins and development tools	"Do you really need D3JS here? Why don't you want just render svg elements using the standard ReactJS?"
iii - Connection	Issues that involved login/password failure	"Can you provide sample user name and password for connect app, since it is redirect to [. . .]"
iv - Deadline	The established deadline for the delivery of solutions	"Is it possible to provide an additional extension of just 24h? I am definitely going to submit this"
v - Inputs	Input variables and parameters of the tasks	"Yes, we've been discussing the metal nu parameter, so if it's not used in the metal calculation, it can be an optional parameter."
vi - Style	Formatting styles details related to the solution user interface / frontend	"Try to wrap the content of _reset.scss into :global wrapper, cause wherever you import styles [. . .]"
vii - Processing	Compilation of code or execution errors in the solution and/or associated servers	"I'm having some trouble with my android studio and i am not able to build the app.Will work at it. Thanks for the help."
viii - Requirements	Technical and functional requirements specification. Might include different artifacts	"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.)"
ix - Output	Output variables of the tasks and solutions	"To clarify, do you mean that we should output a txt file containing data in the format of the sample file?"
x - Scorecard	Relevant scores for the classification of the solution	"I appreciate that there's a custom scorecard for this challenge [. . .]"
xi - Units	Unit measures of a given input/output variable	"It seems that they are considering a ton = 1000 Kg, i.e. ton = tonne [...]"

3.2.2 Classifying Crowd Members. In addition to coding the messages sent by crowd members, we also classified the crowd members in different ways. Table 2 presents this information. It indicates that the forums are somewhat active with high number of messages, but there is a small number of crowd members who actually send messages. To be able to capture this information we decided to label crowd members as *Cn* where *n* indicates the total number of messages sent by the member relative to other members in the challenge. That is, C1 is the member who sent the largest number of messages in the forum, C2 is the member who sent the 2nd largest number of messages, and so on.

Table 2 also indicates that a large number of crowd members had registered for tasks, but did *not* submit solutions. This has been reported by other authors before [45] [55] [54] [41]. In fact, to reflect the different levels of crowd members' participation, Yang and colleagues [54] propose a three-level crowd members classification: Quitter, Submitter, and Winner. Quitters are crowd members who did not submit a solution to a task. Submitters are crowd members who have submitted a solution, but did not win the competition. Finally, Winners are the crowd members who actually won a competition. In short, we classified all crowd members according to the outcome of their participation in the challenges. This information was available to all registered members when the challenge ended. After that, we classified crowd members as *Wn*, where *n* is associated with the final ranking of the member in the challenge: W1 is the member who won the 1st prize reward, W2 is the member who won the 2nd prize, and so on. Note that all analyzed challenges have a W1 member, but not all have W2 or W3 members.

Finally, Table 2 presents information about the classification we used: the number of messages sent by C1, C2, and C3; and W1, W2, and W3 in each challenge. Cells marked with (*) represent crowd members who are submitters, while cells marked with (**) indicate members who are winners. For instance, in the challenge DJ8 the crowd member who sent more messages (C1) was also the winner of this challenge (7**), while C3 also submitted a solution (5*) and C2 did not submit a solution (7). Cells with (-) in columns W1, W2 or W3 indicate that there was no winner in this position even when solutions were submitted. This happened, for instance, when the requester did not approve the submitted solution. Cells with 0 (zero) in columns C1, C1 or C2, etc, indicate that the winner member did *not* send messages in the forum: there were 5 cases out of 25 challenges in which this happened.

Note that challenge DA12 is an outlier: no messages were exchanged by the crowd members, only the co-pilot used the forum. The two winning submissions (W1 and W2) for this challenge were from members who did not send messages on the forum.

4 EMPIRICAL RESULTS

4.1 Collaboration among software crowd members - RQ1

Our first research question is: "[RQ1] Do software crowd members collaborate in natural settings?" Our results indicate that in these settings, crowd members show a collaborative behavior by answering questions and helping each other to understand task instructions, sharing useful information about technical and operational issues, and by discussing task requirements, among other aspects.

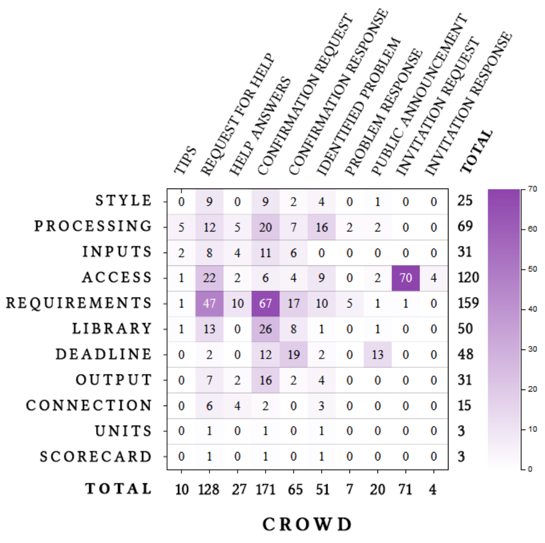
4.1.1 Messages exchanged by Crowd Members. According to Figure 1 the topic “Requirements” is the one most discussed by crowd members with 159 messages. Most of these messages were about the category “Confirmation Request" followed by "Request(s) for Help". Overall, these messages seem to indicate that there are problems (inconsistencies, ambiguities, missing information, etc) in the task documentation. This Figure also indicates that the category “Confirmation Request” has the largest number of messages (171). Again, this category represents a question to check and ensure the understanding of the crowd members on a given topic, mostly the Requirements of a task. However, there also are confirmations about other topics (Libraries, Processing, etc).

It is also important to note on Figure 1 the relatively high number of “Confirmation responses” (65): more than 1/3 of the “Confirmation Requests”. Again, it is important to remember that these are answers to the questions posted by crowd members, which illustrates how members collaborate using in the online forums.

In addition to “Confirmation Requests”, crowd members do also send several “Request for help” (128). By analyzing the topics of these messages, one can note that these requests are, mainly, to obtain help about the task’s requirements, something expected. Although in a significant smaller scale than requirements, requests for help are also about access to repositories, libraries used to create the solutions, and the processing that runs in the solutions among other topics. In short, one can notice 27 messages from other crowd members answering requests for help: a collaborative behavior observed in a natural setting despite the competition.

The third highest number of messages per category is associated with "Invitation Requests" (71). These requests have a small number of answers (4), which is expected since the large majority of them (70) are about accessing repositories and tools which is a co-pilot responsibility [13].

The next largest category is "Identified Problems" (51), which are about the following topics: the expected execution of the solution (Processing), Requirements and Access to repositories and tools. These are, again, aspects managed mostly by co-pilots which explains the small number (7) of Problem Responses posted by crowd members [13]. Another explanation for the large difference between the number of requests and answers is because, in some cases, several crowd members provide information that is addressed by a single post from the co-pilot. In fact, we analyzed the messages sent by co-pilots and confirmed that most questions not answered by crowd members are answered by co-pilots [13]. However, this is out of the scope of this paper.



	TIPS	REQUEST FOR HELP	HELP ANSWERS	CONFIRMATION REQUEST	CONFIRMATION RESPONSE	IDENTIFIED PROBLEM	PROBLEM RESPONSE	PUBLIC ANNOUNCEMENT	INVITATION REQUEST	INVITATION RESPONSE	TOTAL
STYLE	0	9	0	9	2	4	0	1	0	0	25
PROCESSING	5	12	5	20	7	16	2	2	0	0	69
INPUTS	2	8	4	11	6	0	0	0	0	0	31
ACCESS	1	22	2	6	4	9	0	2	70	4	120
REQUIREMENTS	1	47	10	67	17	10	5	1	1	0	159
LIBRARY	1	13	0	26	8	1	0	1	0	0	50
DEADLINE	0	2	0	12	19	2	0	13	0	0	48
OUTPUT	0	7	2	16	2	4	0	0	0	0	31
CONNECTION	0	6	4	2	0	3	0	0	0	0	15
UNITS	0	1	0	1	0	1	0	0	0	0	3
SCORECARD	0	1	0	1	0	1	0	0	0	0	3
TOTAL	10	128	27	171	65	51	7	20	71	4	

Fig. 1. Classification of the messages sent by the crowd members

Finally, Figure 1 indicates that crowd members do send "Public Announcements" (20) and share "Tips" (10) with each other. In both cases, they are sharing useful information about the task that is beneficial to other members. While Public Announcements are usually about the Deadline, Tips range from Processing to Access.

4.1.2 Examples of Collaboration among Crowd Members. Figure 1 presented an overview of the collaboration that took place among crowd members. Below, we will present quotes that illustrate how crowd members help each other, share useful information, and discuss documents associated with the task.

Crowd members helping each other. The first example describes a message exchange between three crowd members in which member A sends a request for help about the requirements. Member B answers that request, and complements that by providing a tip to member A. Member C also answers the request and helps member A.

> **Member A:** "According to the spec, the value of Symmetry is only used for printing "SYM" in some files. Is there another use for 'Symmetry'? I think so. Besides this, as [co-pilot name] said for symmetric materials, [long description here] So I repeat my question: Is there another use for 'Symmetry'?"
> **Member B:** "I think we can safely ignore the sample files and use the pdf. I have validated my code against [short description here]. Hence i think we can ignore the symmetry property except for printing "SYM" as necessary."
> **Member B:** "But just a heads up, the numerical problems aren't free from errors, you better have a calculator nearby:)"
> **Member C:** "Yes. It's used to calculate" [very long description here].

Crowd members sharing useful information. During the data analysis we observed that some crowd members voluntarily shared information that could benefit other competing members. For instance, the following quote illustrates a situation in which the winner member (B) describes his/her trade-off analysis in response to a quitter's request (Member A).

> **Member A:** "I'm not sure about the pros/cons. Later, we'll use the response to serve the user a login page and then reprocess the request after authentication."
> **Member B:** "Pros: 1. This is a standard pattern. All the unauthorized calls has to be stopped at Gateway layer. Here only authentication/token validation only will be performed. [Other pros here] Cons: Not able to think of anything. Reg: Login page use case 1. If the caller is invoking webservice/REST service, [short description here] Hope it helps to make decision. Thanks"

The next example – from a different challenge – illustrates an additional message that we classified as Tip being shared by crowd members:

> **Member D:** Unless storage is not a problem, changing this to VARCHAR instead of CHARACTER will make use of less space.

Crowd members discussing documents. Below, we present an example when a crowd member (D) inquires the co-pilot about how to handle a specific part of the requirements document. However, a different crowd member (C) answers the question before the co-pilot.

> **Member D:** "[co-pilot username], could you tell me how i can verify the calculations of the engineering constants defined in section 9.3? The other pdf on laminate analysis doesn't seem to have anything related to this."
> **Member C:** "In this challenge, [a very detailed answer here] At last, you can calculate Clam and Laminate Properties".

In the following example a quitter (member A) and a winner (member B) engage in a long discussion about the document "Challenge Overview." This document is cited by the quitter in his/her replies to the winner. Eventually, the co-pilot joins the discussion to finish it by agreeing with the winner:

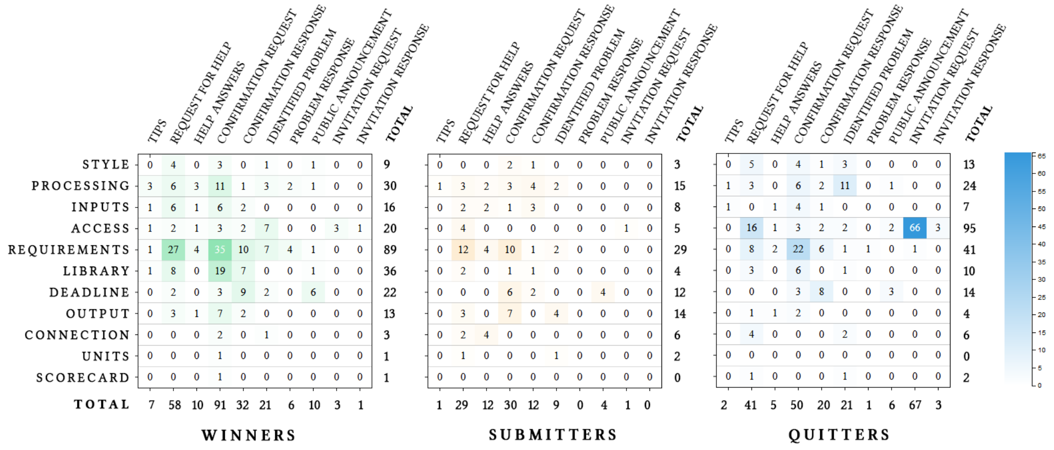
> **Member A:** "Is there a test server we can use for LDAP authentication? If not, how are we supposed to access the LDAP server ...?"

> **Member B:** "It seems you made a wrong question here, this challenge is about proxy, it is not that "LDAP Service" challenge."
> **Member A:** "I might have misinterpreted the scope, but I thought we should rewrite the headers based on LDAP information [description about documentation here].
> **Member B:** "That is just overall info about the whole project. In this challenge, I don't think we need to touch the LDAP."
> **Member A:** "From Challenge Overview: [description of challenge overview here] Could somebody please clarify what we should do here? e.g.: - access an LDAP server - mock the LDAP request - just add some hardcoded headers - or something else"
> **Co-pilot:** "[Winner username] is correct here. We are looking for a proxy POC, and it doesn't need to do anything related to LDAP for now. That will be integrated in a future challenge."

In summary, the analysis of Figure 1 and the previous quotes show that the communication among crowd members using the online forums allows them to collaborate when working in a challenge. They collaborate by helping each other, sharing information and discussing important documents. This happens despite the fact that they are all competing for the financial reward.

4.2 Difference in Collaborative Behaviors - RQ2

We answer our 2nd research question by describing the results of the analysis of what different members communicated about, i.e., the categories and topics of their messages. This is done using the winners, submitters, and quitters classification and is presented on Figure 2.



WINNERS

	TIPS	REQUEST FOR HELP	HELP ANSWERS	CONFIRMATION REQUEST	CONFIRMATION RESPONSE	IDENTIFIED PROBLEM	PROBLEM RESPONSE	PUBLIC ANNOUNCEMENT	INVITATION REQUEST	INVITATION RESPONSE	TOTAL
STYLE	0	4	0	3	0	1	0	1	0	0	9
PROCESSING	3	6	3	11	1	3	2	1	0	0	30
INPUTS	1	6	1	6	2	0	0	0	0	0	16
ACCESS	1	2	1	3	2	7	0	0	3	1	20
REQUIREMENTS	1	27	4	35	10	7	4	1	0	0	89
LIBRARY	1	8	0	19	7	0	0	1	0	0	36
DEADLINE	0	2	0	3	9	2	0	6	0	0	22
OUTPUT	0	3	1	7	2	0	0	0	0	0	13
CONNECTION	0	0	0	2	0	1	0	0	0	0	3
UNITS	0	0	0	1	0	0	0	0	0	0	1
SCORECARD	0	0	0	1	0	0	0	0	0	0	1
TOTAL	7	58	10	91	32	21	6	10	3	1	

SUBMITTERS

	TIPS	REQUEST FOR HELP	HELP ANSWERS	CONFIRMATION REQUEST	CONFIRMATION RESPONSE	IDENTIFIED PROBLEM	PROBLEM RESPONSE	PUBLIC ANNOUNCEMENT	INVITATION REQUEST	INVITATION RESPONSE	TOTAL
STYLE	0	0	0	2	1	0	0	0	0	0	3
PROCESSING	1	3	2	3	4	2	0	0	0	0	15
INPUTS	0	2	2	1	3	0	0	0	0	0	8
ACCESS	0	4	0	0	0	0	0	0	1	0	5
REQUIREMENTS	0	12	4	10	1	2	0	0	0	0	29
LIBRARY	0	2	0	1	1	0	0	0	0	0	4
DEADLINE	0	0	0	6	2	0	0	4	0	0	12
OUTPUT	0	3	0	7	0	4	0	0	0	0	14
CONNECTION	0	2	4	0	0	0	0	0	0	0	6
UNITS	0	1	0	0	0	1	0	0	0	0	2
SCORECARD	0	0	0	0	0	0	0	0	0	0	0
TOTAL	1	29	12	30	12	9	0	4	1	0	

QUITTERS

	TIPS	REQUEST FOR HELP	HELP ANSWERS	CONFIRMATION REQUEST	CONFIRMATION RESPONSE	IDENTIFIED PROBLEM	PROBLEM RESPONSE	PUBLIC ANNOUNCEMENT	INVITATION REQUEST	INVITATION RESPONSE	TOTAL
STYLE	0	5	0	4	1	3	0	0	0	0	13
PROCESSING	1	3	0	6	2	11	0	1	0	0	24
INPUTS	1	0	1	4	1	0	0	0	0	0	7
ACCESS	0	16	1	3	2	2	0	2	66	3	95
REQUIREMENTS	0	8	2	22	6	1	1	0	1	0	41
LIBRARY	0	3	0	6	0	1	0	0	0	0	10
DEADLINE	0	0	0	3	8	0	0	3	0	0	14
OUTPUT	0	1	1	2	0	0	0	0	0	0	4
CONNECTION	0	4	0	0	0	2	0	0	0	0	6
UNITS	0	0	0	0	0	0	0	0	0	0	0
SCORECARD	0	1	0	0	0	1	0	0	0	0	2
TOTAL	2	41	5	50	20	21	1	6	67	3	

Fig. 2. Classification of the messages sent by the Winner, Submitters and Quitters

4.2.1 *Different Types of Crowd Members Discuss Different Things.*

Figure 2 shows that both winners and submitters discuss mainly topics associated with Requirements (Confirmation Requests, and Requests for Help). Specifically, Requirements are about 37% of winners' and 29% of the submitters' messages. Winners also discuss the Libraries (15%) to be used in their solutions; while submitters also discuss the Output (15%) of their solutions – closely followed by the Processing (14%) implemented in their solutions. Meanwhile, quitters mainly send messages associated to Access (43%) to repositories, sites, tools, etc and, then, about Requirements (19%). One should also note that the number of messages about Access from Quitters is more than 2x higher than the number of messages about

Requirements. This seems to suggest that quitters face more difficulties in accessing the repositories and tools required to develop solutions to the challenges.

A chi-square test of independence showed that there was a significant relationship between the performance in the challenge (Winner, Submitter, or Quitter) and the topics discussed in the online forums: $\chi^2(10, N = 460) = 135.37, p < 0.00001$. In other words, Quitters are more likely than Submitters and Winners to send messages about Access and less likely to send messages about Requirements. Winners are more likely to send messages about Libraries than Submitters and Quitters. Finally, Submitters are more likely to send messages about Output than Winners and Quitters[4].

We also computed with whom the crowd members communicated: whether co-pilots or other members. The results are reported on Table 5. In general, winners do send more messages to other crowd members than to co-pilot(s), while submitters and quitters are the opposite, i.e., they send more messages to co-pilot(s). This Table also illustrates whether crowd members asked or answered questions, or sent other messages[5].

Table 5. Analysis of Messages

	Winners	Submitters	Quitters
Asked Question	70.8%	69.4%	51.8%
Answered Question	20.4%	24.5%	12%
Neither asked / answered	8.8%	6.1%	36.2%
Messages to crowd	55.4%	38%	39.3%
Messages to co-pilot	44.6%	62%	60.7%

A second chi-square test of independence for the data from Table 5 showed that there was a significant relationship between the performance in the challenge (Winner, Submitter, or Quitter) and their behavior regarding asking, answering questions or sending other messages: $\chi^2(4, N = 554) = 69.84, p < 0.00001$. That is, Winners and Submitters did ask and answer more questions than Quitters. Meanwhile, Quitters send more messages that could not be classified as neither questions nor answers. This result illustrates how distinct members exhibit different collaborative behaviour.

4.2.2 Examples of Messages according to Types of Crowd Members. In the previous section we could conclude that winners, submitters and quitters all discuss Requirements with different priorities. In second place, Winners discuss Libraries used to implement their solutions while Submitters discuss the Processing and the Output of their solutions. Meanwhile, Quitters discuss mainly Access (to repositories, tools, etc). In this section, we will present one or two, among several examples we identifed, of these messages.

Winners discuss Libraries. In the following quote there is a question addressed to, and answered by, the co-pilot. It is important to notice in this quote how the Winner provides technical a rationale (they are compatible) for using one library instead of the other:

> **Winner:** "can we use ObjectiveC frameworks for JWT generation? Note that Swift and Objective-C frameworks are api compatible: swift frameworks still have c headers

[4]We focused on the analysis of the following topics: Requirements, Access, Library, Processing, Input and Output.

[5]In this case, Asked questions are the following categories: Request for help, Confirmation request, and Identified problem. Answered Questions are composed by the following categories: Help answers, Confirmation response, and Problem response. Finally, Neither asked / answered are Tips, Public announcements, Invitation requests, and Invitation responses.

anyway, so there's no difference to the app, it's still a separate module and no bridging occurs
Co-pilot: "Yes, that's fine."

Submitters discuss about Processing and Output. The next example describes two Submitters discussing the expected output of the challenge solution. They are helped by the Winner of the challenge who reminds them that their proposed approach is not appropriate.

Submitter A: "If we see the specifications, we clearly see a space between MATID and E1 and between MATID and t; however [long description here] In conclusion I think there should be spaces and it doesn't mean having more than 72 characters per line."
Submitter B: "For the ABAQUS output file, the spec mentions [short description here] Since these are not mentioned at any other point in the spec, is it ok to go ahead [question about how to proceed here].
Winner: "Suppose you are writing a program reading this line. You know that each field will take 8 digits. What will you do? [proposion solution here] But we should not add spaces. This is definitely against the spec and the rules."

Quitters discussing Access. Finally, according to Figure 2, quitters mostly send Invitation and Confirmation Requests to have Access to shared documents, code repositories, etc. In this case, it is important to notice that the communication regarding the topic "Access" involves the co-pilot who is the person responsible for granting access to the repositories.

Quitter A: "Dear co-pilot, Please re-check to add my handle = [user] 404 not found now."
Co-Pilot: "See here – [link] You would have received an invite mail from Github. Please make sure to check your spam folder."
Quitter A: "Please resend invite for me. I also check on Spam and All Inbox folder. I see nothing. Sorry!"
Co-Pilot: "I added [user] again"
Co-Pilot: "Please find the API documentation attached. The login credentials are mentioned in the first page."
Quitter B: "Hi, is there a generic account available for accessing the API?"
Co-Pilot: "I responded on this thread with the login info – [link]"
Quitter C: "Can you provide the original public certificate [short description here]
Co-Pilot: "I will ask the client if they can provide it. But the Android implementation was done without requiring this so I'm not sure if it's really needed."
Quitter C "I just found that token cannot be obtained without trusted certificate [short description here] Can you provide it?"
Winner: "FYI 3 is incorrect assumption"
Co-Pilot: "[Winner username] is right here. I've responded on your other thread regarding the same as well."

4.2.3 *Different Types of Crowd Members Discuss Differently.*

A different point we observed during the qualitative analysis of our data is that not only the different types of crowd members discuss different things, but they do so in different ways. For instance, the quote from the previous section, in addition to illustrating Quitters discussing about Access, illustrates that Quitters often ask for information that was already made available to them. For instance, the conversation between the co-pilot and Quitter B. We observed similar situations in other challenges, but all of them associated with Quitters.

Another difference we observed between the messages from Winners / Submitters and Quitters is the following: Quitters ask very simple questions, while Winners and Submitters ask, and answer long and detailed questions. An example of a complex discussion between Submitters and Winners was presented previously when we discussed how Submitters discuss more often about Processing and Output. In contrast, we present a quote from a Quitter discussing the Output of his/her solution using very simple questions:

> **Quitter F:**" 'Re-Send Invitation Code' link also needs to be removed, correct? Also can we delete the code or only comment for later use?"
> **Co-pilot**:"Please keep it for now - no action needed on it"
> **Quitter F**: "So remove the text box , but keep the link correct ? Also the related source code needs to be removed or commented out?"
> **Co-pilot**: "Remove the text box and keep the link. Remove any code related to textbox."

Finally, below we present a quote from a Quitter discussing about a Library that (s)he would like to use to implement the required feature in the challenge. However, the co-pilot explains that (s)he needs to implement this feature as a new service on top of the current libraries.

> **Quitter**: "We want to add a new Outage service to the base libraries that, given an account number, returns the outage details, if any." Is there a particular library we have to use for this or is this already a service that exists in the codebase?
> **Co-pilot**: "You will implement the outage service as a new service in the base libraries, similar to CXP, Einstein, etc..."

In summary, our quantitative results present evidence that Winners, Submitters and Quitters discuss different topics in the online forums. In addition, Winners and Submitters ask and answer questions more often than Quitters. Meanwhile, our qualitative results suggest that they discuss these topics differently: while Winners and Submitters ask and answer detailed questions, the Quitters mostly ask and answer very simple ones.

4.3 Collaboration and Task Outcome - RQ3

Our third research question is about the relantioship between crowd members' collaborative behavior and their outcome in the challenges. Since Topcoder supports limited collaboration by providing online forums asssociated with each challenge (see Section 2.6), we modelled crowd members collaborative behavior as a function of the number of messages they sent in these forums: C1 is the crowd member who sent most messages, C2 is the member who sent the 2nd largest number of messages, and so on (see section 3.3.1). In other words, we explored whether crowd members who were winners of the first, second or third prizes (W1, W2, and W3) were also the ones that sent more messages in the forum (C1, C2 or C3).

4.3.1 The Collaborative Behavior of Winners. The results of our analysis are presented on Table 6. In 2 of the 15 (13.3%) analyzed challenges winners (W1) did not send a single message. Out of the 13 remaining challenges, we noticed that the winner (W1) is the crowd member who sent the largest number of messages (C1) in 8 cases (61,53%) . This Table also presents this analysis for "C1 or C2", "C1 or C2 or C3", and, finally, for "C4 to Cn". In only 2 out of 13 cases (15.38%) the challenge winner (W1) was *not* among the top 3 crowd members who sent more messages.

Out of the 15 challenges we analyzed, only 10 awarded the 2nd place (cell W2, Number of Winners on Table 6). Again, in 2 of these 10 challenges (20%), the winners (W1 OU W2) did not communicate at all in the forums. In 50% of the challenges, W2 was either C1 or C2, while in 60% of the challenges (6 out of 10), W2 is part of the top-3 communication group (C1, C2 or C3). In the other 2 challenges (20%), W2 was not part of the top-3 communication group.

Table 6. Communicating in the Forums and Winning Challenges

	# of Winners / Members	Zero Messages	C1	C1+C2	C1+C2+C3	C4 to Cn
W1	15	13.3%	61.53%	73.33%	80%	15.38%
W2	10*	20%	0%	50%	60%	20%
W3	2**	50%	50%	50%	50%	0%
Submitters	23	43,47%	17.39%	21.73%	47.82%	13.04%
Quitters	37	0%	10.81%	18.91%	29.72%	40.4%

*Five of the 15 analyzed challenges did not have a W2 winner.
**Thirteen of the 13 analyzed challenges did not have a W3 winner.
Note: Cell (W1, C1) indicates that the winner is also the crowd member who sent more messages in the forum. Cell (W1, C1+C2) indicates that the winner (W1) is either the member who sent the most (C1) or the 2nd most number of messages (C2). Accordingly, cell (W2, C1 + C2 + C3) indicates that the second prize winner (W2) is among the top-3 members who sent most messages.

The last observation about the winners is that only 2 out of the 15 analyzed challenges rewarded third places (W3). In this case, the winner of the 3rd place (W3) is the crowd member who sent more messages in the forum (C1) in 1 case (50%) and, in the other 1 case the winner did not sent messages at all in the forum.

4.3.2 The Collaborative Behavior of Submitters and Quitters. The second part of Table 6 describes an analysis for Submitters and Quitters similar to the one just described about Winners. The focus was to check whether more collaborative members (C1, C2, or C3) do or do not submit solutions, i.e., whether they are Submitters or Quitters.

Results indicate that in 13.04% of the challenges, submitters were part of the crowd members who did *not* send more messages (cell Submitters, C4...Cn). This number rises to 40.4% of the challenges when crowd members are Quitters (cell Quitters, C4...Cn). Table 6 also indicates that in 47.8% of the cases, the members who submitted their solutions to the challenges were among the top 3 members who sent more messages (cell submitters, C1 or C2 or C3). This value, 47.8%, is almost half of the same value for Winners (80%). For Quitters, this value is even lower 29.72% (cell quitters, C1 or C2 or C3). These results suggest that collaboration is associated with crowd members winning and submitting their solutions to the challenges in which they were registered and thus, competing for the best solution. Finally, one should note that the total of 37 Quitters all communicated in the forums (see cell Quitters, Zero Messages). This suggests that they seems to be interested in participating in the challenges, although they were not even able to submit a solution.

Overall, our analysis of the Winners suggests that, at least, 50% of them in any category (W1, W2, or W3) are either the crowd members who send the highest (C1) or the second highest (C2) number of messages in the forums. By contrast, the amount of Submitters and Quitters who send the highest number of messages (C1) is relatively small (17.39% and 10,81% respectively). When taking into account C2, these numbers raise a bit to 21,73% and 18,91% respectively. Note also that, for all rows (except "C4 to Cn") the Winner values are higher than Submitter values, which are higher than the Quitter values.

5 DISCUSSION

5.1 Answering our Research Questions

Previous work has revealed that crowdsourcing involves significant communication and collaboration among crowd members [25]. Examples range from volunteers writing articles together on Wikipedia [50] [7], collaborative translation on Mechanical Turk [24] and supporting disaster relief efforts [6]. Specifically, in *competitive* crowdsourcing, collaboration also exists [22] [5] [27] [16] [46]. However, these studies have either been conducted in controlled settings [22] [36] [27] [5] or focused on extremes of a collaborative-support continuum: [16],[46]. Our empirical observations complements these previous works by describing a study conducted in a *natural* setting and by focusing on a competitive software CS platform that is *not* at the extremes of this continuum. In the following paragraphs we answer our three research questions.

Our 1^{st} research question presents evidence that collaboration does occur in natural settings. We observed members asking and answering questions, sharing useful information, discussing documents, and so on. In some cases, crowd members even answered requests addressed to the co-pilots before them. This result is similar to LaToza's et al. [29] where UX designers, despite competing against each other, individually wished to see more opportunities for collaboration.

By answering our 2^{nd} research question we conclude that varied crowd members – winners, submitters, and quitters - communicate about different subjects. The overwhelming number of messages about Requirements suggests that the associated documents that describe a task are not enough to allow a crowd member to develop his/her solution. This result supports the concern in SW CS projects about problems on documentation [45] [29] [32]. Furthermore, it indicates the important role of co-pilots facilitating the communication in these challenges, which is something that has been reported by other authors [13].

Our 3^{rd} research question suggests an association between collaborative behavior and performance: winners were usually the ones who sent more messages. This means they seem to be more likely to collaborate even though they are in a competition. In contrast, Tausczik and Wang [46]) report that the members less likely to collaborate were the ones performing the very best. However, in the platform they studied, rankings were displayed publicly during the contests. Meanwhile, in Topcoder there are no public rankings for the challenges; there is only information about members' ranking in the platform overall. Therefore, it is possible that winners engage in collaborative behavior because they are not aware of how other crowd members are performing.

5.2 Revisiting Previous Work

Crowdsourcing studies in other domains (e.g., LED design [22] and predictive modeling [46]) observed that some individuals work solely competitively, while others work competitively and collaboratively. Tausczik and Wang [46] reported that about 10% of their subjects worked competitively and collaboratively. Our study identified a similar result: some crowd members did not engage in collaboration in the forums, while others were very collaborative. Based on the number of registered crowd members and the number of members who participated in the forums (see Table 2), we estimate that about 19.69% of the crowd members in the challenges we studied had a collaborative and competitive behavior. Our result is in line with Tausczik's and Wang's results.

Previous studies about collaboration in *competitive* crowdsourcing have reported mixed results regarding the benefits of collaboration in individual and community-level performance. Boudreau and Lakhani [5] suggest that collaboration early in the challenge increased collective performance, but reduced individual performance. Meanwhile, Tausczik and Wang [46] reported that "individuals performed better in the short-term and long-term when they collaborated, but that there was no association between collaboration and community level performance". In our study, we did not take into account quantitative data to study community level performance (e.g., the number of submitted solutions per challenge). However, based on our qualitative data, we conjecture that collaboration positively influences collective performance: for instance, when a crowd member shares his/her analysis of the pros and cons of a particular scenario, other crowd members can use this analysis to improve their own solutions. This hypothetical result, increased collective performance, is similar to what has been empirically reported by Boudreau [5]. As for individual performance, our results are in line with Tausczik [46] because we identified an association between individual performance (through the outcome in the challenge) and collaboration (through the number of messages sent): the winners of the challenges were often the more collaborative crowd members.

An association between collaboration and performance has been recognized in the software development literature before. Cataldo and colleagues [8], while studying a *non-competitive* scenario, showed that the most productive developers are the ones who communicated with whom they have code dependencies, i.e., they align their work with the “right people”. Our results provide an initial evidence that collaboration also plays a role in competitive scenarios, since we also identified an association between collaboration and performance. Furthermore, our answer to RQ2 suggests that members with different performance discuss different subjects in the forums. In other words, while in non-competitive SW development the most productive developers collaborate with the “right people”, in competitive software crowdsourcing the members who obtain the best results seem to collaborate about the “right subjects”.

It is important to mention that we did not observe instances of socialization messages, something fairly common in online communities. This result is in sharp contrast to Gray et al's [16] who identified that crowd members used additional tools to collaborate and address “unmet social and technological needs posed by the crowdsourcing platform.” This might be explained by the fact that crowd workers have a very limited time to work on the challenges (about 5 days). Another possible explanation is the fact that Topcoder has a global forum, which is independent of challenges, where such socialization might happen. A third possible explanation is based on the observation that crowd members do not remain idle at the end of competitions, i.e., they come and go over time [45] [44], so perhaps this socialization indeed does not happen. Further research in this topic seems necessary to explain this result.

Finally, our study focuses on community-wide collaboration [46] in which crowd members are part of the same temporary online community and help each other, share useful information, discuss documents, etc. In this paper collaboration is not associated to working towards a shared goal, which would imply in two or more interdependent actors who would need to coordinate their individual activities to successfully reach this goal [30]. In fact, in our analysis of the forums, we did not observe instances of coordinated efforts: for instance, two crowd members planning to use different approaches and later on discussing their individual results. This does not mean this is not possible, or that this does not take place. This only means that the forums that we analyzed did not have examples of such coordinated efforts. Previous work [16] suggest that whenever the collaborative features in the crowdsourcing platform are not satisfactory, crowd members use different tools to achieve their goals.

6 DESIGN IMPLICATIONS

Our results suggest that despite the limited support for collaboration available in the Topcoder platform, crowd members do collaborate and this collaboration is beneficial for them. We believe that software crowdsourcing platforms could leverage different mechanisms to make forum posts easier to use and more useful, and by doing so, improve crowd members experience with the platform as well as co-pilots' work while conducting the challenges [13]. For instance, a SW CS platform could detect the issues being discussed and label them accordingly: if a message mentions sections of the requirements documents, or links to external repositories or libraries a system could label the message as Requirements, Access and Libraries, respectively. These labels could be made available to crowd members facilitating the process of finding relevant and timely information in the online forums. In fact, we observed that Quitters often post messages about issues which have already been mentioned before. Given the impact of online forums in the retention of newcomers [47], more sophisticated tools could be used to further improve the forums, e.g., using machine learning techniques to detect the degree of politeness in the forum messages [20].

In addition to crowd members, visualizations about message labels could be made available to co-pilots facilitating their understanding about what is, or is not, being discussed during a particular challenge. Such visualizations could provide an overview of the dynamics of the challenges [51]: if most of the discussion is still about Access and Libraries and the challenge deadline is near, that likely means very few, if any, solutions will be submitted. In other words, visualizations based on labelled messages could help co-pilots make informed decisions about interventions they should adopt to increase the number of submissions and, accordingly the number the quality of solutions – in this case, postpone the deadline. In summary, based on the observation of the importance of online forums, our first recommendation aims to leverage information from the forum messages to improve the experience of crowd members and co-pilots during the challenges .

In the platform studied by Tausczik and Wang[46] the rankings were displayed publicly during the contests. In their design implications, they suggested to remove "leaderboards that show individuals rankings during the contest" to reduce the competitive nature of the challenges since the study showed that the crowd members less likely to collaborate were the ones performing the very best in the challenges. These results are in line with our results since there are no public rankings during the Topcoder's challenges and winners were usually the ones who sent the largest number of messages. One hypothesis is that crowd members are collaborative because they yet do not know they will be winners of the challenges. While there is no information about the members' ranking during the challenge, Topcoder presents information about members' overall ranking and performance in the entire platform. Thus, our second recomendation for software crowdsourcing platforms is carefully consider which information they will make publicly available about the crowd members' ranking during the challenges, so that these members remain motivated to persevere and collaborate in the challenges.

More broadly, we believe a large body of research in online communities can be used to increase the engagement in the challenges. For instance, co-pilots could adopt software tools that provide insights about the health of the online communities, identification of less engaged crowd members, etc [35]. In our analysis a very simple mechanism – the number of messages sent – was associated with the performance in the challenge. When we combine this information with the topics and categories of the messages, we can infer the classification of crowd members in winners, submitters, and quitters. SW CS platforms could use this information as a proxy to monitor the performance of individual participants while the challenges take place. Again, a co-pilot could use this information to decide which actions to take to improve the results of the challenge. We believe some messages could even be automated, for instance, when quitters request more than once access to repositories. Finally, software CS platforms could perform analysis of the affordances provided by alternative interfaces for online forums, as well as, social presence aspects [37] to find out how these different interfaces impact the performance of crowd members [52].

7 LIMITATIONS

As any other empirical study, ours has limitations. As one limitation, our study concentrated in one SW CS platform, Topcoder. Although it is the most used platform, we can not argue that our results would be similar in other platforms. Moreover, our study focused solely on Programming challenges and did not take into account other Topcoder challenges such as Specification, Architecture, Design, etc. These other challenges could present different collaborative behavior, as well as the identification of new categories and topics.

Another limitation of our work is a consequence of our data collection methods: we analyzed messages posted on Topcoder forums. We did not interview crowd members, therefore, we were unable to gather information that can help us to understand the members' rationale for their decisions. Such interviews could be used, for instance, to explore the reasons why crowd members quit challenges, and whether this decision was somehow affected by the information available in the online forums.

The number of analyzed forums and the manual analysis can also be regarded as limitations. This was alleviated by double checking with another researcher: as mentioned before, the first two authors coded independently the messages and, later on, they revisited all the codes to reach an agreement. As for the manual analysis, we plan to develop software tools to automate this process so that we can analyze a larger number of forums in our future work.

Finally, our analysis only identified a *relationship* between the number of messages sent by crowd members and their outcome in the challenges. However, we can not argue that there is a statistical correlation between these aspects. This would require us to analyze a significant larger number of challenges. To show causality, i.e., that more messages leads to improved performance, it would be necessary to go one step further and conduct a controlled experiment.

8 CONCLUSIONS

This research sought to contribute to our understanding of collaboration in competitive software crowdsourcing. We studied collaboration through the analysis of the messages in the Topcoder's forums, i.e., in a natural setting. These forums are created for each challenge in the platform.

Our results suggest that although Topcoder's challenges are competitions for monetary rewards, crowd members do collaborate: they help each other, share useful information, and discuss documents (RQ1). Different types of crowd members engage in similar collaborative behaviour (RQ2) and, finally, there is an association between the number of messages sent by crowd members and the outcome of these members in the challenges (RQ3). This means that preventing communication and collaboration in competitive SW CS, as it has been observed [32], might negatively influence the results of the challenges. Understanding collaboration during software crowdsourcing challenges can be used as a starting point for investigating factors that aim to provide more adequate collaborative mechanisms for crowd members. Crowd members and requesters can both benefit of such improved collaboration: crowd members because of the positive impact in their performance, and requesters because of the quality improvement in the solutions received to address their tasks.

9 ACKNOWLEDGMENTS

The authors thank the Brazilian Council for Research (CNPq) and FAPERGS for the financial support through processes 311256/2018-0, 312870/2018-3 and, 17/2551-0001 205-4.

REFERENCES

[1] Sabrina Adamczyk, Angelika Bullinger-Hoffmann, and Kathrin Moeslein. 2012. Innovation Contests: A Review, Classification and Outlook. *Creativity and Innovation Management* 21 (12 2012). https://doi.org/10.1111/caim.12003
[2] Nikolay Archak. 2010. Money, Glory and Cheap Talk: Analyzing Strategic Behavior of Contestants in Simultaneous Crowdsourcing Contests on TopCoder.com. 21–30. https://doi.org/10.1145/1772690.1772694
[3] Andrew Begel, Jan Bosch, and M.-A Storey. 2013. Social Networking Meets Software Development: Perspectives from GitHub, MSDN, Stack Exchange, and TopCoder. *Software, IEEE* 30 (01 2013), 52–66. https://doi.org/10.1109/MS.2013.13
[4] Kevin Boudreau, Patrick Gaulé, Karim Lakhani, Christoph Riedl, and Anita Woolley. 2014. From Crowds to Collaborators: Initiating Effort Catalyzing Interactions Among Online Creative Workers. *SSRN Electronic Journal* (01 2014). https:

//doi.org/10.2139/ssrn.2384068

[5] Kevin Boudreau and Karim Lakhani. 2015. "Open" Disclosure of Innovations, Incentives and Follow-on Reuse: Theory on Processes of Cumulative Innovation and a Field Experiment in Computational Biology. *Research Policy* 44 (02 2015). https://doi.org/10.1016/j.respol.2014.08.001

[6] Brabham and Daren C. 2008. Crowdsourcing as a Model for Problem Solving: An Introduction and Cases. *The International Journal of Research into New Media Technologies* 14 (1) (01 2008), 75–90.

[7] Susan L. Bryant, Andrea Forte, and Amy Bruckman. 2005. Becoming Wikipedian: Transformation of Participation in a Collaborative Online Encyclopedia. In *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work (GROUP '05)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/1099203.1099205

[8] Marcelo Cataldo, Patrick A. Wagstrom, James D. Herbsleb, and Kathleen M. Carley. 2006. Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools. In *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work (CSCW '06)*. Association for Computing Machinery, New York, NY, USA, 353–362. https://doi.org/10.1145/1180875.1180929

[9] Lars Christensen. 2014. Practices of Stigmergy in the Building Process. *Computer Supported Cooperative Work (CSCW)* 23 (02 2014). https://doi.org/10.1007/s10606-012-9181-3

[10] Andrew L. Cohen, Debra Cash, and Michael J. Muller. 2000. Designing to Support Adversarial Collaboration. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW '00)*. Association for Computing Machinery, New York, NY, USA, 31–39. https://doi.org/10.1145/358916.358948

[11] J. Corbin and A. Strauss. 2008. *Basics of Qualitative Research (3rd ed.): Techniques and Procedures for Developing Grounded Theory.* Sage.

[12] John W Creswell. 2013. *Research design: Qualitative, quantitative, and mixed methods approaches.* Sage.

[13] Cleidson R. B. de Souza, Leticia S. Machado, and Ricardo Rodrigo M. Melo. 2020. On Moderating Software Crowdsourcing Challenges. *Proc. ACM Hum.-Comput. Interact.* 4, GROUP, Article 14 (Jan. 2020), 22 pages. https://doi.org/10.1145/3375194

[14] Anhai Doan, Raghu Ramakrishnan, and Alon Y. Halevy. 2011. Crowdsourcing Systems on the World-Wide Web. *Commun. ACM* 54, 4 (April 2011), 86–96. https://doi.org/10.1145/1924421.1924442

[15] Alpana Dubey, Kumar Abhinav, Sakshi Taneja, Gurdeep Virdi, Anurag Dwarakanath, Alex Kass, and Suma Mani. 2016. Dynamics of Software Development Crowdsourcing. https://doi.org/10.1109/ICGSE.2016.13

[16] Mary L. Gray, Siddharth Suri, Syed Shoaib Ali, and Deepti Kulkarni. 2016. The Crowd is a Collaborative Network. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, USA, 134–147. https://doi.org/10.1145/2818048.2819942

[17] Rebecca Grinter. 1996. Supporting Articulation Work Using Software Configuration Management Systems. *Computer Supported Cooperative Work (CSCW)* 5 (12 1996), 447–465. https://doi.org/10.1007/BF00136714

[18] Rebecca Grinter. 2003. Recomposition: Coordinating a Web of Software Dependencies. *Computer Supported Cooperative Work* 12 (09 2003), 297–327. https://doi.org/10.1023/A:1025012916465

[19] Christian Heath and Paul Luff. 1992. Collaboration and Control: Crisis Management and Multimedia Technology in London Underground Line Control Rooms. *Computer Supported Cooperative Work - CSCW* 1 (03 1992), 69–94. https://doi.org/10.1007/BF00752451

[20] Erin Hoffman, David McDonald, and Mark Zachry. 2017. Evaluating a Computational Approach to Labeling Politeness: Challenges for the Application of Machine Classification to Social Computing Data. *Proceedings of the ACM on Human-Computer Interaction* 1 (12 2017), 1–14. https://doi.org/10.1145/3134687

[21] 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.

[22] Katja Hutter, Julia Hautz, Johann Füller, Julia Mueller-Seeger, and Kurt Matzler. 2011. Communitition: The Tension Between Competition and Collaboration in Community Based Design Contests. *Creativity and Innovation Management* 20 (02 2011), 3–21. https://doi.org/10.1111/j.1467-8691.2011.00589.x

[23] Evgeny Kaganer, Erran Carmel, Rudy Hirschheim, and Olsen Timothy. 2013. Managing the Human Cloud. *MIT Sloan Management Review* 54 (12 2013), 23.

[24] Aniket Kittur. 2010. Crowdsourcing, Collaboration and Creativity. *ACM Crossroads* 17 (12 2010), 22–26. https://doi.org/10.1145/1869086.1869096

[25] Aniket Kittur, Jeffrey Nickerson, Michael Bernstein, Elizabeth Gerber, Aaron Shaw, John Zimmerman, Matt Lease, and John Horton. 2013. The Future of Crowd Work. *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*, 1301–1318. https://doi.org/10.1145/2441776.2441923

[26] Karim Lakhani, David Garvin, and Eric Lonstein. 2010. TopCoder (A): Developing Software through Crowdsourcing. (01 2010).

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

[28] Thomas LaToza and Andre van der Hoek. 2016. Crowdsourcing in Software Engineering: Models, Motivations, and Challenges. *IEEE Software* 33 (01 2016), 74–80. https://doi.org/10.1109/MS.2016.12

[29] Thomas D. LaToza, W. Ben Towne, Christian M. Adriano, and André van der Hoek. 2014. Microtask Programming: Building Software with a Crowd. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. Association for Computing Machinery, New York, NY, USA, 43–54. https://doi.org/10.1145/2642918.2647349

[30] Charlotte P. Lee and Drew Paine. 2015. From The Matrix to a Model of Coordinated Action (MoCA): A Conceptual Framework of and for CSCW. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work Social Computing (CSCW '15)*. Association for Computing Machinery, New York, NY, USA, 179–194. https://doi.org/10.1145/2675133.2675161

[31] Benedikt Ley, Thomas Ludwig, Volkmar Pipek, Dave Randall, Christian Reuter, and Torben Wiedenhoefer. 2014. Information and Expertise Sharing in Inter-Organizational Crisis Management. *Computer Supported Cooperative Work (CSCW)* 23 (12 2014), 347–387. https://doi.org/10.1007/s10606-014-9205-2

[32] Leticia Machado, Alexandre Zanatta, Sabrina Marczak, and Rafael 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

[33] Thomas Malone, Robert Laubacher, and Chrysanthos Dellarocas. 2010. The Collective Intelligence Genome. *IEEE Engineering Management Review* 38 (03 2010), 38–52. https://doi.org/10.1109/EMR.2010.5559142

[34] Ke Mao, Licia Capra, Mark Harman, and Yue Jia. 2016. A Survey of the Use of Crowdsourcing in Software Engineering. *Journal of Systems and Software* 126 (09 2016). https://doi.org/10.1016/j.jss.2016.09.015

[35] Tara Matthews, Steve Whittaker, Hernan Badenes, Barton A. Smith, Michael Muller, Kate Ehrlich, Michelle X. Zhou, and Tessa Lau. 2013. Community Insights: Helping Community Leaders Enhance the Value of Enterprise Online Communities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 513–522. https://doi.org/10.1145/2470654.2470728

[36] Sreeja Nag, Ira Heffan, Alvar Saenz-Otero, and Mike Lydon. 2012. SPHERES Zero Robotics Software Development: Lessons on Crowdsourcing and Collaborative Competition. https://doi.org/10.1109/AERO.2012.6187452

[37] Catherine S. Oh, Jeremy N. Bailenson, and Gregory F. Welch. 2018. A Systematic Review of Social Presence: Definition, Antecedents, and Implications. *Frontiers in Robotics and AI* 5 (2018), 114. https://doi.org/10.3389/frobt.2018.00114

[38] Wanda J. Orlikowski. 1992. Learning from Notes: Organizational Issues in Groupware Implementation. In *Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work (CSCW '92)*. ACM, New York, NY, USA, 362–369. https://doi.org/10.1145/143457.143549

[39] Xin Peng, Muhammad Ali Babar, and Christof Ebert. 2014. Collaborative Software Development Platforms for Crowdsourcing. *Software, IEEE* 31 (03 2014), 30–36. https://doi.org/10.1109/MS.2014.31

[40] 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

[41] 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).

[42] Gregory Saxton, Onook Oh, and Rajiv Kishore. 2013. Rules of Crowdsourcing: Models, Issues, and Systems of Control. *Information Systems Management* (01 2013). https://doi.org/10.1080/10580530.2013.739883

[43] Eric Schenk and Claude Guittard. 2011. Towards a characterization of crowdsourcing practices. *Journal of Innovation Economics* n°7 (01 2011), 93–107. https://doi.org/10.3917/jie.007.0093

[44] 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 (11 2017). https://doi.org/10.1109/TSE.2017.2774297

[45] Klaas-Jan Stol and Brian Fitzgerald. 2014. Two's Company, Three's a Crowd: A Case Study of Crowdsourcing Software Development. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, New York, NY, USA, 187–198. https://doi.org/10.1145/2568225.2568249

[46] Yla Tausczik and Ping Wang. 2017. To Share, or Not to Share?: Community-Level Collaboration in Open Innovation Contests. *Proc. ACM Hum.-Comput. Interact.* 1, CSCW, Article 100 (Dec. 2017), 23 pages. https://doi.org/10.1145/3134735

[47] Hon Jie Teo and Aditya Johri. 2014. Fast, Functional, and Fitting: Expert Response Dynamics and Response Quality in an Online Newcomer Help Forum. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work amp; Social Computing (CSCW '14)*. Association for Computing Machinery, New York, NY, USA, 332–341. https://doi.org/10.1145/2531602.2531731

[48] TopCoder. 2017. *TopCoder*. Retrieved March 18, 2019 from https://www.topcoder.com

[49] Luis Vaz, Sabrina Marczak, and Igor Steinmacher. 2018. An empirical study on task documentation in software crowdsourcing: the case of the topcoder platform. 62–71. https://doi.org/10.1145/3266237.3266265

[50] Fernanda Viegas, Martin Wattenberg, and Kushal Dave. 2004. Studying Cooperation and Conflict between Authors with History Flow Visualizations. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* 6. https://doi.org/10.1145/985692.985765

[51] Fernanda B. Viégas, Martin Wattenberg, and Kushal Dave. 2004. Studying Cooperation and Conflict between Authors with <i>History Flow</i> Visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04).* Association for Computing Machinery, New York, NY, USA, 575–582. https://doi.org/10.1145/985692.985765

[52] Yi-Chia Wang, Mahesh Joshi, and Carolyn Rosé. 2008. Investigating the effect of discussion forum interface affordances on patterns of conversational interactions. 555–558. https://doi.org/10.1145/1460563.1460650

[53] Wenjun Wu, Wei-Tek Tsai, and Wei Li. 2013. An evaluation framework for software crowdsourcing. *Frontiers of Computer Science: Selected Publications from Chinese Universities* 7 (10 2013), 694–709. https://doi.org/10.1007/s11704-013-2320-2

[54] Ye Yang, Muhammad Rezaul Karim, Razieh Saremi, and Guenther Ruhe. 2016. Who Should Take This Task?: Dynamic Decision Support for Crowd Workers. 1–10. https://doi.org/10.1145/2961111.2962594

[55] Ye Yang and Razieh Saremi. 2015. Award vs. Worker Behaviors in Competitive Crowdsourcing Tasks. *Empirical Software Engineering and Measurement (ESEM), 2015 ACM/IEEE International Symposium on* (10 2015).

[56] Alexandre Zanatta, Leticia Machado, Graziela Pereira, Rafael Prikladnicki, and Erran Carmel. 2016. Software Crowdsourcing Platforms. *IEEE Software* 33 (11 2016), 112–116. https://doi.org/10.1109/MS.2016.151

Received August 2020; revised November 2020; accepted December 2020