

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

LETICIA DOS SANTOS MACHADO

**EMPIRICAL STUDIES ABOUT COLLABORATION IN COMPETITIVE SOFTWARE
CROWDSOURCING**

Porto Alegre
2018

PÓS-GRADUAÇÃO - *STRICTO SENSU*



Pontifícia Universidade Católica
do Rio Grande do Sul

Ficha Catalográfica

M149e Machado, Leticia dos Santos

Empirical studies about collaboration in competitive software crowdsourcing / Leticia dos Santos Machado . – 2018.

161 p.

Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Rafael Prikladnicki.

Co-orientador: Prof. Dr. Cleidson R B de Souza.

1. Software Engineering. 2. Software Crowdsourcing. 3. Collaboration. 4. Crowd. 5. Communication. I. Prikladnicki, Rafael. II. de Souza, Cleidson R B. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).

Bibliotecária responsável: Salete Maria Sartori CRB-10/1363

Leticia Santos Machado

Empirical studies about collaboration in competitive software crowdsourcing

This Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on March 28, 2018

Committee Members:

Prof. Dr. Fernando Figueira Filho (UFRN)

Prof. Dra. Milene Selbach Silveira (PPGCC/PUCRS)

Prof. Dra. Tayana Uchôa Conte (PPGI/UFAM)

Prof. Dr. Cleidson R. B. de Souza (PPGCC/UFPA - Co-advisor)

Prof. Dr. Rafael Prikladnicki (PPGCC/PUCRS – Advisor)

DEDICATION

To my daughter Laura

ACKNOWLEDGEMENTS

After spending these past few years working on my doctorate, I have many people I would like to thank.

First of all, I would like to thank my advisor, Rafael Prikladnicki, for his guidance. He has always inspired and encouraged in leaving the comfort zone.

Second, but not least important, I would like to thank my co-advisor Cleidson R.B. de Souza. The guidance and helpful discussions through all these years have put me on track whenever I got lost.

I extend this gratitude to Prof. André van der Hoek from UCI for having welcomed me virtually in his research group and for the opportunity to interact during the whole time we spent designing the experiment. It was a valuable lesson.

This PhD would not be possible without the agreement PUCRS/HPE for having financed and for having encouraged my research. I extend this greeting to Brazilian Science without Borders Program Project.

I am very grateful to my first advisor Karin Becker during my master course. She opened my eyes to research activity. I extend this grateful for all the members of my qualifying committee and final examining committee. It was a pleasure to receive feedback and comments from a group of such outstanding researchers: Sabrina Marczak, Igor Steinmacher, Erran Carmel, Milene Selbach Silveira, Tayana Conte, and Fernando Figueira Filho.

Along these years, I also made new friends and colleagues. I thank each and every one of them for their friendship: Caroline Queiroz, Carolina Paz, Alexandre Zanatta, Josiane Kroll, Ricardo Marinho, Bernardo Estácio, Graziela Basílio Pereira, Carolina Toscani, Júlia Couto, Olimar Borges, Alessandra Costa. I could not forget all the colleagues from MunDDos research group.

I cannot forget the PPGCC staff for their prompt help. Special thanks go to Régis Escobal da Silva and Diego Cintrão.

Thank you, all participants of the conducted empirical studies, for having accepted to contribute with this research.

Finally, I would like to thank my family: my husband Sébastien, who understood and pushed me to move forward during this journey; my little daughter Laura, my parents, who understood the reasons for my absence. This thesis is for and because of you.

EMPIRICAL STUDIES ABOUT COLLABORATION IN COMPETITIVE SOFTWARE CROWDSOURCING

ABSTRACT

Software Crowdsourcing (SW CS) is an emergent software development strategy where a large number of people have been engaged to contribute in several software activities. Such strategy (based on the crowd), has been used for companies who are seeking to increase the speed of their software development efforts. This strategy is usually structured around platforms that allow a requester submit a task to be performed and connect with the crowd that assigned and provide a solution for the task. These platforms usually explore a competitive approach: members of the crowd independently create a solution while compete against each other by monetary rewards for task completion. While competition usually reduces collaboration, some recent studies surprisingly indicate that there is collaboration in SW CS platforms. These studies have focused on two aspects. First, collaboration concerns between platform and requester in terms of crowd's assignment to the challenges (task allocation and submission) and second, the impact of the collaboration among crowd members in the quality of the submitted solutions. Other aspects of the collaboration among crowd members have been largely unexplored. In this thesis, our goal is to identify collaboration's characteristics and barriers faced by crowd members in competitive software crowdsourcing. To achieve this goal, we have conducted multiple studies, using mixed research methods divided in two phases: one exploratory and one evaluatory. For the exploratory phase, we used data collected from: (i) the three involved parties in SW CS projects (requester, crowd and platform) through semi structured interviews with practitioners and companies, (ii) studies selected via literature review; and (iii) an empirical study about how developer collaborated with each other in a SW CS competitive platform – TopCoder. The most frequent collaboration barrier was associated to lack of proper communication among the parties. Based on this barrier we decided, in the evaluatory phase, to conduct a (iv) qualitative analysis of the main communication channel used by the crowd: forums hosted on TopCoder platform and (v) a survey aimed at developers who had competed on TopCoder to assess the influence of collaboration in task performance. Our results from these evaluatory studies suggest that collaboration among crowd members is correlated with delivering winning solutions in SW CS challenges.

Keywords: Software Engineering, Software Crowdsourcing, Collaboration, Barriers, Characteristics, Communication, Competition, Software Development, Crowd, Platforms, Challenges.

ESTUDOS EMPÍRICOS SOBRE COLABORAÇÃO EM SOFTWARE CROWDSOURCING COMPETITIVO

RESUMO

Software Crowdsourcing (SW CS) é uma estratégia emergente de desenvolvimento de software onde um grande número de pessoas tem se engajado para contribuir em várias atividades de *software*. Tal estratégia (baseada na multidão), tem sido utilizada pelas empresas que estão buscando aumentar a velocidade de seus esforços em desenvolvimento de software. SW CS está geralmente estruturado em torno de plataformas que permitem que um solicitante submeta uma tarefa e conecte-a com uma multidão de pessoas que irá prôver soluções para a tarefa. Essas plataformas geralmente exploram uma abordagem competitiva para realização da tarefa: membros da multidão, independentemente, criam uma solução para a tarefa enquanto competem uns contra os outros em busca de uma premiação financeira ao final da tarefa entregue. Uma vez que a competição pode reduzir a colaboração, recentes estudos, surpreendentemente, indicam que a colaboração existe em plataformas de SW CS. Estes estudos têm focado em dois aspectos. O primeiro, em problemas de colaboração entre plataforma e solicitante com relação a atribuição da multidão e as tarefas a serem desenvolvidas nos desafios de competição (alocação e submissão de tarefas) e, o segundo aspecto, relacionado ao impacto da colaboração entre membros da multidão e a qualidade das soluções submetidas. Outros aspectos referentes a colaboração entre os membros da multidão ainda são amplamente inexplorados. Nessa tese, nosso objetivo é identificar barreiras e características de colaboração enfrentadas pelos membros da multidão em SW CS competitivo. Para alcançar este objetivo, nós conduzimos múltiplos estudos utilizando diferentes métodos de pesquisa divididos em duas fases: exploratória e avaliadora. Para a fase exploratória, os dados coletados foram obtidos a partir de: (i) partes envolvidas em projetos de SW CS (solicitante, multidão e plataforma) através de entrevistas semi-estruturadas com profissionais e empresas, (ii) estudos selecionados através da revisão da literatura e; (iii) estudo empírico sobre como desenvolvedores colaboram entre si em uma plataforma de SW CS competitivo – TopCoder. A barreira de colaboração mais frequente encontrada está associada a falta de comunicação apropriada entre as partes. Baseado nessa barreira decidimos na fase avaliadora conduzir uma (iv) análise qualitativa do principal canal de comunicação utilizado pela multidão: fóruns hospedados na plataforma TopCoder e, finalmente, (v) realizamos um survey destinado aos desenvolvedores que competiram na TopCoder para avaliar a influência da colaboração no desempenho da tarefa. Os resultados obtidos nos estudos avaliatórios sugerem que a colaboração entre os membros da multidão está correlacionada com a entrega de soluções de software vencedoras nos desafios de SW CS.

Palavras-chave: Engenharia de *Software*, *Software Crowdsourcing*, Colaboração, Barreiras, Características, Comunicação, Competição, Desenvolvimento de software, *Crowd*, Plataformas, Desafios.

LIST OF FIGURES

Figure 1 - Research Design	15
Figure 2 - SW CS involved parties	25
Figure 3 - Software development strategies.....	27
Figure 4 – TopCoder phase	35
Figure 5 - TopCoder Screen – List of challenges.....	36
Figure 6 - TopCoder screen – registration screen details.....	37
Figure 7 - Communication flow among involved participants	38
Figure 8 - Challenge’s Forum	38
Figure 9 - Participant's information.....	43
Figure 10 - Extracted data from collaboration barriers	51
Figure 11 - Statistics of barriers to collaboration in SW CS.....	53
Figure 12 - Collaboration barriers model	81
Figure 13 - Model identified per study.....	82
Figure 14 -Example of selectors used for data collection	84
Figure 15 - Example of coding from messages in the forum	90
Figure 16 - Number of messages sent in forums between copilots and the crowd... 93	
Figure 17 – Coder’s classification.....	98
Figure 18 – Message category	100
Figure 19 – Message topics in the forums.....	101
Figure 20 - Forum categories of winners and submitters.....	102
Figure 21 – Forums topics for winners and submitters	102
Figure 22 – Winner communication pattern by categories and topics.....	107
Figure 23 – Submitter communication pattern by categories and topics.....	110
Figure 24 – Quitter communication patterns by categories and topics.....	112
Figure 25 – Number of times participants registered.....	116
Figure 26 - Number of solutions submitted.....	117
Figure 27 – Total of submitters and quitters.....	117
Figure 28 - Summative between submission and communication.....	118
Figure 29 - Results of questions about communication vs task performance.....	120
Figure 30 - Communication channel used between crowd.....	126
Figure 31 - Communication channel used between other developers.....	126

LIST OF TABLES

Table 1 - Crowds' characteristics.....	24
Table 2 - Requester's characteristics.....	24
Table 3 – Platform's characteristics	25
Table 4 – Task's characteristics	25
Table 5 - Enablers and blockers' aspects in SW CS.....	44
Table 6 - Review Literature in SW CS publications.....	49
Table 7 - Snowballing search.....	49
Table 8 - Final set of publications.....	50
Table 9 - Map of Collaboration barriers from LR.....	52
Table 10 - Collaboration barriers and SW CS elements	68
Table 11 - Collaborations barriers from crowd	77
Table 12 - Summary of criteria definitions	86
Table 13 - Examples from category coding.....	87
Table 14 – Message Categories.....	87
Table 15 – Message Topics.....	88
Table 16 - Examples from topics coding.....	90
Table 17 - Challenge data.....	92
Table 18 - Winner coders in the forum.....	94
Table 19 - Winner's communication.....	96
Table 20 - Winners who did not communicate	96
Table 21 – Winners who communicated.....	97
Table 22 - Questions and Answers on Help – <i>Requirements</i>	103
Table 23 – Questions on Identified problems – <i>Processing</i>	104
Table 24 - Questions on Identified problems – <i>Requirements</i>	104
Table 25 - Questions on Confirmation request - <i>Deadline</i>	105
Table 26 - Questions and Answers on Invitation - <i>Access</i>	105
Table 27 - Tips on <i>Processing</i>	106
Table 28 – Winner Quotes from <i>Problems response</i> category.....	109
Table 29 - Quotes winner from Identified problem on Requirements	109
Table 30 - Quotes from Quitters	112
Table 31 - Total number of participants in the communication forum.....	118
Table 32 - Quotes of survey's participant who did not communicate.....	120
Table 33 –Question 2 and quotes by participants who communicate	121
Table 34 - Question 2 and quotes by participants who did not communicate.....	122
Table 35 - Question 4 and quotes by participants who communicate	123
Table 36 – Question 4 and quotes by participants who did not communicate.....	123
Table 37 - Question 6 and quotes by participants who communicate	124
Table 38 - Question 6 and quotes by participants who did not communicate.....	125
Table 39 - Example of quotes about latency information barrier.....	131
Table 40 - SW CS collaboration recommendations.....	133

LIST OF ABBREVIATIONS

AMT	Amazon Mechanical Turk
API	Application Programming Interface
CSCW	Computer Supported Cooperative Work
OSS	Open Source Software
GT	Grounded Theory
HIT	Human Intelligence Tasks
IEEE	Institute of Electrical and Electronics Engineers
SW CS	Software Crowdsourcing
Q&A	Questions and Answers
CS	Crowdsourcing
IT	Information Technology
AI	Artificial Intelligence
HTML	Hypertext Markup Language
GUI	Graphical User Interface
DSD	Distributed Software Development
CSV	Comma Separated Values

TABLE OF CONTENTS

1. INTRODUCTION	13
1.1. Research design and thesis organization	14
1.2. Scope	16
1.3. Main Contributions	17
1.4. Other Results	19
1.4.1 Published Papers	19
1.5.1 Participations	20
1.5.2 Advising	21
1.6. Funding	21
1.7. Collaboration, Research Visits, and Course Work	21
2. THEORETICAL BACKGROUND	23
2.1. Software Crowdsourcing	23
2.1.1 SW CS Elements	23
2.1.2 SW CS characteristics	26
2.2. Application models in SW CS	27
2.3. Phases of software development and SW CS platforms	28
2.4. Opportunities in SW CS	30
2.5. Challenges in SW CS	30
2.6. Collaborative software development	31
2.6.1 Collaboration challenges in software development	32
2.7. Competition Software Development	33
2.7.1 Forum Communication	37
3. COLLABORATION BARRIERS IN SW CS	41
3.1. Exploratory Phase	41
3.2. Semi structured interview from three SW CS elements	41
3.2.1 Settings and methodology	41
3.2.2 Data Analysis	43
3.2.3 Results	44
3.2.4 Discussion	45
3.3. Literature review from three SW CS elements	47
3.3.1 Settings and methodology	47
3.3.2 Data Analysis	47
3.3.3 Results	50
3.3.4 Discussion	67
3.4. Case Study by Crowds' participants	71
3.4.1 Settings and methodology	71
3.4.2 Data Analysis	73
3.4.3 Results	73
3.4.4 Discussion	79
4. COLLABORATION CHARACTERISTICS	83
4.1. Evaluatory Phase	83
4.2. Qualitative analysis of communicattion's forums	83
4.2.1 Settings and methodology	83
4.2.2 Data Collection	85
4.2.3 Data Analysis	86
4.3. Initial Results	91
4.3.1 Coders and Copilots collaboration	91

4.4	Results about Coder’s communication and performance	95
4.4.1	Winners who communicate vs Winners who did not communicate.....	95
4.4.2.	Winners group who communicated the most (C1, C2, C3) vs Winners who communicate (C4-n).....	96
4.4.3	Discussion.....	98
4.5	Communication patterns from coders	100
4.5.1	Introduction	100
4.5.2	Common communication patterns for winner, submitter and quitter	103
4.5.3	Winners Communication Patterns	106
4.5.4.	Communication Patterns of Submitters	110
4.5.5	Communication Patterns of Quitters.....	111
4.5.6	Discussion.....	113
4.5.7	Limitations.....	114
4.6	Survey Data	114
4.6.1	Introduction	114
4.6.2	Data Collection	115
4.6.3	Demographics information	116
4.6.4	Results.....	118
4.6.5	Discussion.....	127
4.6.6	Limitations.....	128
5.	DISCUSSION	129
6.	CONCLUSIONS AND FUTURE WORK	135
	REFERENCES	139
	APPENDIX A	148
	APPENDIX B	149
	APPENDIX C	151
	APPENDIX D	153
	APPENDIX E	154
	APPENDIX F	156
	APPENDIX G	161

1. INTRODUCTION

Nowadays, software products are built by many people located in different places worldwide [KAG13], [AGE15], [KAL15]. Software Crowdsourcing, or simply SW CS, is a particular way of designing and creating software through the engagement of a global pool of online workers – the crowd – who can be tapped on-demand to contribute to various types of software development tasks (e.g., requirements, design, coding, testing, evaluations, and maintenance) [PRI14], [STO14].

The first SW CS studies were published by Archak [ARC10], Begel et al. [BEG13] and LaToza et al. [LAT13]. After these studies, this topic has raised interest of different researchers. For instance, Stol and Fitzgerald [STO14] describe a series of issues in crowdsourced software development projects, including communication and coordination, and quality assurance, among other aspects. Other authors have built tools to support and explore software crowdsourcing. LaToza et al. [LAT14], for example, developed a crowdsourcing application called CrowdCode for decomposing programming work into micro-tasks.

SW CS is usually structured around platforms. These are marketplaces that allow requesters to seek workers to perform their tasks and, at the same time, support workers in finding tasks to work on. Examples of SW CS platforms include TopCoder [TOP17], uTest¹, and Passbrains² [ZAN16]. In general, these platforms explore a competitive approach [LAT16], in which crowd workers independently create solutions, competing against each other anchored by a monetary reward after task completion. While competition reduces collaboration [HUT11], some studies [NAG12], [BOU14], [GRA16] have surprisingly indicated that there is collaboration in competitive SW CS platforms. In other words, the work conducted by a crowd is not as independent, autonomous, and isolated as it is often assumed to be [GRA16]. In fact, recent results suggest that collaboration can improve the quality and quantity of crowdsourced task submissions [LAT15], [TAU17], [YAN16]. SW CS platforms play an important role in supporting, or hindering, collaboration [NAG12], [PEN14]. For instance, Machado and colleagues [MAC17], found out that TopCoder supports communication among crowd members in a restricted way, given the fact that TopCoder explores a competitive model.

The context described above suggests there is a gap between research and practice: current competitive SW CS platforms provide limited support for collaboration, but, at the same time, there is evidence that collaboration does take place in SW CS, which increases the quality of the crowdsourced task. To address such disconnection between research and practice, we have conducted multiple studies, using mixed research methods, to understand how collaboration takes place in SW CS, including the barriers participants face, as well as the characteristics of this collaboration, including which participants are more likely to collaborate with, and how this collaboration correlates with participants' performance in the tasks.

¹ <https://www.utest.com>

² <http://www.passbrains.com>

The broader research focus of this work is on identifying the barriers and characteristics of collaboration among participants in competitive SW CS. Such focus cannot be explored within the limits of a single thesis; thus,

To be more specific, the questions addressed by this thesis are:

- **RQ1.** Which collaboration barriers do the crowd face when performing tasks in a competitive SW CS environment?
- **RQ2.** Which current collaboration characteristics are present in competitive SW CS?
- **RQ3.** How might the collaboration impact in crowd productivity?

The study presented in this thesis was divided in two phases, one exploratory and one evaluatory. In the exploratory phase, we began studying whether collaboration takes place in SW CS, and if so, which collaboration barriers participants faced. In this case, we used semi-structured interviews and a literature review as research methods. In addition, we conducted an exploratory case study to identify the collaboration barriers faced by the crowd when trying to submit their solutions to the most important SW CS platform, TopCoder. The main aspect underlying these collaboration barriers was the communication among members. Therefore, in the confirmatory phase, we decided to conduct a qualitative analysis of the main communication channel used by the crowd: forums hosted on the TopCoder platform. To validate our results, we carried out a survey with developers who had competed on TopCoder.

Our results suggest that collaboration can be found among crowd competitors, crowd-platform, and the platform requester. However, several barriers are faced, generating tension during the SW CS contest, and reflecting on the quality of performed tasks when submitting and winning tasks. Finally, we conclude this thesis by suggesting practical implications on the role of collaboration for the crowd, requester, and the competitive SW CS platform.

1.1. Research design and thesis organization

As presented in the previous section, our goal is to identify collaboration's characteristics and barriers faced by crowd members in competitive SW CS platforms. In order to do so, we adopted qualitative methods of investigation for data collection and analysis. This perspective allowed us to collect information about the collaboration barriers participants face during a competition, and, more importantly, about the characteristics of such collaboration, including who participants are more likely to collaborate with, and how this collaboration correlated with participants' performance in SW CS contests.

We used multiple empirical methods to address this thesis' research problem and answer the research questions. More specifically, this research results from combining a literature review, a qualitative analysis of semi-structured interviews, an exploratory case

study, a qualitative analysis of textual messages from communications on forums hosted on TopCoder platform, and a survey with developers who had competed on TopCoder.

A better understanding of the analyzed context, the people, the artifacts, the processes, the elements, and the relationships involved are necessary actions to reach the proposed goal of this research. With so, it is possible to find issues and points of improvement in the analyzed situation. Figure 1. summarizes the phases (exploratory and evaluatory), and the research strategies employed in this thesis.

Phase I – Exploratory. This phase comprises studies conducted with participants involved in projects in SW CS: the requester, the platform, and the crowd. In the first step, we investigated how SW CS has been adopted in the IT industry, and the main challenges faced during its adoption, i.e., we adopted to work at a macro level of analysis. This was an (i) empirical study using semi-structured interviews with practitioners and companies. We observed that collaboration issues, scarce context on project information, and unclear documentation were by far the most mentioned challenges interviewees reported [PRI14], [MAC16]. In addition, we conducted a (ii) literature review of the available data repository provided by [MAO15a], combined with a snowballing search approach [WOH14] aiming to organize the collaboration barriers identified in the studies. We observed that the collaboration barriers impacted all participants, but, most significantly, on the performance of crowd participants.

In order to empirically explore which barriers crowd participants faced while performing a task in competitive software crowdsourcing, we also conducted (iii) an exploratory case study on TopCoder platform (one of the most important competitive SW CS platforms) [MAC17]. This study suggested that collaboration issues play a central role on competitive SW CS environments in terms of communication.

Thus, the main contribution of the exploratory phase was aggregating and organizing the collaboration barriers evidenced by different studies, and creating a collaboration barriers' model in SW CS.

Phase II – Evaluatory. The evaluatory phase is comprised of two studies. Our goal was to evaluate the collaboration barriers identified in Phase I, but now at the micro level, therefore we focused specifically on crowd members. First, we performed (iv) a qualitative empirical study of the main communication channel – forums – used by crowd participants on TopCoder platform. Collecting detailed data by content post from communication forums is a novel process in SW CS studies since, as far as we know, no previous work has reported content analysis from TopCoder's forum as a strategy of research and analysis in the SW CS area. The outcomes obtained from this analysis provided evidence from the collaboration characteristics, including which participants are more likely to collaborate with, and how this collaboration reflects in the task performance and quality in SW CS contests.

To validate these results, we carried out (v) a last empirical study based on a small survey with developers who had competed on TopCoder. The results suggest that the potential benefits of collaboration among crowd members is correlated with delivering winning solutions in competitive SW CS.

This thesis is organized as follows: in Chapter 2, we present a broader review of

software crowdsourcing, including application models and platforms. In Chapter 3, we describe the collaboration barriers' model obtained through the exploratory phase, followed by the description of the adopted methods for data collection, and the analysis in each one of the settings. Chapter 4 presents the results of the evaluatory phase, illustrating the qualitative analysis and the data from the survey. It discusses the correlation between collaboration characteristics and task performance. Finally, in Chapter 5, we present the discussion about our research, and, in Chapter 6, our conclusions and future directions.

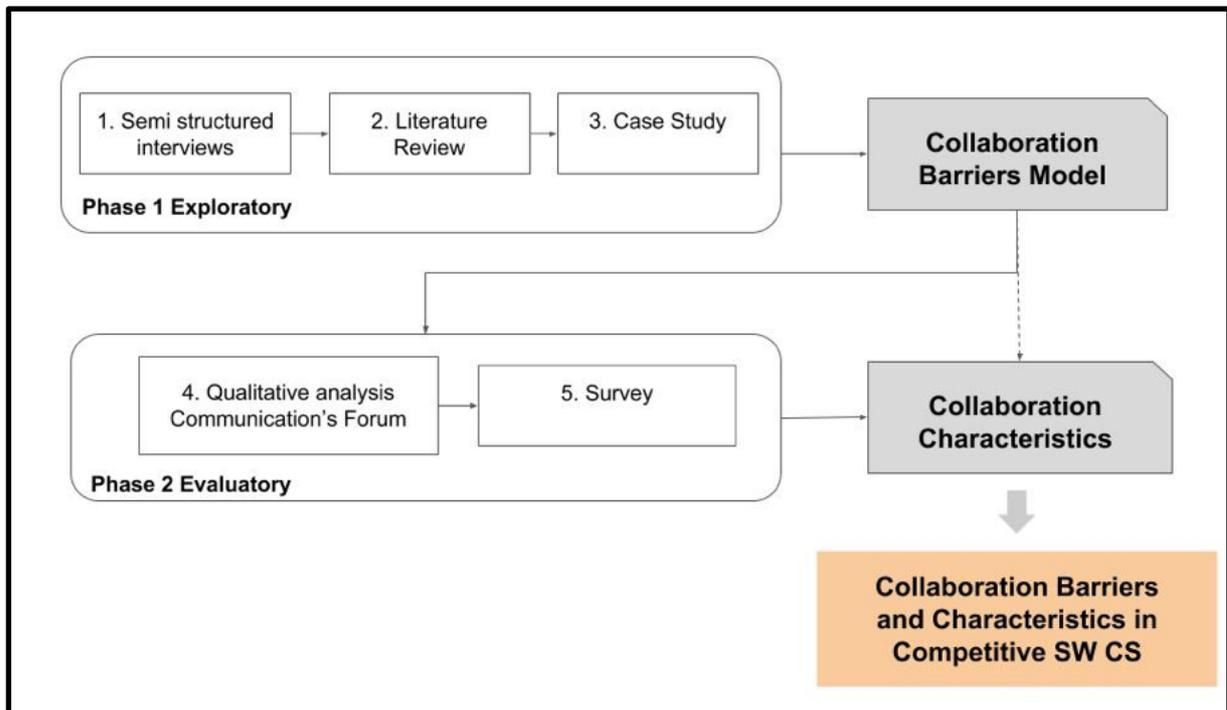


Figure 1 – Research Design

1.2. Scope

In this subsection, we describe the scope of this research and define some terms. Regarding of focus, we restrict this thesis to paid and competitive software crowdsourcing models instead of general crowdsourcing projects, because of the distinct nature of the former. Therefore, our claims are not directly generalizable to SW CS in other models.

In addition, when we refer to “software crowdsourcing” we mean open call [HOW06] within software engineering tasks [MAO15b]. We focus on competition crowdsourcing model [LAT16], in which development is typically carried out by an undefined and potentially large group of online workers (the crowd), who competes with each other. In an open call from competitive crowdsourcing software development, projects, challenges, and tasks can be publicly visualize. However, to access the complete parties of the project (specification documents, artifacts, communication channel, etc.) by any skilled person who wants to participate of the challenges (s)he first needs to became a member of the current SW CS platforms, choose, and register in a challenge.

In our study, we investigate the crowd-level collaboration with other participants (crowd, platform and requester), in several challenges on one site, TopCoder – the leading online SW CS contest platform. On TopCoder, participants compete for monetary prizes, reputation points, and, sometimes, job opportunities.

In addition, we avoid projects from programming marathons, data science, and microtasking contests, including platforms for specific software development phases such as crowdtesting, or crowd design; and platforms for analytics and predictive modeling. These kinds of crowdsourcing projects have other characteristics and may involve different levels of collaboration, which can obscure some possible barriers encountered by participants, since the projects are related to domain and specific technologies.

In TopCoder platform there are different categories and subcategories for those participating in SW CS project contests, as mentioned before, including algorithm marathons, design, data science, and development. In this thesis, we solely focus on the development challenges' category and the coding tasks' subcategory during registration and submission phases. Therefore, we define crowd participants as developers (i.e., people with a development background), who want to participate in a competitive SW CS project, regardless of whether or not they have won previous challenges. More specifically, the crowd participants on the analyzed platform, TopCoder, are mentioned in this thesis as coders (i.e., platform members who participate in the development challenges).

We aim to identify how characteristics collaboration takes place in competitive SW CS among the involved participants. We are not interested in analyzing what had called crowd participants' attention to join a contest, for instance.

Instead, we seek to understand how collaboration barriers and characteristics influence task performance among participants, alongside with who participants are more likely to collaborate with, and how this collaboration correlated with participants' performance in the contests. In this sense, we present a collaboration barriers' model that would be influential to participants' task performance. Besides that, we show which participants are more productive and win the challenges through the collaboration patterns found in SW CS contests.

We claim that a major issue within the SW CS strategy is not providing potential collaboration ways to support the collective performance among participants during the software competition.

1.3. Main Contributions

In this work, we discuss the barriers and characteristics of collaboration brought by the new phenomenon in a contemporary software development context - competitive SW CS. In this new work setting, specifically in the work of software development through online systems (platforms) that allow the broadcasting/ distribution of "open" and "crowd-enabled" calls, there is a need for a clearer and deeper understanding of how and who the participants

who collaborate are, as well as how collaboration impacts on productivity and quality solutions submitted during SW CS contests.

By answering the proposed research questions, this thesis presents the following main contributions. First, it provides an empirically grounded understanding of collaboration characteristics and the barriers faced by crowd workers during SW CS task fulfillment. Moreover, another contribution of this dissertation is a discussion on the three involved parties in the SW CS context – requester, crowd, and platform, which helps to bring up the collaboration aspects that influence this contemporary software development strategy. Among such aspects, communication (as expected) and coordination are the most important ones. In other words, this thesis illustrates how the crowd works, and submits their solutions to the platform. A third contribution comprises how crowd workers collaborate to improve their productivity, winning the challenges they are taking part of, which is demonstrated through an analytical framework of communication patterns. The fourth contribution is by means of insights about crowd competitors, conditions, processes, and outcomes of collaboration to both research on and design of SW CS contests. In order to do so, we present an approach for identifying dependencies/correlations between better communication and productivity in development software contests among crowd workers, that is, an approach for the requesters who demand SW CS tasks, for the crowd, who delivers/solves solutions to the tasks, and, finally, for the platforms, which intermediates this market.

In this sense, we investigated to what extent SW CS is a collaborative software development strategy, which barriers crowd workers face when performing a task in a SW CS contest, and how communication is necessary to support crowd workers in boosting submission rates and winning SW CS challenges.

The platforms can benefit from the communication patterns presented in this thesis in various ways. Once collaboration and their characteristics between submission and winning ratio are related to these communication patterns, it becomes possible to examine the actual communication over forum challenges and, from this measure, identify potential submitter and winners. The platform could make the decision to elaborate alternative competitions and, take steps to facilitate more suitable flows of communication.

These results suggest that communication strongly influences the crowd's productivity, besides impacting on the amount of crowd workers who registered for the task but did not submit their solution. Thus, we shall say that there is an unrealized potential for more collaboration in competitive SW CS.

This thesis contains main novel contributions, which map into the research questions presented in Chapter 1. and, are related to the artifacts generated in research phases I and II, as presented in Section 1.1. Therefore, our mentioned contributions are:

- An empirical identification and modeling of collaboration barriers in competitive SW CS;
- Understanding crowd workers' communication behavior, assisting platforms to better design software crowdsourcing development challenges;
- An analytical framework of communication patterns that was used to analyze and understand the impact on crowd workers' productivity in competitive SW CS.

1.4. Other Results

This research resulted in scientific publications, undergraduate diploma theses, and project funded by national agencies. In the following subsections, we present these results.

1.4.1 Published Papers

During the PhD program, we published papers related to the topic of this thesis. So far, the main results were published in papers at emergent SW CS events, namely, ICSE, FSE, ICEIS, and IEEE Software. The summarized references for the papers originated from this research are presented as follows:

2018

Melo, R.R.M.; Machado. L.; Prikladnicki. R.; Souza, C.R.B. “Um Estudo Qualitativo sobre o Crowdsourcing: Análise da Colaboração na plataforma TopCoder”. In: XXI Congresso Ibero-Americano em Engenharia de Software (CibSE), 2018. (to appear)

Zanatta, A.L; Machado. L.; Steinmacher, I. “Competence, Collaboration and Time Management: Barriers and Recommendations for Crowdworkers”. In: 5th International Workshop on CrowdSourcing in Software Engineering (CSI-SE). Collocated with the 40 International Conference on Software Engineering (ICSE), 2018. (to appear)

2017

Zanatta, A. L.; Steinmacher, I.; Machado, L. S.; Souza, C.; Prikladnicki, R. “Barriers Faced by Newcomers to Software-Crowdsourcing Projects”. *IEEE Software*, vol. 34, 2017, pp. 37-43.

Machado, L. S.; Zanatta, A. L.; Marczak, S.; Prikladnicki, R. “The Good, the Bad and the Ugly: An Onboard Journey in Software Crowdsourcing Competitive Model”. In: 4th International Workshop on CrowdSourcing in Software Engineering (CSI-SE). Collocated with the 39th International Conference on Software Engineering (ICSE), Buenos Aires, 2017, pp. 2-8.

2016

Zanatta, A. L.; Machado, L.; Pereira,G.; Prikladnicki, R.; Carmel, E. “Software Crowdsourcing Platforms”. *IEEE Software*, vol 33(6), 2016, pp.112-116.

Machado, L.; Kroll, J.; Marczak, S.; Prikladnicki, R. “Breaking Collaboration Barriers through Communication Practices in Software Crowdsourcing”. In: Global Software Engineering (ICGSE), 2016 IEEE 11th International Conference, pp. 44-48.

Machado, L., Kroll, J., Prikladnicki, R., de Souza, C. R. and Carmel, E. “Software Crowdsourcing Challenges in the Brazilian IT Industry”. In: 18th International Conference on Enterprise Information Systems (ICEIS), Rome, Italy, 2016, pp. 482-489.

Machado, L., Meneguzzi, F., Prikladnicki, R., Carmel, E.; de Souza, C. “Task Allocation for Crowdsourcing using AI Planning”. In: 3th International Workshop on *Crowdsourcing* in Software Engineering (CSI-SE). Collocated with the 38th International Conference on Software Engineering (ICSE), Austin, Texas, 2016, pp. 36-40.

2015

Machado, L.; Prikladnicki, R. “Software Crowdsourcing: Barriers Faced by the Crowd.” In: 2nd Latin-American School on Software Engineering (ELA-ES), UFRGS, 2015.

Machado, L.; Pereira, G.; Prikladnicki, R.; de Souza, C. R. B; Carmel, E. “Uma Visão sobre a Adoção do Crowdsourcing para Desenvolvimento de Software no Brasil. In I Workshop sobre Sistemas de *Crowdsourcing* (SCrowd) in conjunction with Congresso Brasileiro de Software: Teoria e Prática, Belo Horizonte, MG, 2015.

2014

Prikladnicki, R.; Machado, L.; Carmel, E.; de Souza, C. R. B. “Brazil Software Crowdsourcing: A First Step in a Multi-year Study”. In: 1st International Workshop on Crowdsourcing in Software Engineering (CSI-SE). Collocated with the 36th International Conference on Software Engineering (ICSE), Hyderabad, India, 2014, pp. 1-4.

Machado, L.; Pereira, G.; Prikladnicki, R.; Carmel, E.; de Souza, C. R. “Crowdsourcing in the Brazilian IT industry: what we know and what we don't know”. In: 1st International Workshop on Crowd-based Software Development Methods and Technologies (*CrowdSoft*). Collocated with the 22nd Foundations of Software Engineering (FSE), Hong Kong, 2014, pp.7-12.

1.5.1 Participations

ICSE 2017 - Student Volunteer

Poster Session – “Software *Crowdsourcing* in Brazil IT Industry”. Collective Intelligence Conference. Santa Clara, CA, 2015.

Authors: Leticia Machado, Rafael Prikladnicki, Cleidson Souza e Erran Carmel

Poster Session – “Software Crowdsourcing: A Transformação da Indústria de Software”. Les Doctoriales Rio Grande do Sul. Bento Gonçalves, RS, 2015.

Authors: Leticia Machado e Rafael Prikladnicki

Poster Session - “Is Software Crowdsourcing a collaborative software development model?” SIGCHI Writing Workshop at IHC/SBSC 2015, Salvador, BH, 2015.

Authors: Leticia Machado, Sabrina Marczak, Rafael Prikladnicki e Igor Steinmacher

Poster Session – “Crowdsourcing: Software Industry transformation and disruption”. Warm Up Symposium for ICSE 2017. Co-located with Congresso Brasileiro de Software: Teoria e Prática (CBSOFT), Maceió, AL, 2014.

Author: Leticia Santos Machado

1.5.2 Advising

Graduation Thesis 1. Marcos Cezar Szczepanik. “Desafios para o Gerente de Projetos em Ambiente de Software Crowdsourcing”, 2017. Trabalho de Conclusão de Curso - Universidade do Vale do Rio dos Sinos. Advisor: Leticia Santos Machado.

Graduation Thesis 2. Felipe Amadeus Junges. “Software Crowdsourcing: Um estudo sob a perspectiva da multidão”, 2016. Trabalho de Conclusão de Curso - Universidade do Vale do Rio dos Sinos. Advisor: Leticia Santos Machado.

Graduation Thesis 3. Sâmara Knorst. “Crowdtesting: Um estudo da caracterização das plataformas de teste para a multidão”, 2014. Trabalho de Conclusão de Curso - Universidade do Vale do Rio dos Sinos. Advisor: Leticia Santos Machado.

1.6 Funding

- a) Scholarship granted by HPE Company
- b) Brazilian Science without Borders Program Project (PVE) – “Brazil Crowdsourcing: Software Industry transformation and disruption”

1.7 Collaboration, Research Visits, and Course Work

During this study, we had the opportunity to collaborate with professors from American University, Washington/DC and University of California, Irvine (UCI). Professor Erran Carmel, who is a recognized researcher in the area of globalization of technology work, which involves global teams and global sourcing, visited Brazil as a Visiting Professor of the Science Without Borders Program, which the author of this thesis was part of. The project on the study of the subject “*Crowdsourcing na Indústria de TI Brasileira*” involved the partnership between Federal University of Pará, American University and PUCRS during the period from 2015 to 2017.

We did a research visit to UCI to interact with Professor André van der Hoek and other researchers. During our time there, we mainly had the opportunity to design and carry

out an experimental study in the form of an online contest to understand how crowd collaboration can improve the quality of the results of a competitive software development task. This experiment was based on a previous one [LAT15], focused on software design, to assess the value of recombination approach.

Our collaboration and research visit to UCI contributed mainly to the follow aspects of this study:

- Acquiring a better understanding of the quantitative research method
- Reviewing the research design
- Partially executing the evaluatory phase

Furthermore, still during the academic course of this thesis, it was possible to develop a collaboration that resulted in the analysis and characterization of collaboration through messages obtained on TopCoder platform forum, carried out in partnership with then master's student Ricardo Marinho and his advisor, Prof. Dr. Cleidson R.B. de Souza.

Part of the obtained results were presented in Chapter 4 and served as input for the extension of the messages analysis of TopCoder's forum that resulted in the collaboration characteristics described in subsection 4.3 of this thesis.

Throughout the courses attended during the PhD program, the student carried out work related to Crowdsourcing in the fields of AI, Natural Language – Ontology, and Emerging Themes in Database.

2. THEORETICAL BACKGROUND

The theoretical background represents an important research step [YIN01], containing the main concepts and theories of the researched areas. In this chapter, we present the fundamental concepts on SW CS, competition model and, SW CS platforms.

2.1 Software Crowdsourcing

Crowdsourcing in software development derives from Crowdsourcing (CS) on the whole and keeps in its definition the act of engaging a global set of online workers [MAO15b], who contribute to providing software solutions or services on demand [PRI14].

Technology encourages unprecedented levels of collaboration among people from different backgrounds and farthest geographical locations, with online communities at the heart of CS providing the context and structure within which "work" takes place [HOW08], [ZHA14].

With the broad growth and accessibility of the Internet, driven by Web 2.0 [PEN14], [STO14], we can notice the emergence of online communities organized according to different areas of interest, thus, favoring CS activities [EST12], [BEG13]. The advantage of using global and heterogeneous resources by assigning a problem or task to the public rather than passing it on to a single company seems to be indisputable to most authors who study the area of Crowdsourcing [DOA11], [SCH11], [KIT13], [KAG13], [SAX13]. Thus, as in other application domains in which CS is used such as creative [KIT10], [BOU14], and innovation [DOA11], Software Engineering (SE) also seeks to employ the benefits of open collaboration.

In SW CS literature, several definitions of the term have been found according to different authors who study the adoption of CS for SE [WU13], [LAT13], [TAJ13], [STO14], [MAO15a]. The SW CS definition adopted in this research will be that of Mao et al. [MAO15b], that refers to the act of externally transfer any task of the software development process to a potential and undefined large group of online workers – the crowd, in an open call format.

2.1.1 SW CS Elements

In each area where CS is applied, including the area of SW CS, it has always been noticed the use of four main elements, defined as: (i) requesters (companies or individuals), who post the issues they want solved— as a freelance job or competition; (ii) the community of online workers – crowd participants, who have signed up for the platforms and then decide (as individuals or small teams) whether to take on a challenge (open call); (iii) the platform, that mediates and connects requester with online workers to solve tasks, and (iv) the task, a unit that has been fragmented to be assigned as the activity/problem to be solved. Based

on the view of Hosseini et al. [HOS14a], those four elements: crowd, requester, task, and platform, were considered as the main pillars of SW CS.

Considering the extremely distributed nature in SW CS, in [HOS14a] the authors recognized that the participants involved in SW CS projects have different characteristics as described. The central unit that involved participants - the software task, is also mentioned according to its features.

Table 1 presents the five distinct features from the Crowd.

Table 1 - Crowds' characteristics

Feature	Description
Diversity	It means the recruitment of different people within the crowd to accomplish a task. Such diversity can be divided in spatial diversity, different backgrounds and, competence.
Unknown-ness	Is the condition or fact of being anonymous. The crowd participating does not know the requester (crowdsourcer) and does not know other members.
Largeness	Refers the large potential number of the crowd participating in a crowdsourcing activity.
Undefined-ness	It means randomness selection procedures, without imposed borders to select a group of people.
Suitability	It means suiting a given propose, ocasion. Crowd suitability means the fit of the crowd for performing a crowdsourcing activity.

The requester's characterisrtics are show in **Table 2**.

Table 2 - Requester's characteristics

Feature	Description
Incentives	Referes to incentives provided as a kind of extrinsic or intrinsic motivation for the crowd as payment and peer recognition, respectively.
Open call	It means that the task is open to anyone (public) who is willing to try out an act.
Ethicality	It means conforming to moral standards, or to the standards of conduct of a given profession or group.
Privacy	It means that the requester should not disclose the crowd's personal and private information to other participants, other organizations and other entities.

Table 3 summarises the features of the platform according to [HOS14a].

Table 3 – Platform’s characteristics

Feature	Description
Crowd-related interactions	Refers to interactions such as enrolment, authentication, submission, feedback mechanisms, etc. provide by the CS platform between the crowd and the platform.
Crowdsources-related interactions	Refers to register, authentication, broadcast, negotiation, verification mechanisms, etc. provide by the Cs platform and crowdsourcer (requester).
Task-related facilities	Include aggregation results, storing history of completed tasks and threshold mechanisms for the quality and quantity of the obtained results.
Platform-related facilities	Include online environment, feasible interface, attractive and interact interface and payment mechanism.

Finally, Table 4 summarise the features of the crowdsourced task.

Table 4 – Task’s characteristics

Feature	Description
Modularity	Decompose complex tasks into a set of smaller tasks.
Complexity	Condition or quality of being complex or simple task.
Solvability	Capability to be solved for humans.
Automation characteristics	A task difficulty to automate or expensive to automate.
User-driven	Activity that the crowd should provide a solution to a particular problem.
Contribution type	Contribution of the crowd can be individual and can be a collaborative contribution.

Figure 2 represents the interaction flow among task, requester, platform, and crowd, in which:

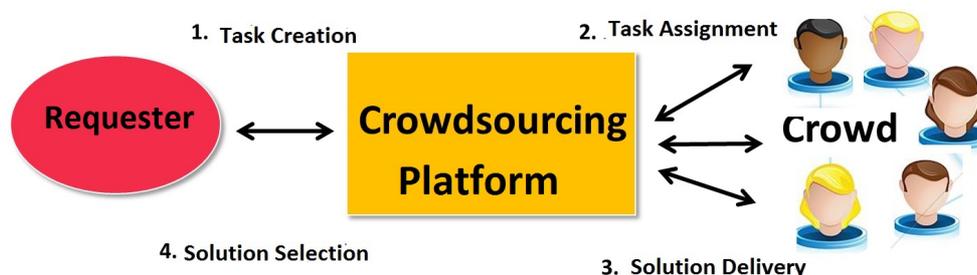


Figure 2 - SW CS involved parties

1. Posting task: it represents the disclosure of tasks submitted by the requester in a certain platform.

2. Selecting task: it refers to the selection and reception of tasks by the crowd. At this stage of interaction, the platform can play an important role in directing certain tasks to crowd members who meet the requirements of interest, competence, and experience that the task may require.
3. Submitting solutions: it characterizes the submission flow of solutions sent by the crowd. At this stage, validations can be made for the selection and aggregation of solutions that have met the specifications and quality criteria of the tasks initially defined by the requester.
4. Obtaining deliverables: it represents the moment of choosing the solutions submitted by the crowd. In this step, the consolidation of the obtained results can be reviewed by the requester, so that the task remuneration is accomplished, finishing the task.

2.1.2 SW CS characteristics

To assist in understanding SW CS characteristics related to other development strategies such as outsourcing, opensourcing, and innersourcing development, , extracted from [ÅGE15], is presented. Through the figure, the authors highlight SW CS characteristics considering whether the group knew each other or not, and in terms of professionals' payment format.

In Figure 3 a quadrant matrix has two dimensions. The first represented dimension is the participants' degree of knownness in a software project. In both strategies, *innersourcing* and *outsourcing*, the workforce is "known", that is, all project members know each other face-to-face or establish a virtual contact where each person's basic social and technical information (e.g., experience, origin, etc.) is known, and it is possible to have a certain guarantee of collaboration, although the degree of collaboration may vary. In the traditional outsourcing scenario, a customer clearly knows with whom they close a contract, the location and form of work that has been specified. In the innersource, developers will be known by their corporate ID (for example, corporate email address). In *opensourcing* and *crowdsourcing* strategies, this identification does not occur [SCH09]. In these approaches, developers are usually unknown.

From the software managers' point of view, utilizing external, unknown, uncontrollable crowd workers would put their projects under greater uncertainty and risk compared with in-house development [SAR17].

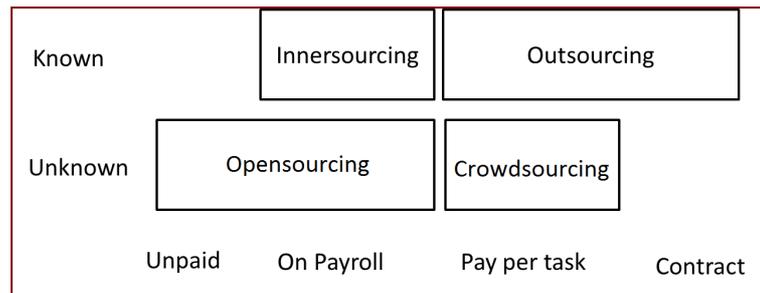


Figure 3 - Software development strategies
Font: [ÅGE15]

The second dimension of Figure 3 represents the payment format of software development strategies. Developers involved in an opensourcing context may or may not be paid. A project that will be carried out through the open source community is initially a project that could attract volunteer developers, similar to CS projects. At the same time, the organization that performs open source projects may still be involved in projects where its developers can be paid to keep them engaged in the project. In opensourcing, communities' knowledge is shared with a focus on developing better software and few or no attention to profitability. In general, opensourcing is more based on the voluntary aspect of participation, where the network of developers tends to be more stable, with long-term collaborative ties.

Regarding crowdsourcing developers' networks tend to be more competitive, short-term collaborative and driven by monetary participation [OLS13], the common motivation relationships between open source and crowdsourcing are based on the reputation that such activities can offer and flexibility in performing activities (usually carried out at home), to explore the self-interest about a certain subject or knowledge [OLS13].

Developers of an innersource project are, by definition, always paid, because they are employees of an organization. Developers in a "conventional" outsourcing context can also be paid "by task" (short-term), or they can be hired (long-term). In turn, crowdsourcing differs from outsourcing, because, besides developers not knowing each other, they will never be "hired" in a CS setting but will always be paid by task.

2.2 Application models in SW CS

According to the CS definition, the CS *open call* format to request a particular task, the involvement of unknown crowd participants, and the potential group of people that may be covered in these tasks distinguish CS from other software development strategies, as it was presented in subsection 2.1.2 of this document. According to [LAT16], by varying the aspects of how a complete task can be or not decomposed into smaller tasks, how crowd members collaborate, among others, a number of CS models have emerged and made the concept of collective intelligence and open innovation common in the performance of software development activities.

In [LAT16], peer production, microtasks, and competition are proposed as the three main CS models for SE. Such classification is based on the form of crowd participation in software projects.

Competitions: The competition model receives much attention in software development, since TopCoder platform, pioneer in SW CS tasks, offers to requesters access to diverse solutions in which it is possible to obtain results with high quality. At the same time, additional not foreseen costs in this model can arise due to specific knowledge of the tasks, the number of participants involved, and the quality of the solutions [STO14], [LAT16].

Microtask: The microtask model, in which work is partitioned and can be completed in a few minutes, is often used for a number of application domains because of its high scalability. In software development, the authors [LAT16] discuss the success of this model for testing tasks – crowd testing. There are a large number of platforms intended for testing (functional, interface, etc.), which will be presented in subsection 2.3. Since it becomes possible to quickly allocate specialized workforce, or not, to complete test tasks with greater coverage of devices, operating systems, among others; in less time, this model offers good opportunities for the market.

Peer production: The peer production model is a very widespread example in the open source community. It follows a model in which control is decentralized, where contributions and decisions of project scope and objectives are made by the members of the community themselves, motivated by a good cause or by the reputation they can achieve. In addition to OSS, other forms of peer production are currently being used, such as the Q&A site and StackOverflow³, where developers share knowledge through answering questions asked on the site [LAT16], [STO10].

2.3 Phases of software development and SW CS platforms

Innovation in modern crowdsourcing is in the platform itself— and the services it provides. These services include management and coordination of processes and people at both technical and business levels. For example, Topcoder [TOP17], Upwork⁴, and Crowdplat⁵ have tools that support project managers, team leads, and any other governance needs [ZAN16]. The requester creates a task and submits a request describing the main requirements, including instructions, constraints, acceptance criteria, and goals. The requester also defines the target audience, taking into account the crowd's abilities, and the task's duration, resulting in a document that goes through the platform. The platform assigns the task to the crowd. The crowd, in turn, participates and executes the request. At the end of the process, the requester validates the request and rewards the crowd on the basis of the accepted solutions.

Some general platforms, such as Amazon Mechanical Turk⁶ (AMT), Stack Overflow,

³ <https://www.stackoverflow.com>

⁴ <https://www.upwork.com/>

⁵ <https://crowdplat.com/>

⁶ <https://www.mturk.com/>

Innocentive⁷, and Freelancer⁸, are not specifically for software development, but they support it. Amazon Mechanical Turk is the main platform for microtasks, and it has been employed, for instance, to support GUI testing [KIT08], [KAU11]. Microtasks are self-contained units of work that are granular and might take a few seconds or minutes to complete, with correspondingly small payments. Stack Overflow is a question-and-answer portal that has been used to improve software documentation.

Multiple Development Phases - TopCoder is the world's largest software-crowdsourcing platform, which manages competitions in online algorithms as SRM (Single Round Matches), data science, development, and design competitions. The crowd often works on each phase separately as a competition. The standard phases include requirements' exploration (application specification), architecture, design, component, deployment, and testing. The software produced is licensed for profit by TopCoder, and the competitors involved in its creation receive royalties based on sales. Upwork is a giant platform with services ranging from graphic design to software development. CrowdPlat is a small startup. Its model differs somewhat from that of Topcoder once it connects requesters to either virtual freelance project teams or project teams from small consulting firms.

Requirements - Crowdsourcing can support the requirements' phase because the crowd could be potential users of software, designed to meet their own requirements. Requesters could harness the power of the crowd to understand its requirements as part of requirements' elicitation. The open source model has validated this approach. Some research, such as CrowdREquire [ADE12], has examined this phase in the SW CS tasks.

Design - Parsing design out to the crowd is challenging. Numerous platforms do design, although most of it is visual— from logos to webpages. DesignCrowd uses a competition model and freelance jobs to distribute projects. Requesters can send the crowd private messages and view a specific design during a competition. Evaluation of designers' performance is through reviews and positive or negative feedback. Requesters can offer second and third-place prizes or even pay designers just to compete as a guarantee. The biggest player in this niche is 99designs⁹. Competitions have four stages: the qualifying round, selecting the finalists, the final round, and selecting the winner.

Coding - This phase is covered largely by the general-purpose platforms we mentioned before. Basically, requesters describe their problem and post a task, and programmers bid or compete to provide solutions. We recently notice platforms here: Codeforces¹⁰, HackerEarth¹¹, HackerRank¹² and, Codewars¹³.

Testing - Crowdsourcing's immense power is evident in software testing. Here, a large crowd tests software, using many testing platforms in which the applications run on

⁷ <https://www.innocentive.com/>

⁸ <https://www.freelancer.com/>

⁹ <http://99designs.com>

¹⁰ <http://codeforces.com/>

¹¹ <https://www.hackerearth.com/>

¹² <https://www.hackerrank.com/>

¹³ <https://www.codewars.com/>

different devices, operational systems, browsers, and language versions. Testing can be quick, with ramp-up and ramp-down, in different environments and situations. Typically, requesters pay only for the valid bugs found. Some crowd-testing platforms have experience with diverse skill levels, minimum functionality, and security during operational execution. A crowd-testing platform must coordinate activities in the crowd, track work progress, guarantee task deadlines and quality, and ensure project confidentiality, safety, security, and terms and conditions. Some of the main crowdtesting platforms are uTest, Passbrains, BugFinders¹⁴, Testbirds¹⁵, and 99tests¹⁶.

For some software applications such as Kaggle¹⁷, a data science SW CS platform is one of the most representative examples [TAU17], [ZHO17].

SW CS taps global inputs in new ways. Crowdsourcing platforms have introduced dramatic innovations to software development, from competitions, to coder ratings, and massive crowd testing. These benefits come with some complications. The platforms have been evolving, so both requesters and workers must choose the best platform for a project. Additionally, sourcing some parts—or the whole—of a project increases both the need for coordination and the risks. Challenges include cross-task coordination, lack of communication, virtual-team organization, determining the target audience, integrating crowd’s deliverables, and ensuring software quality.

2.4 Opportunities in SW CS

The opportunities obtained for the adoption of SW CS are widely reported in several studies [STO14], [TAJ13] [LAT16]. These opportunities are: faster time-to-market, cost reduction, higher quality through broad participation, allocation *on demand* of resources to distribute tasks and, generating ideas and proposing unconventional solutions

Since the first three benefits mentioned above (cost, time and quality) directly address with well-known three central problematic areas of the Software Engineering, SW CS has the potential to become a common strategy to software development according [ÅGE15].

2.5 Challenges in SW CS

In crowdsourcing software projects, some development challenges are inherited from Distributed Software Development (DSD) [CAR01], and Open Source Software (OSS) [OLS13]. There is a consensus in the literature about the main challenges encountered in SW CS projects according to [MAO13], [LAT15], [PEN14], [LAT16], [MAO15a], [STO14], [LI15], [ÅGE15]. These challenges were organized through a framework presented in [STO14]. The set of six key areas with particular relevance in the context of SW CS are task

¹⁴ <https://www.bugfinders.com/>

¹⁵ <https://www.testbirds.com/>

¹⁶ <https://99tests.com/>

¹⁷ <https://www.kaggle.com/>

decomposition, coordination and communication planning and scheduling, quality assurance and, knowledge and intellectual property. This thesis is focused on the coordination and communication challenge in SW CS projects.

2.6 Collaborative software development

Software Development is rarely an individual activity. In general, it is a collaborative activity with the work of several professionals to design solutions and produce quality code. Members of a software development team need to coordinate their activities, plan new actions, make decisions, carry out planned activities, and also communicate to develop software [WHI10], [SOM11].

The word collaboration comes from the Latin words *com* (prefix together) and *laborare* (verb to work), It means that two or more individuals work jointly on an intellectual endeavor [OXF17].

Collaboration is a complex, multi-dimensional process characterized by constructs such as coordination, communication, meaning, relationships, trust, and structure, according to [KOT05]. Another definition of collaboration is given by Pimentel and Fuks, [PIM12] where communication, coordination, and cooperation appear together in action among two or more people.

In the Computer Supported Cooperative Work (CSCW) domain, design of systems for collaborative work is modeled sociability [LEE15]. They argue that concepts of “group” and “group work” can denote special types of cooperative relations characterized by shared responsibilities. For a long time, CSCW focused on small and homogenous groups to support cooperation, and it ignored or even dismissed the major challenges posed by the system design for collaborative work by a large and maybe undetermined number of participants. In [LEE15], the authors presented MoCA, a new conceptual mapping of collaborative work types, consisting of seven dimensions of coordinated action in order to cover the new characteristics in which collaboration is involved: synchronicity, physical distribution, scale, number of communities of practice, nascence (describing the creation and use of temporary and unstandardized artifacts), planned permanence (short-term or long-term), and turnover.

Software development work is inexorably a collaborative activity that is intertwined with individual activities [WHI07]. A collaborative activity refers to any activity that requires interaction with another person, for instance, informal meetings, hallway conversations, phone calls, and e-mail messages [GON11]. Individual activities do not require other developers’ participation (e.g., coding), whereas collaborative activities require more than one person to be involved (e.g., a meeting).

Thus, knowing that the collaboration is present in the process of software construction (local or distributed), it must be ensured that it occurs in a satisfactory way to successfully provide the developed software product or service. In the work of [OLS00], they pointed five factors that contribute to the success of a collaboratory computer-mediated science, usually

at a distance: the nature of work, common ground, collaboration, readiness, management, and technical readiness.

A SW CS project involves a decentralized and online developer's community who collaborate to produce or bring something new in a software product. The main characteristic to collaborate in SW CS is the communication (synchronous and asynchronous) using forums, mailing lists, specification's documentation, and concurrent versioning systems (CVS). Although each crowd developer work individually (isolate codebase and artifacts copies) to building their software solution they collaborate. The crowd makes indirect and direct collaboration through communication. Crowd's developers participate in the forum communication supported by major of SW CS platforms. We use the term indirect and passive collaboration to represent the group of crowd developers who only read and consume the text messages posted by other developers on forum during task execution. The direct and active collaboration represents the group who communicate with other crowds' developers on forum.

In this sense, collaboration is represented in this thesis as a kind of activity or action that occurs in a group within a specific context, and it involves in its core communication between two or more people. Communication is essential to collaborate occurring in planned or imprompty interaction [WHI07]. In the SW CS, collaboration and communication concepts are integrated once the communication among crowd and platform during each development challenge is strongly focus on the task (goals, requirements, restrictions, technology and so on) in terms of request for help, help answers, identified problem, and problem response.

2.6.1 Collaboration challenges in software development

Since collaboration is strongly associated with communication among those involved in the software development process, some characteristics of computer-mediated communication through which there is collaborative development are presented [HER02].

Communication media are usually classified in the classic time/space matrix, according to both the spatial dimension (collocated/distributed, i.e., where the interaction occurs), and the temporal dimension (synchronous/asynchronous, i.e., when the interaction occurs). Media can also be classified according to another dimension, 'richness', which can be defined as the media ability to convey a large amount of information in different forms.

Common Ground (CG) theory [STR07] posits that people should attempt to achieve common ground (i.e., mutual understanding) with techniques available in a communication medium that lead to the least collaborative effort. CG theory presents eight properties that a medium may impose as constraints on the grounding process: co-presence, visibility, audibility, contemporality, simultaneity, sequentiality, reviewability, and revisability. No medium has all the attributes at the same time. When some medium lacks one of these characteristics, it forces people to use alternative grounding techniques with different costs for the speaker, the receiver, or both. Participants in a face-to-face conversation usually establish common ground on the fly, as they have access to clues like facial expression,

gestures, and voice intonation. Instead, when participants communicate over media, the fewer clues they have, the harder to construct it is. As a consequence, people who do not share mutual knowledge largely benefit from using audio/video channels for completing collaborative tasks, whereas those who have an extensive preexisting common ground can communicate effectively also on lean media such as email.

The classic models of knowledge collaboration in groups give particular weight to the need for convergence. Convergence around a single goal, direction, criterion, process, or solution helps counterbalance the forces of divergence, allowing diverse ideas to be framed, analyzed, and coalesced into a single solution. In fluid online communities, convergence is still likely to exist during knowledge collaboration, but the convergence is likely to be temporary and incomplete, often implicit, and it is situated among subsets of actors in the community rather than the entire community [FAR11].

The challenges of working as a team increase under conditions where it is difficult for members to communicate and coordinate with each other and effectively manage their mutual dependencies [OSL00]. We refer to these more difficult conditions as “team coordination complexity.” Specifically, we are considering the challenges of team size and dispersion.

Compared to the DSD challenges at the time it arise, the changes that SW CS development are causing also have a major impact [KAG13], [JOH11], [KIT13], not only on the market itself, but on the way, products are being created, modeled, built, tested, and delivered to requesters [KAZ10]. Moreover, the authors [STO14], [AGE15], [WU13], [TAJ13], and [LAT16] point out that working with SW CS (to all involved parties: crowd, requester and platform), is one of the greatest challenges that the current development environment presents, from the point of view of the nature of the model, where some issues stand out such as decomposition and allocation of software tasks, communication and coordination among those involved, motivation and compensation of tasks, and quality assurance of solutions submitted by the crowd.

The barriers that had affected DSD development in the past decade were basically influenced by the geographic and temporal distance that the teams involved faced on their different sites. Other barriers imposed by distance can also be cited: language and culture distance, organization, process and management issues, reduced informal communication, infrastructure and product architecture complexity, and trust. These challenges have strongly impacted collaboration between teams in global and distributed environments [CAR01].

2.7 Competition Software Development

Competition software development, as already mentioned in subsection 2.2, is a CS model that operates around online software platforms. The platform supports the independent software development challenges (by decomposing into multiple sub-tasks) in which the crowd developers compete with each other by monetary rewards [MAO15b].

TopCoder is one of the main platforms for competitive SW CS projects worldwide, created in 2001 [ARC10], [LAK10]. It covers all phases of the software development lifecycle, from elicitation to deployment, in which an open call/open contest to the crowd is made for each of these phases.

TopCoder has over 1 million workers from over 190 countries, averaging 30 thousand logins every 90 days, 7 thousand challenges hosted per year, and 70 million dollars in challenge payouts. In this sense, TopCoder has been investigated by several empirical studies in the literature, [STO14], [MAO13], [ARC10], [LAK10], [MAC17], [ZAN16], [YAN16], [YAN17].

Each development task is organized as a challenge through open competition. Designed to enable wide task accessibility and self-selection, TopCoder platform allows crowd developers to freely choose to engage tasks based on their personal skills, experience, and interests [YAN16], [SAR15].

A larger pool of potential developers may lead to a significant increase in registrations and participation in competitions on TopCoder. The growth of the pool of available developers appears to have dramatically increased over TopCoder's history, from 50,000 members in 2004 to over one million members in 2017.

Due to the perception that the crowd with expertise requisite is very large, this brings implications for requesters/companies who are seeking increased speed of software development through crowdsourcing [STO17]. However, an important issue in competitive SW CS is that worker decisions are highly volatile from task registration to task submission. In several research findings, there is a negative effect on the crowd's submission/ interest in competitions [STO14], [YAN15], [YAN16], [SAR17]. The negative effect is the difference in number of those workers who only register for a task, but do not submit their solution.

The open call is composed of six phases [MAO15a], [MAO15b], [LI15], [LAK10] as illustrated in **Figure 4** and described as follow.

- **Posting:** the disclosure of a task happens through its publication on the TopCoder's website (posting phase). The basic information contains an overview of the task, desired technologies for performing the task, the award value, and the deadline for solution submission. Its duration includes two important dates: the deadline for task registration, and the deadline for task submission.
- **Registration:** all developers who are interested in participating must announce their decision publicly through their registration in the competition. This means that online developers can observe information from opponent developers, including performance history and skill ranking, including the information exchanged in the task forum. The registration phase is usually available for a few days. Subscribers can obtain detailed documents about the task and participate in the forum's communication channel supported by the platform for each task.
- **Submission:** registered members are asked to submit their solutions until the submission deadline.

- **Review:** All submitted solutions are collected by the platform, so that a community review board shall perform a thorough review based on scorecards (pre-defined evaluation criteria for each challenge). Once the evaluation process is finished, the coders are individually notified on their submission results. The reviewed evaluations during the review phase are announced, and the classified competitors are ranked according to the average score given by the reviewer of the solutions for each task. The award structure and reputation mechanisms among members of the TopCoder community are mainly based on the analysis of the evaluation received by the fulfillment of the delivery quality requirements in the competitions.
- **Appealing:** competitors get a chance to appeal the decision made by the reviewers after submissions have been evaluated.
- **Winner:** once all appeals have been resolved, the placement is determined by the average score, and the winner(s) is/are announced.

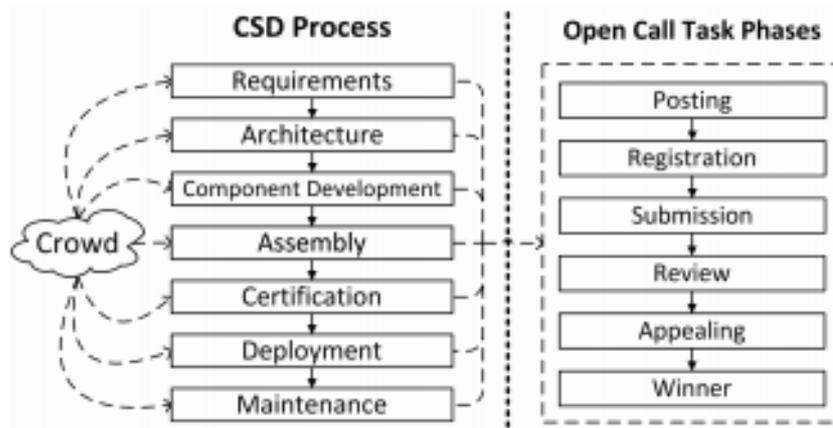


Figure 4 – TopCoder phase
Fonte: [MAO15a]

Thus, to each phase of SW CS process such as Requirements, Architecture, Component development, Assembly, Certification, Deployment and, Maintenance (Figure **Figure 4**), TopCoder associates the phases of the open call task, previously described.

On TopCoder, there is a mediator between the platform and the crowd, named copilot, to follow up on task development. The copilot may be a member of the crowd playing the representative role of the requester who demanded the task. The copilot's main responsibilities are related to managing questions submitted by crowd members in task-based forums, and answering them, updating information, and so on. It is possible for task participants to open new threads within the forum and reply to messages from anyone (crowd and copilot) who is active in the forum.

The platform offers services expressed in the form of tasks organized into challenge categories, as follows: Design Challenge, Development Challenge, Data Science Challenge, and Competitive Programming.

A task in the Development Challenge category is categorized in one of the following sub-categories: Architecture, Assembly, Bug hunt, Code, Concept, Content creation,

Copilot, Component design, Component development, First to Finish (F2F), Marathon, RIA, Spec, Test scenarios, Test Suites, or UI prototype. Despite the category, each task has a scope and is composed of technical requirements that define the expected behavior of that software task, and the necessary interfaces to integrate with other parts of the system. For the development challenge-based tasks, TopCoder has started making available UML tools, such as sequence, class, use case, and activity diagrams.

Figure 5 illustrates examples of the open challenges for the “Development” category. Each challenge has its challenge title, technologies, prize value, phase, phase deadline, number of subscribers, number of submissions, and deadline.

The screenshot shows the TopCoder website interface. At the top, there are navigation tabs for 'COMPETE', 'LEARN', and 'COMMUNITY'. The user is logged in as 'leticiars'. The main section is titled 'Challenges' and includes a search bar and filters for 'Design', 'Development', and 'Data Science'. The 'Development' filter is selected. The main content is divided into two sections: 'My Challenges' and 'Open for registration'. The 'My Challenges' section shows a challenge titled 'Heroku Compliance App - Frontend' with a prize of \$1,500, 44 subscribers, and 1 submission. The 'Open for registration' section shows two challenges: 'Counting Objects in Images' with a prize of \$10,000, 33 subscribers, and 0 submissions, and 'Slack bots series - Slash Command Accelerator' with a prize of \$1,500, 40 subscribers, and 0 submissions. A sidebar on the right shows a summary of all challenges and filters.

Challenge Title	Prize	Subscribers	Submissions	Phase	Deadline
Heroku Compliance App - Frontend	\$1,500	44	1	Submission	1d 9h to go
Counting Objects in Images	\$10,000	33	0	Open for registration	6d 15h to register
Slack bots series - Slash Command Accelerator	\$1,500	40	0	Open for registration	3d 13h to register

Figure 5 - TopCoder Screen – List of challenges

The registration deadline specifies the time by which all individuals willing to participate must register for the competition, which is usually two or three days after the competition posting date. The submission deadline specifies the time by which all solutions must be submitted, which is usually within five to seven days' interval after the competition posting date. Every competition has associated payment that is given to the competition winner, and, in some cases, 50% of this amount is given to the first runner-up. it

The registration information is public, so that competitors can see who else has registered. This is achieved by clicking on items in the “Registrants” tab. The result may look like what is shown in Figure 6. Moreover, information is updated instantly: as soon as one competitor has registered for the contest, others can see it.

The screenshot shows the registration details for a challenge titled "Heroku Compliance App - Frontend". At the top, there are tags for "Code", "TC018", "Angular 2+", "Heroku", "ReactJS", and "Other". Below this, the "PRIZES" section shows a 1st place prize of \$1,000 and a 2nd place prize of \$500. To the right of the prizes are two buttons: "Unregister" (red) and "Submit" (blue). A dark grey bar indicates the "Next Deadline: Submission" is "1d 9h until current deadline ends", with a "Hide Deadlines" link. Below this is a progress bar with five stages: "Started" (Jan 08, 02:01), "Registration" (Jan 12, 02:01), "Submission" (Jan 12, 02:01), "Review" (Jan 14, 02:01), and "Winners" (Jan 15, 14:01). The time zone is listed as "America/Los_Angeles". At the bottom, there are navigation links: "DETAILS", "REGISTRANTS (43)", "SUBMISSIONS (2)", and "CHALLENGE FORUM". A "Challenge Overview" section is partially visible at the bottom, with a "Support" button.

Figure 6 - TopCoder screen – registration screen details

2.7.1 Forum Communication

During TopCoder's development challenge, the communication flow is established basically by online forums, and interaction frequently happens between Platform and Crowd workers, and Platform (crowd specialist – copilot) and Requester (in-house specialist), as shown in the Figure 7.

According to Topcoder phases, crowd participants who are interested in the challenge need to subscribe, and they will then have access to the complete description of the task, documents, artifacts, etc., besides accessing the task's specific communication forum. All communication happens via online forum. The platform sends emails only to notify about the start and finish of the phases after task's submission (review and appeals).

The Requester communicates with the platform (TopCoder) in order to plan (schedule, budget, etc.), post tasks, select solutions, and pay the winner. On the other hand, the platform communicates with the requester to coordinate changes and confirm task requirements' details.

The interaction among the platform and the crowd (attending the challenge) occurs to monitor task performance and submission, and to give feedback via the copilot. The challenges that present high technical and interface complexity can count on the presence of two platform copilots to mediate while the competition is occurring. In this flow, the platform acts as a requester's representative. The interaction between the crowd and the platform (copilot), in turn, may happen (through opening a new thread on the forum or

through a reply mechanism) to exchange information about different topics during the competition, as it will be detailed in Chapter 4.

However, the interaction among crowd workers in the same challenge (acting as competitors) can only happen through replying (some of them send the message using @nick as an id).

The interaction between the requester and the crowd and vice versa does not happen most of the time and it was represented indirectly through documents of task specification.

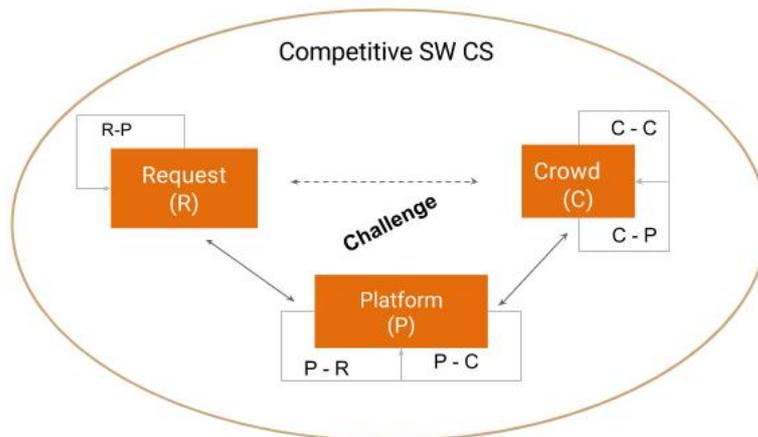


Figure 7 - Communication flow among involved participants

Figure 8 represents a view of the forum threads. It is possible to observe that the platform shows information on the number of views of messages posted on the forum, that is, this can be an indicator of the forum's audience. TopCoder platform has used as practice in development challenges to address two main threads for code questions and code documents.

Thread	Author	Replies	Views	Last Post
Submission format	shouvikroy	0	4	Jan 10, 2018 at 2:30 PM EST
endpoints	Sky_	2	30	Jan 9, 2018 at 3:08 PM EST
Heroku	evilkYRO1965	1	38	Jan 9, 2018 at 8:00 AM EST
Heroku deployment ?	zsudraco	3	64	Jan 8, 2018 at 6:07 PM EST

Figure 8 - Challenge's Forum

There is little research considering that contest communities both promote and benefit from simultaneous collaboration and competition, and that both types of relationships need

to be emphasized at the same time. Some studies such as [GNY11], CS [TAJ13], [NAG12] argued that the concept of “co-opetition” might be relevant for innovation’s success on the individual level within contest communities.

Hutter et al. [HUT11], introduce the term ‘communitation’– community-based collaboration among competing contest participants – to refer to the phenomenon of co-opetition in contest communities. The concept includes elements of competitive participation without disabling the climate for co-operation.

[HUT11] and [BUL10] showed how competition and collaboration as extreme poles of individual behavior might occur at the same time in competitive co-creation and community-based innovation contests, similarly to the concept of co-opetition on the firm level [TAJ13], [NAG12]. Interestingly, Bullinger et al. [BUL10], showed 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. They also found a higher probability for so-called communitators to be ranked high by the community evaluation and, claimed a positive correlation between both competitive and collaborative behavior, improving the quality of submission ideas and allowing the future potential of an idea to shine through the so-called ‘wisdom of the crowd’ [HOW06].

These studies addressed the collaborative competition or competitive collaboration by community-based innovation and creative contests.

In software crowdsourcing, only a few studies have focused on collaboration and competitive behavior as a characteristic that would influence in the rate of task’s submission, and the quality of solutions in SW CS market. Most of them focused on the influence of task pricing [MAO13], inappropriate task-worker matching [STO14a], and imperfect information of the crowd about the task’s goal and affordability [YAN15], [YAN16].

One of the pioneering studies that discusses collaborative competition through the phases and the process adopted by TopCoder platform was carried out by Nag et al. [NAG12], [NAG13]. In the authors' view, TopCoder uses competition and collaboration hybrid that organizes individuals to work towards a common goal through financial incentives and reputation. Collaboration allows these individuals to achieve larger goals together; however, they point out that development through competition requires a careful balance between competition and collaboration so that those goals are achieved and, finding ways to bring collaboration to the competitive model without losing the benefits of competition.

In addition, most studies are from the perspective of task requesters or crowdsourcing platforms [STO14a], [MAO13], [YAN15], and there is lack of research on qualitative analysis from individual crowd workers, and collaboration under the crowd workers’ perspective. Other studies investigating the crowd perspective have used transactional data available via the TopCoder platform’s API to generate statistics and predictions on crowd behavior. To our knowledge, no other study has used qualitative data of the platform, through interviews with platform members. This thesis also presents, for the first time, the content analysis of the forum repository that the platform uses during the competition phases. Therefore, once the way how collaboration between the crowd and the platform predicts success in

submissions has not been systematically investigated in the literature, this thesis aims to address such gap.

The present thesis differs from related work by considering which collaboration challenges are present during the task performance of the crowd workers in the specific context of competitive SW CS. We have been exploring how communication and collaboration among coders during the challenges can improve the quality and quantity of submissions.

To do so, it was based on the standards of messages that are sent on the task forums during the competition progress.

From the winning submissions, we could relate the variable communication and non-communication, in which coders who communicated better, by using a set of communication messages over challenge timeframe time, were those who won the challenges by becoming more productive and achieving higher performance.

3. COLLABORATION BARRIERS IN SW CS

3.1 Exploratory Phase

To identify and organize the collaboration barriers faced by the involved elements (requester, crowd and platform) in SW CS projects, we conducted a qualitative study relying on three different sources: semi structured interviews, literature review and, an exploratory case study on TopCoder platform. To analyze the data, we used Grounded Theory coding techniques [STR07], which are increasingly used to study the human aspects of Software Engineering [HOD10].

We analyzed the data separately and after, we integrated the results from three exploratory studies adopting a unique set of categories and topics [STR07].

We present our results in an integrated way – Collaboration barriers model in SW CS. The model was built to organize the identified collaboration barriers, mapping and categorizing them as a taxonomy of collaboration barriers faced by crowd workers in SW CS projects.

In these studies, we observed that the competitive nature in SW CS poses as an inhibitor of collaboration between the involved parties, especially for crowd workers. Although it may seem unnecessary to co-operate in an environment where each crowd worker competes with each other, collaboration may becomes important under two points: the first one refers to the crowd's own engagement in completing SW CS challenges, thus, becoming eligible for the best solution award, and the second point is directly related to the requesters, who expects to receive solutions to their problem, and the platform that, in turn, needs to keep the supplier and provider to its business.

In the following sections, we detail the methods used to conduct the interview field study, the literature review, and the exploratory case study to collect data from the crowd, platform, and requester, followed by details about the data analysis.

3.2 Semi structured interview from three SW CS elements

3.2.1 Settings and methodology

During Phase 1 from research method illustrated in Figure 1, the opportunity was taken to visit a pioneering researcher in the area of global software development (follow the sun model), Professor Erran Carmel (from American University), to conduct an exploratory field study based on semi structured interviews with companies, researchers, and IT professionals to better understand how crowdsourcing software has been adopted in the Brazilian IT industry. The visit was part of the activities planned in the “Special Visiting Professor” project of the Science without Borders program supported by the Brazilian

government. The project also has the partnership of Professors Rafael Prikladnicki (PUCRS), and Cleidson de Souza (UFPA), in addition to the researcher and author of this thesis.

In order to identify the candidates for the first stage of interviews, professionals and IT organizations were allocated at TECNOPUC (Technology Park of PUCRS), and other regions of Brazil (SP, MG, and RJ). Contact was made via LinkedIn and research collaborators' recommendations.

In the second stage of interviews, in order to compare the initial results on the adoption of CS in the IT market, the data collected was gathered in national and international scientific congresses between 2014 and 2015. We interviewed participants during the Brazilian Conference on Software: Theory and Practice (CBSOFT 2014 in Maceió, and CBSOFT 2015 in Minas Gerais), during the I Workshop on CrowdSourcing in Software Engineering (ICSE 2014 in India); and during the Collective Intelligence Conference in Santa Clara / CA, 2105.

We conducted a total of 20 interviews, in which the majority of interviewees were male (18), and the other female (2). Thirteen participants were from the industry, and 7 participants were from the academic field, as shown in **Figure 9**. They have had different work experience, including IT managers, media and Internet corporations' members, Brazil's CS platform CEOs, and academic researchers. Participants have had at least three years of working experience in average.

The participants are classified under three different CS elements: requester, platform, and crowd [MAC14]. We interviewed participants from two pioneer Brazilian CS platforms – Crowdtest¹⁸ and WeDoLogos¹⁹, which represented, at that time, the two largest crowd testers and crowd designers in Latin America.

¹⁸ <http://www.base2.com.br/en/crowdtest/>

¹⁹ <https://www.wedologos.com.br/>

Participant #	Job Title	Type of experience (Academic or Industry or Both)	Element
P1	Tester	Industry	Crowd
P2	Tester	Industry	Crowd
P3	Tester	Industry	Crowd
P4	Tester	Academic	Crowd
P5	Tester	Academic	Crowd
P6	Tester	Academic	Crowd
P7	Developer	Industry	Crowd
P8	Developer	Both	Crowd
P9	Assistant Professor (System analyst)	Academic	Crowd
P10	Assistant Professor (System analyst)	Academic	Crowd
P11	Assistant Professor (System analyst)	Academic	Crowd
P12	Developer	Both	Requester
P13	Manager (IT Consultant)	Industry	Requester
P14	IT Manager (Host Service Company)	Industry	Requester
P15	IT Manager (E-Commerce Company)	Industry	Requester
P16	Manager (IT Company)	Industry	Requester
P17	Manager (Media Company)	Industry	Requester
P18	CEO (Innovation Consultant)	Industry	Requester
P19	CEO (WeDoLogos)	Industry	Platform
P20	CEO (Crowdtest.me)	Industry	Platform

Figure 9 - Participant's information

Semi-structured interviews were conducted firstly with Brazilians and, following, with global practitioners from different countries (India, USA, Europe) with the intent of understanding the main challenges, opportunities, and any other relevant phenomena to adopt CS in the IT industry. Questions were designed to encourage the participants to talk about their experiences in SW CS under different aspects: organizational motivations for leveraging the crowd, impacts on the organization, experience participating in this new market (as suppliers and requesters), professional and business aspects, measuring success, and so on. In order to guarantee that the same topics of interest were addressed in different interviews, we used an interview guide, which was, to some extent, used in both national and international interviews to guarantee that similar issues were addressed. We present the interview guide in Appendix A.

Each interview lasted between 30 and 60 minutes. We conducted the interviews face-to-face, by voice or video conference call, and by email. Some conversations were not audio recorded because of companies' confidentiality issues.

3.2.2 Data Analysis

Our data analysis was guided by techniques associated with less procedural versions of the grounded theory (GT). Specifically, we applied the techniques of coding and constant comparison as recommended by Corbin and Strauss [STR07]. These techniques helped us to elicit emergent themes in the Brazilian IT industry, to identify concepts in the collected data and to link these concepts to higher level categories.

GT does not require a prior theory about the data, that is, a set of hypotheses to be tested. Instead, grounded theory's goal is precisely to generate theory grounded exclusively on existing data. In other words, it aims to develop a theory or explanation about what is going on in the field, or more specifically, what is available in the collected data. In open coding, we analyzed the interview notes manually (line-by-line) to identify categories. Categories grouped concepts together under a more abstract high-order concept to explain what is happening [STR08]. The categories were defined at a level of detail that could allow us to understand which enablers and blockers' factors were, based on three perspectives: crowd, requesters, and platform.

An enabler factor means a characteristic that promotes or motivates SW CS practice. On the other hand, a blocker factor means a characteristic that inhibits or limits SW CS practice. Table 5 shows these factors.

3.2.3 Results

Our findings showed that, even though the CS area in software development was still incipient in Brazil, the international IT market also shares most challenges and opportunities that SW CS development presents.

As enablers, there is the collective intelligence of the software engineering industry with more diversity, creativity, and knowledge sharing. Scalability, cost reduction, and time-to-market are also important enablers for platforms and requesters. Based on the crowd's perspective, the main opportunities pointed out were knowledge exchange and being up-to-date with the new technologies, besides the possibility to earn extra money [MAC16a].

Table 5 - Enablers and blockers' aspects in SW CS

Actors	Enablers	Blockers
Crowd	Extra money	Poor feedback
	Shared knowledge	Few collaboration
	Curiosity	Scarce context project information
	Free time	Unavailability of documentation
Requester	Save money and time	Low quality of services
	Creativity	Maturity of suppliers (crowd)
	New ways to do the same	Identifying a specific process
Platform	Fast delivery	Data confidentiality
	Reduced cost	Very specific business rules
	Diverse types of testing	Laws and taxes involved

As blockers, the requesters pointed out the low quality of services, the difficulty to identify a specific process to distribute tasks to the crowd, the maturity and, adoption of CS. Users are not familiar with CS processes. Participants from the requester's and crowd's groups emphasized limitations to adopt SW CS initiatives in terms of collectively coordination, communication and collaboration:

“Brazil is a very conservative country and it needs to prepare people to work with crowdsourcing. It is necessary to have a strong process behind the platform and people to support business.” – Participant 13

“Crowdsourcing is difficult in our company because it is necessary to have a visibility of tasks (progress activities, for instance “to do”, “doing” and “done”). We have a strong work process orientated on quality and productivity. Crowdsourcing could be a new direction to the future but it requires a maturity level and another mindset.” – Participant 16

Considering the crowd's side, it was observed challenges related to lack of communication in terms of little collaboration with other crowd workers, poor feedback, and task information issues.

“Disappointed with the other testers of the CS platform because it isn't a “real” community and collaboration just to peer production.” – Participant 1

Some cultural aspects in SW CS are presented in terms of language and participants' diversity of experience, practices, and background. While communication in international platforms happens in English, populations who do not speak other languages face a communication barrier.

“Brazil is a conservative country in terms of distributed software development. Usually, Brazil receives a lot of outsourcing demands but it is not used to outsource. The most of the time, Brazil is much more a supplier than a consumer.” – Participant 19

“The country is a special case in terms of software development. Brazilians companies and labor participate in global CS marketplaces, but it also “plays in their own sandbox. It may happen because of language issues.” – Participant 18

3.2.4 Discussion

CS is an emerging topic in software industry. It provides a new approach for companies involve their workers with innovation activities. However, despite the positive effects, many challenges are identified for CS practice. The Brazilian IT industry has specific challenges that make this country different from others. Even with some differences, we

found that the main challenges faced by practitioners and companies are concentrate in three areas Tasks, Processes, and People. We also found ten enablers and blockers factors for the CS practice in Brazil and other countries as mentioned in **Table 5**.

Tasks are difficult to manage in CS environments. Requesters expect to receive a task with certain level of quality. However, in some cases the delivery does not attend the expectations of the requester. The factors of quality for CS tasks are the number of participants, tasks assignment to workers according to their individual expertise, and the reward amount [LI13]. The inappropriate worker-task matching may harm the quality of the software deliverables [MAO15a]. On the other hand, workers report the lack of information that can result in a task delivered with low quality. When workers understand what information is need for the task specification, it will be possible to provide solutions to problems that meet customers' needs. However, to Mao et al. [MAO13] the vendor selection has a direct correlation with the quality of an outcome. Workers are attracted by an open call format rather than being selected. It encourages the non-skilled workers to participate. The list of countries with higher level of active members showed Brazil on 16^o ranking position²⁰ between 50 countries in 2017.

The lack of processes definition is another challenge faced by IT industry. To take advantage the power of software CS, it is important to define the proprieties, elements, responsibilities and interaction flows of software CS as a new software development process [KIT13], [HOS14a]. While other countries like United States adopt and invest in crowdsourced development processes, Brazil adopts a timid posture regards to it. Brazil has only two CS platforms to support software activities. We believe that online markets for software CS tasks such as software project development activities, still have not received attention from companies and workers. In literature, few authors explore region-specific practices in CS software project.

Europe and United States are well populated with CS participants, but that still does not say much about potential differences in acceptance of CS across the globe. In our study, some cultural aspects in CS are presented. Brazilians are highly creative in their own way, but the country is still underdeveloped in terms of software CS. Cross-cultural differences in the adoption of CS and open approaches to business are still under-explored. Every country is unique and has its own specific challenges when it comes to change the way of work, like implementing software CS. This study gives a starting point on region-specific practices in crowdsourced software development.

²⁰ https://community.topcoder.com/stat?c=country_avg_rating

3.3 Literature review from three SW CS elements

3.3.1 Settings and methodology

A literature review was used to explore current research in SW CS area to support our research methodology according Figure 1.

As the SW CS literature, focuses on aspects such as decompose tasks [LAT14], planning and scheduling [MAO15a], [MAC16b], process concerns [YAN16], motivation [MAO13], quality assurance [WU13] and coordination [STO14], several other forces can influence in the success of SW CS projects, and these studies neglect focusing on the level of collaboration barriers of the parties involved in SW CS projects. Counter examples are [PEN14] [MAC16c], and [MAC17], which explicitly focus on studying collaboration barriers that influence the success of SW CS projects among involved parties in terms of productivity, task fulfillment, and receiving solutions from the problems demanded by the requester and the platform.

Considering the findings from the initial study, in which SW CS is a very particular approach of designing and creating software and, recognizing that there is a gap in the current literature on how crowdsourcing platforms affect modern collaborative software development among the involved parties (i.e., the crowd, requesters, and platforms), we conducted a Literature Review (LR) to identify possible collaboration issues in SW CS projects [MAC16c]. The goal was to list the barriers evidenced by the different studies in a collaboration barrier model. Since we were interested in gathering studies related to software development, we restricted the literature review to the SW CS domain. The following question guided our LR.

What are the barriers to collaboration in SW CS projects?

By answering this question, we aim to capture the existent evidence on barriers to collaboration in SW CS. “Barriers” are the forces that present obstacles to the involved requester and crowd workers who perform a SW CS task in an online platform. These forces can inhibit communication and coordination in terms of orchestrating highly distributed crowd workers and promoting a period to share insights and experiences by the crowd to ensure the success of SW CS projects for all the involved parties. We make no distinction regarding the process phase of SW CS (i.e. posting, submission, screening/review, and winner), or the involved parties (requester, platform, and crowd).

3.3.2 Data Analysis

We split our LR into two steps. In the first one, we conducted a qualitative mapping analysis in an already consolidated systematic literature review report about SW CS [MAO15b]. Second, we organized a snowballing review based upon guidelines established by [WOH13] and [WOH14].

A data repository built by [MAO15b]²¹, was adopted to identify relevant papers/publications in SW CS collaboration barriers. The database includes papers in English from diverse categories (e.g., peer-reviewed conference papers, journal articles, technical reports, and master and PhD theses) published between January 2006 and May 2015. Online library search using seven major search engines: *ACM Digital Library*, *IEEE Xplore Digital Library*, *Springer Link Online Library*, *Wiley Online Library*, *Elsevier ScienceDirect*, *ProQuest Research Library* and *Google Scholar*.

Based on results from the literature review, we performed a backward and forward snowballing search approach as the second step. The goals of this literature review (study 2) were to identify as many SW CS papers as possible extending the publication of Mao et al. [MAO15b] and provide an in-depth understanding of the collaboration barriers, their relations, and their relevance in SW CS context. We conducted this search between April to May 2017.

According to [WOH14], the key actions of the snowballing search strategy are: 1) identifying a starting set of primary papers; 2) identifying further primary papers using the reference lists of each primary paper (backward snowballing); 3) identifying further primary papers that cite the primary papers (forward snowballing); 4) repeating steps 2 and 3 until no new primary papers are found.

We used a set of 18 primary SW CS papers (seeds) provided by automatic and manual searches as input for our snowballing strategy.

The exclusion criteria consisted of removing duplicate results, full papers not available, papers that were not within the scope of this research, papers not written in English, and thesis and dissertation publications.

When the title and abstract of one publication did not give enough information for making a decision, we further reviewed full-text of the paper. This step excluded 163 publications from the survey in Mao et al. [MAO15b] in which refereed papers were not meeting the inclusion criteria as shown in Table 6.

In total, 210 publications were reviewed by examining their titles and abstracts from this repository that fit our research questions. Regardless of collaboration definition the publications were screened according challenges, difficulty, barriers and problems related to collaboration in SW CS scenarios to guide our inclusion and exclusion decisions for both approach (literature review and snowballing) used in the search process. For a study to be included it must report barriers faced by involved parties (platform, crowd, and requester) in SW CS contexts.

In the first step - *qualitative mapping analysis* based on systematic literature review report in [MAO15b], we excluded 171 publications from, and all other candidate publications

²¹ <https://github.com/Rhapsod/software-crowdsourcing-papers>

were analyzed by at least two reviewers. Finally, we retrieve 39 unique publications remained for further analysis in this publication (Table 6).

Table 6 - Review Literature in SW CS publications

Exclusion criteria	Excluded	Total of remaining publication
Initial set of publications		210
Duplicates	2	208
Full publications not available	1	207
No related to collaboration in SW CS	163	44
Thesis/Dissertations publications	5	39
Final set of publications	171	39

In the second step – snowballing search, the exclusion criteria discarded 969 publications from the backward and forward snowballing search. We found out that some publications were much less cited than others, or even having no citation at all. We argue that publications without a minimum number of citations after getting published for a specific period could be considered as not significant in terms of research impact and continuation. It is also important to note that every candidate publication is cross-checked by two reviewers before any inclusion or exclusion decision. After all, we have ended up with five publications (Table 7).

Some of the publications selected in our snowballing process were also cited by [MAO15b] publication. Since these 13 publications [STO14a], [TAJ13], [WU13], [Nag12], [NAG13], [PEN14], [BOU14a], [LAT15b], [LAK10], [LAT13], [KAZ09], [BEG13], [LAT14] were already included as candidate publications in the selection process in the literature review step, we have decided to exclude them from our snowballing final set of publications.

Table 7 - Snowballing search

Exclusion criteria	Excluded	Total of remaining publication
Initial set (Backward: 686, Forward: 282)		968
Not written in English (Backward: 0, Forward: 16)	16	952
Title and abstract (Backward: 539, Forward: 164)	703	249
Duplicate (Backward: 12, Forward: 36)	48	201
Thesis/Dissertations publications (Backward: 1, Forward: 9)	10	191
Full publications not available (Backward: 1, Forward: 1)	2	189
No related to collaboration in SW CS	171	18
Duplicates in the LR final set	13	5
Final set of publications	963	5

The final set of SW CS publications related to collaboration were 39 from qualitative mapping, and five (5) publications from snowballing, accounting for a total of 44 publications as show Table 8.

Table 8 - Final set of publications

Exclusion criteria	Excluded	Total of remaining paper
Total set of papers (LR and snowballing)		1178
Duplicates	1128	50
Full papers not available	1125	3
No related to collaboration in SW CS	791	334
Thesis/Dissertations publications	776	15
No written in English	760	16
Title and abstract (snowballing)	57	703
Duplicates in the LR final set (snowballing)	44	13
Final set of papers		44

Given the set of studies, we created a list of barriers that each publication evidenced. After this, each barrier was linked to supporting text segments from the publications in which they were identified. Using the text segments, we classified the barriers applying coding procedures from Grounded Theory (GT) [STR07] grouped according to their properties to represent categories.

A summary of this literature review is presented in Appendix B and the related results are detailed below.

3.3.3 Results

This study aimed to understand that collaboration assumes specific characteristics in SW CS projects, and these characteristics should impact the involved actors (requester, platform, and the crowd) positive and negatively. In SW CS projects, there is a main aspect that makes these software projects peculiar - the competitive nature to design of work, where this characteristic aggravates the problems of the temporal and physical distance of software development. While SW CS projects enable an increase of alternatives and non-conventional solutions for the same problems due to the crowd diversity, it can, on the other hand, lead to more complicated coordination and communication aspects among competitors in SW CS project.

Our LR revealed preliminary 36 barriers to collaboration in SW CS. For each selected publication the author of this thesis and two more researchers were involved into three steps as follow: each researcher independently read the entire publication, quoted and coded any collaboration barrier cited in the respective publication (using the text segments) [STR07]. We used a spreadsheet to catalogue the extracted data (Figure 10) in publication (author, year, title and venue), collaboration and description of the barriers.

	A	B	C	D	E	F
	Author	Year	Title	Venue	Collaboration barriers	Description
3	Dwarakanath, A. ; Accenture Technol. Labs., Bangalore, India ; Chintala, U. ; Shrikanth, N.C. ; Viridi, G. / Fitzgerald, Brian; Stol, Klaas-Jan; / Peng, Xin; Ali Babar, Muhammad; Ebert, Christof; - / Razieh Lotfalian Saremi, Ye Yang / Walter S. Lasecki et al	2015	Crowd Build: A Methodology for Enterprise Software Development Using Crowdsourcing	(CSI-SE), 2015 IEEE/ACM 2nd International Workshop	high effort of the requester to answer crowd	answering a large number of questions
4					scarce motivation's crowd	crowd participations' motivation
5					competition model	in competition - different crowd providers of the same task
6					difficult to orchestrating virtual teams	lack comprehensive support for building virtual teams
7					few support for collaboration among Crowd members	few support for collaboration among Crowd members
8					few transparency in CVS	no support share version control and integrating issue trackers
9	Peng, Xin; Ali Babar, Muhammad; Ebert, Christo	2014	Collaborative Software Development Platforms for Crowdsourcing	IEEE Software	large scale collaboration of distributed members	large scale collaboration of distributed members
10					linear microtask decomposition	more elaborate and collaborative workflows need to be designed to decomposed micro tasks (tasks design).
11	Zhao, Mengyao; Hoek, Andre van der; / LaToza,	2015	A Brief Perspective on Microtask Crowdsourcing Workflows for Interface Design	Proceedings of the 2nd International Workshop on Crowdsourcing in Software Engineering	few iteration (task design)	Dwarakanath, A. ; Accenture Technol. Labs., Bangalore, India ; Chintala, U. ; Shrikanth, N.C. ; Viridi, G. / Fitzgerald, Brian; Stol, Klaas-Jan, / Peng, Xin; Ali Babar, Muhammad; Ebert, Christof; - / Zhao, Mengyao; Hoek, Andre van der; / LaToza, Thomas D.; Ben Towne, W.; van der Hoek, Andre; Herbslieb, James D.;

Figure 10 - Extracted data from collaboration barriers

Once we finished data extraction that each publication evidenced and, coding each collaboration barriers, we peer reviewed the results. Discrepancies around barriers and coding were discussed between the 2 researchers and resolved with the help of a third one. Next, the author of this thesis reanalyzed the barriers obtained and proposed categories to aggregate the collaboration barriers. The goal of this reanalysis was to combine the findings to accommodating all the evidenced barriers in a more unified mapping. After all, we discussed among 3 of the researchers (including the author of this thesis) and critically reviewed in several review sessions until it was considered stable the categories and collaboration barriers. This was done to mitigate the bias caused by a single researcher and to reach a common understanding about the code nomenclature and categories.

The preliminary results of this mapping are part of the results published in Machado et al. [MAC16c] in ICGSE.

The SW CS collaboration barriers' resulting mapping of the analysis was the emergence of 23 barriers grouped into five categories: social interaction, process management, cultural diversity, competition models, and task design. We also identified with frequency each barrier is mentioned in studies (see column 3 of Table 9). An overview of the results can

be seen in **Table 9**. We describe each category and its associated barriers in the next section.

Table 9 - Map of Collaboration barriers from LR

Category	Barriers (B)	Freq.	Reference
Social Interaction	(BA1) Lack of Informal communication	7	[STO14a], [TAJ14], [GUA15], [TAJ13], [NAG12], [MAC17], [GRA16]
	(BA2) Lack of interaction among crowd members	6	[PEN14], [BOU14a], [GRA16] [MAC17], [LAT14], [RIE17], [LAT15a]
	(BA5) Lack of real-time collaboration	5	[LAS15], [MEH14], [GOL11a], [GOL11b], [BOU14a]
	(BA7) Scarce social media support	4	[BEG13], [BAR13], [STO14b], [TSA14]
	(BA8) Few interaction among involved participants inside the platform	4	[TAJ13], [FIT15], [NAY14], [BOU14a],
	(BA15) Asynchronous communication	3	[BOU14a], [PEN14], [MAC17]
	(BA19) Weak internal collaboration between the platform and the requester	2	[STO14a], [FIT15]
	(BA23) Lack of feedback among participants in the platform	1	[Hoß14]
Process Management	(BA3) Information management complexity	6	[STO14a], [BOU14a], [FIT15], [GUA15], [MIN12], [DWA15]
	(BA6) Technical and infrastructure setting issues	5	[PEN14], [GOL11], [TSA14], [LAT14], [HOS14c]
	(BA11) Difficulty in team management in large scale distributed settings	4	[PEN14], [HOS14b] [KAZ09], [NAY14]
	(BA16) Hurdle single point of contact	4	[STO14a], [FIT15], [MEH14]
	(BA13) No unified software process methodologies	3	[STO14a], [FIT15], [KAZ10]
Task Design	(BA9) Low global project view	4	[STO14a], [TAJ13], [LAT13], [LAT15c]
	(BA10) Micro-task decomposition issues	4	[STO14a], [ZHA15], [LAT14], [SCH11]
	(BA17) Lack or incomplete documentation	3	[STO14a], [FIT15], [LAK10].
	(BA21) Security and privacy issues	2	[STO14], [Hoß14]
	(BA22) Difficulty to find fit task allocation	2	[STO14a], [LAT15b]

Category	Barriers (B)	Freq.	Reference
Competition Model	(BA4) Difficulty for dealing with competitors	6	[WU13], [NAG13] [HOS14b], [WU13], [NAG12], [BOUG14b]
	(BA12) Weak commitment between involved actors	3	[TAJ13], [NAP13], [HOS15] [HOS15]
	(BA14) Difficulty to increase and keep participants' motivation	3	[TAJ13], [BOU14a], [HOS15]
Cultural Diversity	(BA18) Teams heterogeneity	3	[TAJ13], [BOU14a], [HOS15]
	(BA20) Diversity of languages	2	[PRI14], [MAC16a]

We ranked these categories according to the number of barriers that evidence them (show **Figure 11**). In each category, some studies support more than one barrier. Some barriers also are cause and effect of other barrier. For example, "*information management*" is a barrier that can be posed to "*low global project view*" and "*weak interaction among involved actors*" barriers, because of reduced opportunities to communicate in SW CS platforms.

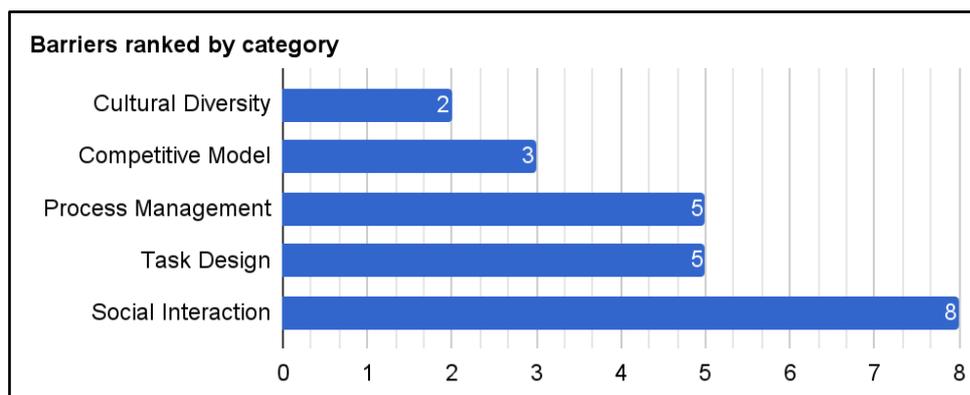


Figure 11 - Statistics of barriers to collaboration in SW CS

The category with the greatest number of barriers is related to social interaction among actors involved in SW CS environment, accounting for 30 studies distributed in eight barriers.

Secondly, process management and task design categories grouped five barriers each one. The other two categories range from Competitive model and Cultural diversity each one distributed in three and two barriers, respectively.

a) Social Interaction

It represents the collaboration barriers related to the possibility to exchange messages among involved parties and, the way they interact with each other during the task performance in the SW CS platform, including issues related to how the three involved parties communicate, which mechanisms are available for collaboration, and what social

structure are present. This category is one of the most evidenced among the selected studies, appearing in 33 studies (70%). Within this category, we identified evidence of eight different barriers that can influence collaboration in SW CS: BA1 (Lacking of informal communication), BA2 (Lacking of interaction among crowd members,) BA5 (Lacking of real time collaboration), BA7 (Scarce social media support), BA8 (Few interaction among involved parties inside the platform), BA15 (Asynchronous communication), BA19 (Weak internal collaboration between requester and platform) and, BA23 (Lack of feedback among participants in the platform).

Lacking of informal communication (BA1) is evidenced in SW CS studies by misunderstandings, no direct communication, and fleeting relationship. We found evidence of this barrier in seven of the primary studies [STO14a], [TAJ14], [GUA15]; [NAG12], [MAC17], [GRA16], [TAJ13]. This barrier can have big impact on the quality of the final solution, and thus may have an impact on the contributors end up doing something that is slightly different from employers' wishes. To [TAJ14] lacking of informal and direct communication affects the satisfactory understanding of task's goals. So, because of that, communication between the crowd and the clients is vital for the success of a CS project, mainly to answer crowd's questions. [GUA15] have also appointed the necessity of more informative and communication-oriented crowd because of the number of crowd questions about task details on testing platforms, specifically. On the other hand, [STO14a] emphasize that this fleeting relationship due to the indirect (asynchronous) nature of the communication with the crowd filter through of the co-pilot (platform mediator) provided by TopCoder platform does not promote a real opportunity to build up a more active collaboration with any crowd developers: *"However, there is no informal communication mechanism. You cannot yell at the person in the next cubicle and get the answer very quickly. In contrast to distributed development which typically involves other developers from the same organization, the only relationship which tended to build over time was that with the TC co-pilot. There was no real opportunity to build up a relationship with any of the TC developers, as interaction was filtered through a number of layers"* [STO14a], [NAG12] attribute that few communication channels reduce the level of interactions among involved parties (crowd, platform and requester) about topics of interest to this community assigned to the same task or without any associated task.

Lacking of interaction among crowd members (BA2) occurs on disperse and competitive SW CS settings. This barrier is related to the perceived lack of communication among crowd workers. In this way, SW CS platforms also do not provide alternatives communication channel. According six studies in the literature there is insufficient support for collaboration among crowd members specifically during the execution of the same tasks or in different ones. This barrier was evidenced in six studies by [PEN14], [BOU14a], [GRA16], [MAC17], [LAT14], [RIE16].

The incentive to crowd's interact by side of the SW CS platforms is restricted as mentioned by [PEN14] *"Crowdsourcing platforms provide little support for collaboration among crowd members"*.

Boudreau et al. [BOU14A] add: *"Regarding socially emergent processes, we find evidence of a virtuous circle of effort and collaborative interactions taking place. The likelihood an individual chooses to participate relates to levels of teammates' communications and communications among teammates begot more communications and more effort. These patterns of reciprocating complementarities are consistent with the importance of setting productive social interactions and dynamics into motion in order to catalyze collaboration even in this rather weak social context."*

Riedl and Woolley [RIE17], investigate how to effectively incorporate team-based collaboration in a setting that has been individual-based. In a field experiment on online platforms they evaluate the influence of members skills, incentives and emergent collaboration process on performance of crowd-based teams. The authors evidenced this barrier through of coordination, a collaboration characteristic in SW CS: *"temporal patterns of coordinating are a particular challenge in crowd-based teams, as, in addition to being globally distributed participants also tend to touch their contributions in around the edges of their "regular" activities, leading to an even less regular schedule than we would observe in global teams in work organizations"*.

The study conducted by [GRA16] shows *when platforms do not natively support collaboration among the crowd, workers create widespread yet invisible forms of collaboration that take place off-platform*. They focus on understanding how workers collaborate organically with other crowd worker assigned to the same tasks. Their findings have already associated with BA1. [MAC17] presents another case of a barrier supported by evidence from a case study where their results show that collaboration in SW CS competitive model was perceived as explicit and direct between the platform and crowd via forum and, in a more indirect way between crowd and crowd, according participants who work in a task.

LaToza et al. [LAT14] in the opposite way, offers some strategies to overcome restrict visibility among crowd workers. They implemented a *CrowdCode* tool that provides a number of features to help workers maintain awareness of the current state of the project, it added to a personal activity feed letting workers track their work. Other example is the "Ask the Crowd feature" to enable the crowd to still make progress in the face of unexpected information needs.: *"It also enables workers to go off topic and forge closer relationships with other workers... A majority of participants who worked on a small programming task agreed that the opportunity for communication beyond what was provided would help them to work more effectively. Participants cited a desire to share technical experience, clarify tasks, ask questions about material that others had written."*

Lacking of real time collaboration (BA5) occurs when technical decisions limit communication between parties, e.g. the CS platform needs to be able to support the exchange of messages among requesters and crowd in order to reduce gaps and ambiguity in timely manner. We found evidence of this barrier in five of the primary studies [LAS15], [MEH14], [GOL11a], [GOL11b] and, [BOU14a].

In [GOL11a] and [GOL11b] the authors discuss *"a scenario where the lacking of real time communication introduces problems in a collaborative coding where program*

compilation errors introduced by other users". Results from these studies revealed that synchronous collaboration and transparency between programmers allows team members to better understand and identify changes made in the code since the last handoff session, thus it avoid rework. To [LAS15], platforms should provide real time CS infrastructure *"to facilitate self-managing and real time crowd coordination"*. Furthermore, *"timely completion of the subtasks and their collaboration can only be achieved through real-time collaboration and synchronization between the crowd-workers"*, [MEH14].

Scarce social media support (BA7) describes scarce integration of the social media technologies supported by SW CS platforms that crowd workers face when trying to communicate through other channels. We found evidence of this barrier in four of the primary studies [BAR13], [STO14b], [BEG13], [TSA14]. In three studies using Q&A websites (Stack Overflow), social coding (GitHub)²² [BAR13], [STO14b], [BEG13], it is approached social media for collaborative development and crowdsourcing content in software engineering. Even solitary developers need to interact directly or indirectly with others to learn, to understand requirements and to seek feedback on their creations. Low integration with social media systems in CS platforms was mentioned as a constant threat to smooth interaction and it also reduces the collaboration to share knowledge among online communities.

To Storey et al. [STO14b], developers should use social media to understand how they communicate and collaborate, and to gain insights into the challenges that they face. In preliminary results from a large survey with developers who work on projects mediated through GitHub, several respondents mentioned *"issues when the tools they use lack integration with programming artifacts. This lack of support caused friction when switching between tools and led to poor traceability between discussions and software artifacts."* Besides, Begel et al. [BEG13], reports the role social networking play in today's software development world among different tools and systems.

Few interactions among involved parties inside the platform (BA8) refers to the friction that happens when specific business rules of some tasks shall require contact with the client/request for further clarification and decision-making and it is important to answer the questions by the crowd in time during task execution specially, for competitive SW CS platforms because the deadline of the contest. This barrier is evidenced in four of the primary studies [TAJ13], [FIT15], [NAY14] and, [BOU14A].

[FIT15], describes collaboration and interaction between client and requester as *"a required discipline to ensure that submissions of the crowd can be internally reviewed by client in a short period to report possible problems"*. Topcoder platform, for example, has a process in which the solution review phase can have several interactions between platform and requester, who needs to accept or reject the submissions within a specific timeframe. Considering the task estimated deadline, interactions happen in a short term and the client needs to be available for feedback so that the process is not delayed or does not take too long. Thus, in case interaction through internal communication among the parties does not

²² <https://github.com/>

happen within the given forecast time, it poses a collaboration barrier for the crowd. In this way [NAY14] report *"that misalignment among involved parties (crowd, requester and platform) as a major risk for successful completion of software projects mainly due to no software physical presence and restricted and asynchronous communication."*

Other studies provide more evidence towards weak interaction among involved parties and its importance for promote collaboration in SW CS environment. For instance, [BOU14b] claims that *"the crowdsourcing projects are also often transitory or short-lived. Thus, the baseline conditions for any sort of collaboration and team effort towards a collective joint goal seem sparse at best and potentially fraught with failure."* [TAJ13] focus on three determinants to achieve success in CS software development: the characteristics of the project, the composition of the crowd and the relationship among key players. They highlight the influence of maintaining a communication quality to the success of CS software development: *"Relations among developers and between crowdsourcer and community are of utter importance. (...). On the other hand, trust formation, communication quality, and identification with the project team are factors that have been associated with success of OSS and due to similarity of the development methods we can expect the same to hold true for crowdsourcing."*

Asynchronous communication (BA15) occurs when SW CS platform just provide asynchronous interaction channels among involved parties while conducting work in the task's period. In this way, the asynchronous interactions no providing rapid feedback among members it becomes one of the communication issue on SW CS collaboration. This barrier was mentioned by three studies [BOU14b], [PEN14], [MAC17] and, it is also associate with other barriers emerged by our analysis such as - *lack of informal communication (BA1), lacking of real time collaboration (BA5), weak interaction among involved actors inside the platform (BA8) and, lacking of interaction among crowd members (BA2).*

Boudreau et al. [BOU14], mention several problems relate to communication via forums such as: communication feedback' is delayed, and interruptions or long pauses in communication often occur not facilitating the direct and online communication. At the same time that asynchronous communication mechanisms are useful to support and stored the exchange information about task's details in different timezones, and it pose as thread off within SW CS communication's projects. The task update's that happen during the forum board can't be updating of the spec documentations of the task. The authors appointed: *"In an asynchronous discussion, typically many topics are active at the same time, with team members making contributions at different times, possibly on different topics. This pattern can increase information overload and may reduce the synergy of team members if there are no links among the responses."*

In addition, [PEN14] says: *"Crowdsourcing platforms support communication by providing task-specific forums for crowd members to ask questions and communicate with each other... However, crowdsourcing platforms provide little support for collaboration among crowd members."*

In a more recent study Machado et al. [MAC17] presented a case study using TopCoder platform try to understand how crowd workers perceived collaboration and which

challenges they faced to perform a task. They found that consider the competitive environment, the platform intentionally supports communication among members of the crowd in a restricted way via forum *"tasks can lead to misunderstanding and thus may have a big impact on the quality of the final solution and in the influx of newcomers"*. Thus, the asynchronous and unstructured communication on online forums barrier suggests that a significant challenge facing online teams' coordination and collaboration of the temporal patterns of group behavior, which has been shown to be critical to performance.

Weak internal collaboration between requester and platform (BA19) is evidenced in two of the primary studies [STO14], [FIT15]. It occurs when at least two parties request inside collaboration in a particular subject, e.g. internal collaboration between requester and platform where one of them is required to elaborate adequate documentation and artifacts for the requirement task specification to than share it with crowd participants. The second point is to ensure internally reviewed of the submitted solutions by crowd and communicate the platform about reviewer's decisions [FIT15]. The period is too short to create documents, coding stands and templates, evaluation criterias, and rewarding crowdsourced tasks provides between requester and platform. Besides that, report problems of solutions submitted by crowd emerge also how a barrier in SW CS to internal coordination and collaboration [STO14a].

Lack of feedback among actors in the platform (BA23). Be available during the CS task through feedback channels is an important issue to promote collaboration and engage crowd workers while they performing task. The feedback channel should be accessible throughout the whole task, not only at the end of it.

We found evidence of this barrier in Hoßfeld et al. [Hoß14]. The authors showed that: *"Even with simplified language, proper instructions, and training sessions test participants might face issues while participating in tests. These issues might either occur due to misunderstandings or other unclear points, but also due to hard- and software issues. Therefore, it is important to provide a feedback channel to the users to contact the experimenter."*

Some providers do not allow contact between crowd workers and employers outside the platforms. External, interactive feedback channels should be provided by platform include e.g., live chat or email support for small tests or forum threads for larger test groups.

In general, all questions from the users should be answered. An intensive discussion with the workers and a reasonable support helps to improve the task design and respectively the task results. Moreover, it helps to increase the employer's reputation, as workers tend to gather in virtual communities and share their experiences with certain employers and tasks according [Hoß14].

b) Process Management

This category describes project and process management problems which have been evidenced as barriers to SW CS participants. Such problems are related to outdated information between documentation and decision making happening in the communication

channels, coordination among several participants and so on. We have identified five barriers in this category, namely BA3 (Information management complexity), BA6 (Technical and infrastructure setting issues), BA11 (Difficulty in team management in large scale distributed settings), BA13 (No unified software process methodologies), and BA16 (Hurdle single point of contact). Evidence gathered from primary studies representing the difficulty to mediate SW CS virtual teams in terms of extra overhead under the management perspective was highlighted by [GUA15], [STO14a], [BOU14a], [MIN12], [DWA15], [FIT15], [KAZ10], [TAJ14], [NAY14], [PEN14].

Information management complexity (BA3) is a barrier due to lack or overhead of information. It may also influence on the solution quality, as previously mentioned in the asynchronous communication barrier, because contributors (crowd) end up doing something that is slightly different from requesters' expectations. The need of a more communicative and informative platform to serve the crowd and to disseminate practices (testing, expected results, objectives) of the tasks were emphasized according to [GUA15], specifically for the crowdtesting environment. We found evidence of this barrier in six of the primary studies [STO14], [BOU14a], [FIT15], [GUA15], [MIN12], [DWA15].

The level of temporal and asynchronous coordination across widely distributed crowd contributors remains as important challenges as mentioned in [BOU14a] principally pela poorly designed CS tasks and insufficient interaction among the crowd and requesters. Thus, the authors in revealed the importance to provide as much as possible interaction about task's instructions or share information with each other.

During task execution period (often short-lived), the crowd participants can need information on time provided by platforms in terms of requirements details, technical clarifications, artifacts, setup environment to start the tasks and so on. Some of this information are gathered though requester that demands the task and it can imposed overhead communication based on the effort necessary of the requesters to answer the SW CS community inquires during the realization of the tasks according to [STO14a]. In this way, information via clearly requirements documentation plays an important role, once this is the key channel through which crowd developers will know what to develop: "*Finding the right balance is important; giving either too little or too much information will result in a deliverable that is likely to be unacceptable*" [FIT15].

The exchange information between requester, platform and crowd need to be coordinated during the task's period. How each participant is an individual member who develops and executes your task solutions's in isolated actions, SW CS platforms should be guarantee visibility and low latency of information via online communication to reduce gaps of requirement and tacit assumptions of the tasks as mentioned in [DWA15]. As also indicated by [MIN12], "*the time waiting for human response while conducting work during the task's period can be critical for the crowd participants*", specially in SW CS competition approach's. A significant amount of time in provide these answers by platform affect the collaboration during the activity due to manage information complexity in real time as mentioned in *lack of real-time collaboration* barrier (BA5).

No unified software process methodologies (BA13). This barrier was evidenced by [STO14], [FIT15], and [KAZ10] once in SW CS environment, a requester has no knowledge of the developers in the crowd that deliver the software, nor of the process who they might follow and, therefore has no control over these aspects. Aligning different software development process cultures with SW CS projects can be problematic and difficult to deal with in terms of collaboration and coordination among requesters and crowd participants.

In [KAZ10] the authors highlight the new norm to conduct software projects by open teams: *"In the past we managed development as traditional projects, employing closed teams of developers who work from a consistent set of requirements. Now, volunteer projects and decentralized production processes with no managers and no centralized decision-making processes are becoming the norm."*

In the case study conducted by [STO14], they pointed a set of concerns in crowdsourcing software development: *"Of particular concern in crowdsourcing is that a customer has no knowledge of the developers that deliver the software, nor of the process that they might follow, and therefore has no control over these aspects."* A complementary study in [FIT15] reported: *"It is important to become familiar with the crowdsourcing process at the outset, so that architects, developers and project managers can prepare and discuss internally what needs to be done for a smooth interaction with the crowd"*.

Technical and infrastructure issues settings (BA6). We found evidences of the technical and infrastructure issues that influence on collaboration level by crowd in terms of effort to each setting up environment task's, effort to understanding unclear tasks instructions and not providing share code repositories. Besides that, in a collaborative software development all community members should be able to access the same artifacts and environment to write and test their codes. Out of five studies that mention this barrier, three of them are results reported from tools which were developed by the own authors in order to alleviate technical and infrastructure problems. Such tools comprise distributed code edition in real time, system for tracking work, and Q&A system to support crowd work during work in progress.

Few transparency during software development can cause a problem introducing compilation coding errors by other users. In [GOL11b], the author proposed a *Collabode tool* to support real time code edition and transparency between programmers as mentioned in *"Lack of real-time collaboration"* (BA5).

In [LAT14] and [LAT15], the authors developed *CrowdCode*, a cloud IDE for crowd development to track changes, enable workers to write code, reuse functions, test, and debug within self-contained microtasks. As already mentioned in the *"Lack of interaction among crowd members"* barrier (BA2), this tool also provides social features to help workers maintain awareness of the current state of the project. Besides that, the authors in Besides that, the authors in [LAT15] investigated if the Q&A would help crowd workers share knowledge about technical decisions and artifact conflicts. The result obtained by the study that used a Q&A system in a controlled experiment was: *"Workers also used the Q&A system to ask questions about how CrowdCode itself worked. Finally, workers sometimes used Q&A for explicit coordination, requesting others who might be working on an artifact"*

to take an action with it or to gain an understanding of the current overall status of the project."

High heterogeneous infrastructure needs adequate mechanisms to catalyze collaboration specially in different SW CS tasks demand. Hence, the absence of a centralized management and environment to control code commit, issue trackers, and share version control is particularly sensitive to the external collaboration success, as cited by two studies. In Tsai et al. [TSA14], the authors mentioned that crowdsourcing platforms should support setting environment during task broadcast by the crowd: *"Given software crowdsourcing's distributed nature, it needs a powerful development environment to facilitate software design, coding, test, and deployment across distributed and heterogeneous infrastructures"*.

In the same way, [PEN14] mentioned: *"It's quite easy for a group to conduct its development tasks if the platform can automatically allocate the required resources, such as virtual machines, tools, libraries, and testing environments"*.

Difficulty in team management in large scale distributed settings (BA11). Large scale collaboration with distributed and heterogeneous members is mentioned by literature how a barrier in terms of coordination, quality control and task decomposition. This barrier was highlighted by literature in four studies as follow.

Nayebi et al. [NAY14] reported some issues that this barrier can cause: *"In the presence of large-scale and distributed development, decision processes include various stakeholders across different locations. Group decision support systems are intended to accommodate this challenge."*

According [PEN14], *"developers collaborate at different levels and some of them might work on the same piece of the project (source code, UML diagram or interfaces design) or collaborate on a set of shared artifacts with the support of version control systems."*

In [KAZ09], the authors cited important aspects related team management in distributed setting: *"However, future crowdsourced systems will be community-driven and decentralized, with little overall control. (...) opening a project to the crowds, management accepts that they consist of unknown people at disparate locations anywhere on the Internet and in time zones, countries, and cultures. This is certainly the case for nontrivial OSS projects. Managing them means the periphery shares in their success and, to a large extent, is self-governing and self-adaptive."*

In terms of coordination, the difficult to mediate and orchestrate virtual teams is potentialize in SW CS by dynamic and transitory nature where the crowd members can leave or entry arbitrariamente during task's performance. Besides that, more overhead in management (feedback, screening and select task's solution) process by solution submitted among different crowd providers of the same task as mentioned by [HOS14c].

There is not a lot of group cohesion because of the mish-mash of background, perspectives, artifacts and tools when trying to development a solution. In contrasting of

small and homogenous groups, SW CS context have a "discontinuities" of team distribution due your unstable, large, open and diverse participant's nature.

This barrie is also associated with the lack of real time collaboration (BA5), internal collaboration between platform and requester and, (B3) weak interaction among involved parties.

Hurdle single point of contact (BA16). Interacting flow with the crowd during the task's period can be very time-consuming activity due the nature of interaction in SW CS environments. In [FIT15], the authors evidenced the import issue in terms of "*assign a special person to answer questions from the crowd*" to alleviate the communication and collaboration difficulties among requester and crowd. The election of one focal point from the customer side to manage communication (technical or operational answers), review and feedback of the crowd on the Q&A forum is crucial during work in progress to provide more guidance for the crowd workers. However, coordinating different participants with different capabilities can overload the requester side and the eligible mediator from platform. For example, at TopCoder, the interaction with crowd contestants is mediated by co-pilots, in the uTest, the platform elected a Testing Technical Leader (TTL) how point of contact among crowd tester and requesters. This mediator, often can not respond in time all questions demands of the crowd, especially for issues that depend on a decision of the requester. Such a situation, effort and energy to respond each participant and delayed client responses influences the collaboration among SW CS actors according to [STO14a].

There are new several roles which emerge in crowdsourcing software development. The identification of roles and responsibilities of the involved actors is an important issue to carry out collaboration between each role as evidenced by [MEH14].

Machado et al. [MAC17], pointed in their case study, that "*communication and coordination via forum can impose an overload and time latency during SW CS projects in terms of the effort to prepare task specification, documentation, significant amount of answers crowd questions, evaluating and feedback to submitted solutions by crowd participants*".

c) Competition Model

This category represents the problem related to competitive behavior that crowd workers face on software development contests in the platforms that follow a competitive engagement model, where tasks are posted as a challenge. Workers, based on their interests and skillset, register for the challenge and submit their solutions for a monetary reward. Paid, online and competitive CS platforms include general-purpose marketplaces (e.g. Innocentive, Upwork) as well as markets for specific expertise (uTest, 99Designs, TopCoder).

Difficulty for dealing with competitors (BA4). According many authors, competition can be an important issue that affects collaboration in SW CS environment among crowd members because of the monetary incentives. Besides that, platforms use ranking and

reputation level of more experienced competitor (with a high rating) to classify the crowd participants. These issues may help explain the crowds' low levels of collaboration, and the barrier was evidenced in six studies. Different crowd providers' solutions for the same tasks by a monetary reward.

Hosseini et al. [HOS15] discuss the effect of the ranking and reputation in the CS competitive model to establish collaborative relationships with other participants, especially among seasoned and newcomer participants: *"The competent crowd might also include participants' inflated egos which would then reduce the level of collaboration and lead to conflicts and inconsistency."*

The competition model can inhibit participation from crowd developers' newcomers as reported Wu et al. [WU13]: *"While competitions promote creativity and support quality software development, but stiff competitions may also restrict massive participation. (...) Thus, stiff software competitions may restrict the activities of the crowd."* Related this problem, the authors proposed in a continuous study, an evaluation framework to analyze collaborative and competitive nature of software crowdsourcing processes toward different goals such as broadening participation [WU13].

[NAG13] and [NAG14], discuss the applicability of crowdsourcing and competition for problems that require a collaborative or cooperative effort to be successful. The author showed the results of a development spaceflight software tournament in TopCoder platform. *"Development through competitions requires a careful balance of competition and collaboration to achieve its goals." Much of the collaboration in TopCoder is structured collaboration, i.e. the TopCoder process dictates how that collaboration takes place"*.

Regarding the SW CS competition model, the authors in [BOU14a], found that levels of effort are driven by cash incentives and the presence of other interacting teammates. In contrast, the level of collaboration was not sensitive to cash incentives but for actively interaction among individual crowd: *"Instead, individuals increased their communication if teammates were also actively participating."*

Weak commitment among involved actors (BA12). Collaboration in paid, online, competitive, and often unknown participants (know only by aliases), as the SW CS context, influence in reduced social interaction among involved actors. Competitive SW CS platforms, introduces new challenges such as ensuring reliability participants and cheating prevention. This barrier is mentioned by [TAJ13], [NAP13] and [HOS15].

Low relationship in which the firm and community act as "no good member" damaging collaborative interactions in various aspects - *"failure to deliver promised code, code misappropriation, free riding, law firm involvement, or attempts by firms to influence the community in their own interest"*, according observed in [NAP13]

SW CS Competitions models can influence participants in establish trust among each one. Besides that, it is difficult to implement collaboration with anonymity and cash incentives. The anonymity also encourages some participants works sloppy or to cheat in order to increase their income, by maximizing the number of completed tasks per time [HOS15]. *"Furthermore, anonymity is one way of assuring users' privacy and security."*

However, it can also be risky as it would allow malicious users to join in. Finding the right incentives and how it is linked to competence, intrinsic motivation and anonymity are other research challenges to investigate."

Tajedin and Nevo, [TAJ13], argue that "*cohesion among project team members would lead to more effective communication and learning*". For the cohesion make possible in SW CS context is needs to provide trust information, and communication quality. These attributes have been associated with collaborative projects as presented by the authors.

Difficulty to increase and keep participants' motivation (BA14): Crowd motivation in collaborating can be decreased due competition aspects, onboarding process, task documentation and so on. According to [TAJ13], "*the motivation could have different levels and can be related to intrinsic (reputation in the community) and extrinsic motivation factors (explicit rewards)*". Other factor that can be influence on keeping crowd participants motivated to persevere and collaborate with platform (e.g. to accept more tasks) is related to feedback process during task completion: "*Providing feedback to the crowd is often seen in a positive way. (...), which can improve the performance of participants and also motivate them to persevere and accept more tasks.*" [HOS15]

A contradictory result was present in a study by [BOU14a] from a field experiment on TopCoder platform where they discuss the role of incentives versus social processes in catalyzing motivation to collaborate. The authors found that there is "*a growing body of work on participation and motivations of workers in collaborative online contexts that points to the importance of non-cash-based incentives and motivations*".

d) Task Design

This category represents the issue related to difficulties that the three actors involved face when planning, decomposing, broadcasting and assigning a SW CS task. That happens because work is split into a set of smaller tasks instructions, which are often incomplete or ambiguous. Besides that, the participants who submit solutions are not part of the organization (requester) which demands the task, reducing the context of work from crowd participants. This category has five barriers: BA9 (Low global project view), BA10 (Micro-task decomposition issues,) BA17 (Lack or incomplete documentation), BA21 (Security and privacy issues) and BA22 (Difficulty to find fit task allocation).

Low global project view (BA9): Once the crowd does not take part in the organization that demanded the task, the concerns regarding intellectual property and, business rules are addressed as few contexts that crowd members receive in the tasks of the SW CS initiatives. About this issue, it is crucial to provide information as much as possible to attenuate the communication issues and provide more guidance to the crowd better understand what is expect according to [TAJ13].

In [LAT13], as consequences of all this scarce global view during SW CS tasks', the collaboration barriers are influenced in terms of communication and information fragmentation: "*Crowd development may not be well-suited to all domains, such as those that require large amounts of domain expertise, safety critical systems, or those with*

sensitive business information". In [LAT15c] the authors cast doubt on what aspects of software engineering can be done with only local information, and what aspects require a global view of the project. Once the crowd does not have a vision of the whole project, just a task goals' overview, it is more challenging for workers to understand the impact and meaningful of the task.

In [STO14a] the authors argue the scarce context about tasks' specification: "*There is a fine balance between providing a sufficiently detailed specification for the tasks being crowdsourced on the one hand, and hiring innovation or critical business rules with overly detailed specifications on the other hand*". Some information about software aspects cannot be shared globally with anonymous crowd workers due to privacy or intellectual property concerns, and this issue discouraging collaboration among the crowd workers.

Microtask decomposition issues (BA10) is a barrier in SW CS because there are a lot of dependencies between tasks. [LAT14] argue that "*decomposing smaller pieces of work can increase the amount of overhead leading to corresponding challenges, such as communication since more workers may need to understand some of the same aspects of the current status of the work to move on and contribute with it.*" Similar challenges can be seen in coordinating contributions and managing dependencies among microtasks' software during development work by crowd. [STO14a], describe that decomposing a software project as a key concern in SW CS: "*Given the interdependencies in software, different developers working on a task can't know how their code fits into the resulting software product, in terms of understanding interfaces and assumptions made.*" Thus, to enabling mass contribution by crowd developers who can be worked at the same time, requires an extra work to ensure information about coding standard and templates, and technical specifications. This aspect was pointed in the *information management complexity* barrier (BA3) as a concern to promote symmetric information between crowd and requesters.

To overcome the information loss caused by task decomposition, in [SCH12] the authors, conducted an experiment by recruiting workers from Amazon's Mechanical Turk to perform code verification with VeriWeb. In their results, he: "*characterize the minimal communication overhead incurred when VeriWeb is used collaboratively by observing two pairs of developers each use the tool simultaneously to verify a single program*".

In this way, for interface design phase, the authors in [ZHA15] presented a comparison between a normal decompose workflow with an interactive workflow that helps crowd designers to broadcast and manage their work in different parts of the design that they are working on. The result of this study provided insights into communication, earlier feedback and collaborative view about interface design tasks among the crowd workers who participated of the experiment.

Lack or incomplete documentation (BA17). 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. Regarding task documentation, [STO14a] pointed: "*Organizations may be hesitant to provide too many details on a certain task (i.e., module or component) that is crowdsourced, yet sufficient detail in the specification is necessary for developers in the crowd to understand what the crowdsourcing organization is requesting*". "The same

authors [FIT15], in a complementary study, reported a common problem in CS software development: *"The documentation that specifies the context and the requirements for the software development task at hand must be easy to understand and provide sufficient information for crowd developers to do their task"*. Overwhelming the crowd with unclear and insufficient instructions impact in few or even no participation task's submissions and poorly solutions by crowd.

As Lakhani et al. [LAK10] 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 or vast in scope"*.

The evidence found in the literature suggest that provide proper documentation will likely lead to decrease time spent on responding to crowd's queries about task's documentation as mentioned in barrier about *hurdle single point of contact* (BA16).

Difficulty to find fit task allocation (BA22). In open software development contributions by the unknown crowd, new challenges emerge by side of requester and platforms in coordination at scale, collaboration to work together and, to achieve the highest possible quality of service. To distribute and allocate tasks to appropriate and competent people, SW CS platforms should in promoting the group formation or self-organization of people with either similar or diverse, cross-functional skills or background.

This barrier is evidenced by [LAT15c], *"How can workers be matched to microtasks, most efficiently allocating the knowledge workers bring to bear to the work to be done? Which aspects of software work benefit most from expertise, and how can this expertise best be leveraged?"*

In [STO14a] the authors reports: *"Multiple tasks in parallel have implications for coordination and quality in order to attract sufficient crowd participants with required skill to be allocate or assignment on crowdsourced tasks"*.

Security and Privacy issues (BA21). This barrier refers for collaboration and cooperation among requester (task creator) and crowd in two aspects: (i) Intellectual property (IP) concerns, the solutions that crowd workers submit are not theirs as happen in open source code and (ii) confidentiality of the data to protect interests of requesters.

In [STO14a], the authors comment that *"organizations may be hesitant to provide too many details on a certain task (i.e. module or component) that is crowdsourced."*

Hossfeld et al. [Hoß14] mentioned: *"Along with these new opportunities, however, come a host of technical challenges as well as privacy-related issues."*

Regarding privacy, security, or intellectual property concerns are need to balance in how much information should be shared to motivate and retain workers and how to share context without introducing confidential problems to the core business value.

e) Cultural Diversity

Geographical, organizational, and cultural diversity has been shown to cause of several problems in software development. In SW CS, it is amplified due to large and open

community nature. As it increases in diversity, it also deteriorates the performance of coordination and communication and, it requires proper mechanisms to lead with these wide differences. In this category, we identified two barriers: BA18 (Teams' heterogeneity) and BA20 (Language's diversity).

Teams' heterogeneity complexity (BA18) is related to the huge heterogeneity of language, backgrounds, and expertise of the crowdworks. This aspect can influence in collaboration to keep willing participants to delivery good contributions in order to establish a common grounded on task's specification (technical and operational view), and effective communication among involved parties.

This barrier is reported in three studies [BOU14a], [TAJ13], [HOS15] and [BOU14b] describes how heterogeneity and sparse social context among crowd individuals' results in a miscommunication and different point of views around the tasks specification increase the level of temporal coordination within the team. On the other hand, Tajedin et al. [TAJ13] highlights the strength points of CS is the different viewpoints and methods to solving problems: *"But one can imagine a maximum point for this after which not only with the increase of diversity the performance does not improve but also it deteriorates due to problems that emerge because of coordination and communication."* To Hosseini et al. [HOS15], geographical diversity, inadequate communication, difficulties in knowledge management and, issues related to timezone differences are causes of several problems in the field of crowdsourcing for requirements elicitation.

Language's diversity (BA20) occurs in global software environments. When working on a global software project, the language diversity between large communities and customers poses a threat to effective team-customer collaboration by limiting their understanding of each other's perspectives. Workers with limited ability to read/write a common language are likely to incur greater overhead when communication with others workers and when you put together a diversity of skills and perspectives. This barrier is evidenced by [PRI14] and [MAC16a].

To Prikładnicki et al. [PRI14] languages differences poses challenges to collaborate between for non-native English speakers. English is a universal technical language for software development projects and, the crowd participant's must be able to speak the same language for large communities in order to be more effective communication, learning and collaborative relations [MAC16a].

3.3.4 Discussion

Considering collaboration barriers from different perspectives such as platform, requester, and the crowd, some of these barriers cannot be considered a barrier. For example, from the platform's perspective, *"competition model"* category is a desired "feature" (rather than a "barrier") used by many crowdsourcing platforms such as TopCoder and uTest. In turn, this "feature" can create a collaboration barrier in terms of lateral communication from the crowd's standpoint.

On the other hand, our findings reveal which barriers are common across all the actors, and which ones are unique to each one of them. There are five barriers to collaboration that are common to all actors in SW CS environment: i) Few interactions among involved actors inside the platform, ii) Security and privacy issues, iii) Lack of feedback among actors, iv) Weak commitment between involved actors, and v) Technical and infrastructure setting issues.

The literature shows that lateral communication and coordination are important to most actors in the SW CS. A particular challenge for the crowd and platform is the diverse mix of people. Specifically, for the platforms' side, the instability of resources (control willing crowd workers) represents a huge collaboration barrier, and workers' decisions are highly volatile from task registration to task submission, given that zero or failed submissions may cause negative effects on SW CS projects.

As it was possible to observe through evidence found in the literary (Table 10) the greater number of collaboration barriers refer to *social interaction category* and affects crowd workers. In this way, it is crucial to understand how collaboration barriers limit interaction in terms of lack of communication and restricted communication channels supported by competitive SW CS platform, affecting workers when they try to execute such activity and intent to submit their software solutions. Those collaborative barriers may impact in few or even no participation of tasks, and poor solutions by the crowd, putting SW CS projects at risk.

Table 10 - Collaboration barriers and SW CS elements

ID	Barrier	Crowd	Request	Platform
1	Lack of Informal communication (BA1)	x		
2	Lack of interaction among crowd members (BA2)	x		
3	Information management complexity (BA3)	x		
4	Difficulty for dealing with competitors (BA4)			x
5	Lack of real-time collaboration (BA5)	x		
6	Technical and infrastructure setting issues (BA6)	x		
7	Scarce social media support (BA7)	x		
8	Few interactions among involved parties inside the platform (BA8)	x		
9	Low global project view (BA9)	x		
10	Micro-task decomposition issues (BA10)		x	x
11	Difficulty in team management in large scale distributed settings (BA11)			x
12	Weak commitment between involved parties (BA12)		x	
13	No unified software process methodologies (BA13)		x	
14	Difficulty to increase and keep participants' motivation (BA14)		x	x
15	Asynchronous communication (BA15)	x		
16	Hurdle single point of contact (BA16)		x	

17	Lack or incomplete documentation (BA17)	x		
18	Teams heterogeneity (BA18)			x
19	Weak internal collaboration between the platform and the requester (B19)			x
20	Diversity of languages (BA20)	x		
21	Security and privacy issues (BA21)	x	x	
22	Difficulty to find fit task allocation (BA22)	x		
23	Lack of feedback among participants outside the platform (BA23)	x		

Looking at collaboration barriers in SW CS projects, we evidenced:

a) Collaboration over limited human interaction

The competitive SW CS environment imposes communication limitations. Crowd workers are limited to written communication and lack spontaneous discussion. They have little real-time interaction with each other. By checking literature, we observed that collaboration is strongly connected to message exchange and seeking information about the task on communication forums. No direct communication to share information freely during the development of SW CS tasks may lead to misunderstandings and thus may have a big impact on the quality of the final solution.

The incentive to crowd's interact by side of the SW CS platforms is restricted as mentioned by [PEN14]:

"Crowdsourcing platforms support communication by providing task-specific forums for crowd members to ask questions and communicate with each other... However, crowdsourcing platforms provide little support for collaboration among crowd members."

The study conducted by [GRA16] said:

"(...) when platforms do not natively support collaboration among the crowd, workers create widespread yet invisible forms of collaboration that take place off-platform."

LaToza et al. [LAT4] implemented a *CrowdCode* tool and the findings revealed about communication's issue:

"A majority of participants who worked on a small programming task agreed that the opportunity for communication beyond what was provided would help them to work more effectively. Participants cited a desire to share technical experience, clarify tasks, ask questions about material that others had written."

b) Collaboration over competition

Regarding SW CS that operates on a structure of competitions the level of collaboration and mutual support drastically decreases when crowd community become rivals and compete against each other.

The developers are concerned about their solutions would be stilled and reluctance to exchange information and share knowledge. Differently of open collaboration and free revealing found in the context of open-source software development [HUT11].

The competing behaviour present in CS platform-based contests implies in reduced participating actively in network interaction, sometimes intentionally, provoke by platforms in order to benefit from good performance. Therefore, crowd workers desire to socialize and to interact with others who share similar questions and interests, resources and, information.

Hosseini et al. [HOS15] discuss the effect of the ranking and reputation in the CS competitive model to establish collaborative relationships with other participants, especially among seasoned and newcomer participants:

"The competent crowd might also include participants' inflated egos which would then reduce the level of collaboration and lead to conflicts and inconsistency."

[NAG12] and [NAG13] discuss the applicability of crowdsourcing and competition for problems that require a collaborative or cooperative effort to be successful. The author showed the results of a development software tournament in TopCoder platform.

"Development through competitions requires a careful balance of competition and collaboration to achieve its goals. Much of the collaboration in TopCoder is structured collaboration, i.e. the TopCoder process dictates how that collaboration takes place".

c) Collaboration over dynamic coordination

The coordination in SW CS is complex to manage all the contingencies (e.g. deadline, artifacts, number of tasks, questions from the crowd, uncertainly resources and so on). In this way, coordinate across individuals with different expertise and capabilities require an extra management of the resources [KIT13]. Unstable resources and problems that emerge unpredictably in the course of actions during task's period in SW CS environment have to be resolved through flexible coordination. In SW CS, people don't work together effectively, people must not agree on how the product will be developed. Instead of, would be sufficient to establish effective coordination with each crowd worker para melhor o quantidade e qualidade de submissoes de solucoes.

In Stol and Fitzgerald, [STO14a] the authors reports:

"Multiple tasks in parallel have implications for coordination and quality in order to attract sufficient crowd participants with required skill to be allocate or assignment on crowdsourced tasks".

In [RIE17] the authors investigate how to effectively incorporate team-based collaboration in a setting that has been individual-based. In a field experiment on online platforms they evaluate the influence of member skills, incentives and emergent collaboration process on performance of crowd-based teams. The authors evidenced this barrier through of coordination:

"(...) temporal patterns of coordinating are a particular challenge in crowd-based teams, as, in addition to being globally distributed participants also tend to touch their

contributions in around the edges of their "regular" activities, leading to an even less regular schedule than we would observe in global teams in work organizations".

3.4 Case Study by Crowds' participants

3.4.1 Settings and methodology

After the literature review study, as illustrated in the first chapter (research design), we decided to focus on competitive SW CS, which is a model of CS in software development [LAT16]. Then, we decided to gather more evidence of the research problem we were addressing. We conducted a case study (Figure 1), to explore how crowd participants perceived collaboration in a competitive SW CS platform, what collaboration challenges they faced to perform a single task during a contest, the suggestions to overcome the challenges based on their experience and, reflecting upon their feedback from which collaboration, or the lack of it, influenced the submission of task solutions during a TopCoder challenge, [MAC17]. Therefore, it was in the interest of the research to provide the participation of computer professionals in a new approach to development, and, more importantly, to understand what practical implications of collaboration barriers are faced by participants in SW CS challenges.

We found that for most participants, the perception of collaboration among crowd members during task performance was weak, restrict, and indirect. Besides that, collaboration was strongly connected to message exchange and seeking information about the task on communication forums. There was evidence that reduced collaboration impacts crowd workers in a negative way when it comes to performing the task and completing the challenge. After conducting this study, we defined the collected data directly from communication forums from TopCoder challenges and started Phase II studies of this thesis to verify if lack of or reduced communication would influence in the crowd workers' decision-making process to drop out or not submit the tasks' solution.

According to [HOP96], the case study is particularly suitable for the exploratory examination of phenomena that have not yet been studied and that need to be investigated in their environment of occurrence. The application of this method is indicated when one has to learn about the state of the art and generate new theories supported in practice, to understand the nature and complexity of the process as it happens, and to bring new factors and information, evidenced during the execution of the studied process [YIN13].

Being case studies empirical investigations especially employed when the boundaries between the contemporary phenomenon and the actual occurrence context are not clearly evident [YIN13], the exploratory case study was defined as part of the strategy of this thesis to bring to life the real-world phenomenon in a contemporary software development context.

The case study took place as part of a graduate course project on Collaborative Software Development (CSD) at PUCRS, Brazil (Appendix D). The course discusses the

history of collaborative systems and CSCW principles, as well as different models of collaborative software development including global software engineering and software crowdsourcing. We selected TopCoder as the platform for the project to take place in the Development Challenge (DC) category to work on.

The course is offered once a year during 16 weeks (during August and December in 2016), has, in average, about 15 students enrolled per session.

The students were told they had to work independently and that they have six weeks to conclude the SW CS task between August and December in 2016, and then submit they would have to submit it on the platform. Besides that, they were given two additional weeks to observe and respond to any feedback, if given, from the platform. An assignment was associated to each of the above-mentioned activities as follows:

- a) Report 1 - Task Description: the student reported on the selected task and provided us with a brief description of the task goal, sub-category within the Development Challenge category, and estimated period of completion. A brief explanation on the reasons for selecting this task was also provided;
- b) Report 2 – Open experience report: the student had to respond to nine open questions (Appendix C), where students were able to debrief and explain their SW CS experience in terms of collaboration activities and barriers about the onboarding process while trying to place their code solution on Topcoder;
- c) Report 3 – Open-ended online questionnaire: this questionnaire provided us with profiling information on the students' background and Likert-scale-based questions about their opinion on the course project.

As the case study was embedded in a larger project that involved other research topics related to SW CS (task's requirements, newcomers' challenges and motivation), only Report 2 was used in order to answer the defined research questions for the sake of this thesis. Thus, of the total of nine questions presented to the group of participants, only three of them correspond to the objectives of this study and will, therefore, be reported in this section.

For validation of the questions used to compose Report 2, face-content validation was performed by two research colleagues in SW CS domain, and the course teacher who has extensive experience in collaborative software development and SW CS. The validation phase contributed to the refinement of the questions that made up the final version of the questionnaire. The original questions were reordered and rewritten to allow greater consistency in the terminology adopted by the platform.

From the group of 21 students, 16 (76%) participated in a SW CS initiative for the first time, that is, they submitted a SW CS task on Topcoder for the first time, whereas some of them (24%) had already submitted SW CS task solution in other platforms such as Upwork, and People per Hour²³, besides TopCoder. The 13 graduate students (62%), have had an average of over six years of software industry experience, ranging between 26 and 30 years

²³ <https://www.peopleperhour.com/>

old. The main professional activities mentioned were programming (57%), and system analysis (24%). The other activities comprised project manager and tester jobs.

The students took part in “F2F” and “Code” tasks on TopCoder, which means that only these two types had been chosen from DC category. A task in the DC category can be categorized in several subcategories, as mentioned in subsection 2.7, such as: architecture, assembly, code, component design, component development, First to Finish (F2F), etc.

3.4.2 Data Analysis

The data from the “Case Study” were collected through the analysis of the open questions which guided the elaboration of the reports delivered by the participants of the study, and also complemented by the group discussion among the participants. The data were used to: 1) identify relevant codes in the context of collaboration in SW CS, 2) analyze the relationship between these codes and 3) identify the categories to group these previously found codes. The synthesis of these categories is a list of the main collaboration barriers associated with SW CS contest, more specifically the collaboration barriers associated with the TopCoder platform, as well as the characterization of the collaboration in this context illustrated from the selection of significant sections of the answers that mention them. As described, the case study was divided into three main deliveries; however, for the specific objectives defined in the exploratory case study of this thesis, only part of Report 2 was used.

Similarly, to our previous study, we analyzed the open-ended questions using coding techniques from Grounded Theory [STR07]. We integrated the results from the literature study adopting a unique set of codes (category and topics), extracting the most significant results, that is, discovering, through the conditions in which phenomena occur, the barriers faced in crowdsourcing software activities.

Most of the categories identified showed a consensus with the LR performed in study 2 of the exploratory phase of this thesis. The categories patterns obtained through data analysis were related to collaborative forces or barriers that include: restricted communication between the involved parties during the execution of the task, the competitive nature of the SW CS platform, and unclear task documentation, as described in the following section.

3.4.3 Results

The most evident result in the case study was reported by the crowd members in the recognition of collaboration on the TopCoder platform through asynchronous communication via forum, and, in the same way, the most significant collaboration problems were mentioned to be the restricted communication via forum between members and the platform, and the competitive nature of software development.

The question related to participants' collaboration perception was as follows: *“How did you perceive collaboration in SW CS on Topcoder among crowd members, and between the platform (requester) and its members?”*

The collaboration perception in SW CS on the platform and the task execution sought to investigate how and when the involved parties (crowd, platform, and requester) collaborate in the work setting. As a consequence, this feedback helped us understand to what extent SW CS competitive model allows and requires collaboration during task execution among members, contributing to the same software product development. Findings suggest that the collaboration perception in SW CS competitive model was fuzzy, which was gathered through three answers given by them: those who perceived collaboration, those who partially perceived it, and the ones who did not perceive collaboration while participating in the task execution on Topcoder.

Some quotes are presented, referring to the answers from those who perceived collaboration clearly or partially, which are strongly connected to message exchange, and seeking information about the task on communication forums.

a) Among platform members:

“Collaboration on the platform, as far as I have noticed, works through messaging forums, where developers ask direct questions to the requester, and the answers can be viewed by everyone who is attending that particular task and can also make use of that information”. – P14

“... I noticed that other colleagues had already identified problems in the archives and had exchanged information that was useful to carry out the activity.” – P6

“Among the members who were developing solutions to the problem posted, I could see collaboration through the platform to clarify doubts and get reference materials that could help in obtaining relevant information to the development of the solution.” – P13

b) Between the platform (or requester) and platform members:

For most tasks performed in the study, participants reported the “presence” of a copilot in the forums, which may justify the responses/feedback from participants about the perception of collaboration more explicitly between the copilot and the crowd to refine the scope and support doubts about task documentation.

“Collaboration happens better between the requester and the platform members with the use of forums.” – P9

“(...) Between the requester and the user, there is certain collaboration through a forum that the platform makes available, but this collaboration is more used to document the task, and check on doubts about some item of the task that was not clear” – P12

The email tool was perceived as a collaboration between the platform and the crowd, cited by only one of the participants. Emails were mentioned mainly in the post submission review process of the solution developed for the task.

"The information that the platform makes available to the developer refers to reviews of the task going through and the deadline. Other information I received from the platform was through emails about the process, but most of the time they were repeated." – P14

Regarding those participants who reported not noticing collaboration in SW CS, it was observed that this fact is intrinsically bound to three main reasons: (i) not working together with other members to write code, (ii) not using the forum to communicate during task development, and (iii) competition inhibiting collaboration.

"On the challenge forums, I did not identify any kind of collaboration regarding coding"
– P1

"I could not visualize the collaboration within the platform, at least not in the task I performed." – P16

"In fact, the very nature of the platform, competition, makes this type of collaboration discouraged, which in my opinion is something negative." – P4

Communication as a way to collaborate in Competitive SW CS

It is noteworthy that, through the patterns found in the analyzed data, the communication via forum on TopCoder platform is seen as a synonym of collaboration in the task development process according to crowd workers. Thus, the forum – considered as an asynchronous tool of communication among task participants (crowd), and between the task mediator (*copilot*) and the crowd – represents the most used means to share task specification documents, and exchange information. Furthermore, narrowing down to TopCoder, forums are used as a task communication and coordination resource, and they are just visible to those participants who have registered to the task. From this moment onward, participants can post messages or simply read and gather information from the posts of other task participants.

"I also read some posts with questions from other developers." – P8

"(...) A participant's question can be shared and answered only by reading the forum."
– P9

"I resorted to the forum to find more details, since the description of the activity was very succinct." - P8

The lack of collaboration through communication problems on the platform was mentioned by the participants. For most of them, the perception of collaboration among crowd members during the task performance was weak, what might have been influenced by the context of TopCoder's online contest (competition), the financial reward, and the time to be the first to send the best solution, as mentioned by some participants in this study.

"I only noticed the collaboration between the requester and developers through the forum. And I think this is the downside of the model adopted by Topcoder. As work is conducted as a competition, collaboration among members is minimized." – P20

"I did not observe collaboration among users, they are trying to finish the task as soon as possible so that they can get the reward." – P19

For example, participant P6 reported:

"Collaboration was undoubtedly essential, and it worked objectively. Specifically, in the activity I chose, I did not find the files to download and I searched for information on the task's Forum. Not only did I find the files but I realized that other colleagues had already identified problems in the files and had exchanged information that was useful for the execution of the task." – P6

In addition, another participant states:

"Collaboration among members of TopCoder platform is non-existent; identifying other members on the platform is very difficult. Between the requester / platform and the user, there is certain collaboration through a forum the platform offers, but this collaboration is more used for task documentation, and clarifying doubts about some items of the task that were not clear." – P1

This answer indicated that there are interaction and communication among involved participants but, in the participant's point of view, collaboration is more significant between platform and requester. This quote can be associated with specific characteristics of the platform in terms of anonymity of the requester and crowd community and, lack of social ties and informal communication inside the platform.

Collaboration Barriers in SW CS Projects

The questions related to participant's collaboration challenges was as followed: *"What collaboration difficulties did you face on TopCoder platform?"*

- a) Analysis considering competitive behavior vs. collaborative behavior

Open calls for competing solutions in SW CS projects apply a model of independent and individual work that can inhibit the opportunity for developers of the crowd to collaborate in exchanging information, sharing experience, ideas, and so on. It was observed in results that some SW CS activities are performed individually by crowd members, i.e., individual behavior does not require other developers' participation (e.g., coding, reading documentation, onboarding task process), whereas collaborative behavior requires more than one person involved to interact, for instance, decision-making, informal conversations, chat in real time communication, and email messages. Thus, during the execution of SW CS tasks, the collaboration takes on new characteristics in this competitive model, influenced by the transitory, remote, disperse, undefined, and unknown/anonymous aspects of the crowd, making it easier to recognize individual behavior rather than collaborative behavior among platform members who share an award for the best submitted solution as referenced by Machado et al. [MAC17].

As mentioned by [HER99] and [HER01] global distance affects collaboration in the process of software development, and, given that SW CS development operates on a structure of competitions [5], these can impose a restriction to crowd members to collaborate

with one another. In this way, we were surprised that participants reported about collaboration during their experience in SW CS tasks on TopCoder.

Thus, new aspects to coordinate distributed individuals online and manage the flow of communication emerge in a competitive SW CS context: Shall we expect collaboration? Shall we reconsider what we know? Tasks are being defined to a single developer to work on his/her own, but is this possible?

b) Analysis on the use of asynchronous communication to interact and collaborate

Informal communication among crowd members during tasks' execution is restricted to asynchronous tools, according to what was observed in this study. Asynchronous communication via forum in each task performed by the participants of the study reinforces that interaction among crowd members and the copilot may impact collaboration for task development. We may think that the platform intentionally supports communication among members of the crowd in a restricted way, by considering the competitive model it acts. No direct communication during the development of SW CS tasks may lead to misunderstandings and thus may have a significant impact on the low solution submission [YAN16], and the quality of the final solution [STO14], [PRI14].

Asynchronous interactions do not provide rapid feedback among members. For instance, in forums, feedback communication is delayed, and interruptions or long pauses in communication often occur [BOU14]. However, forums have a considerable influence on the interaction for communication during a SW CS task execution on TopCoder, as we have found in the results. In connection to the usage of asynchronous communication to collaborate and interact in this platform, it becomes possible to think if forum resource in competitive SW CS represents a particular collaboration way among the involved parties (crowd, requester, and platform) to development software tasks in this context.

Once again, we merged some barriers, reorganized them, and added new ones (**Table 11**). The categories and topics were obtained after coding analysis. By categories we mean the group of the concepts of the phenomenon (collaboration in SW CS). Meanwhile, topics were used to classify each collaboration barrier that crowd workers face in SW CS challenges. This analysis happened with the open questions reported by specifically crowd workers after registered on TopCoder platform.

Table 11 - Collaborations barriers from crowd

Case study categories	Topics emerged from the case study	Literature review categories
Communication	<ul style="list-style-type: none"> • Communication only via forum and restricted between crowd - crowd • Communication focused on the task (without informal communication) • No communication between crowd and requester 	

	<ul style="list-style-type: none"> • Integration with other collaboration tools (GitHub, stackoverflow) • Long time responses on the forum for doubts • Review process without interaction with stakeholders 	Social interaction
Visibility	<ul style="list-style-type: none"> • No synchronization of decisions taken in forum vs. documentation (and vice versa) • Evolution transparency in each phase of the challenge 	
Task	<ul style="list-style-type: none"> • Unclear and inconsistent documentation • Setting up the environment for task implementation • Documentation with little detail • Task onboarding 	Task design
Competition	<ul style="list-style-type: none"> • Competition discouraging collaboration • Tendency for the oldest ones (seasoned TopCoder members) to compete and win • Financial reward 	Competition model
Cultural	<ul style="list-style-type: none"> • Language difficulty 	Cultural diversity
Process	<ul style="list-style-type: none"> • Feedback (code review) without personalization • Information management • Platform Onboarding 	Process management

Crowd workers in SW CS contests in relation to the competition model consider the environment setting as a collaboration barrier, and this might be due to the fact that the platform does not use collaboration tools like code hosting (GitHub, BitBucket) for task development. Likewise, regarding the barrier related to the category "Process Management" and to the theme feedback without personalization, and technical code review without tool support, participants pointed out:

"(...) On the development environment, this was one of the factors that I considered negative, since each task (not being tasks from the same project) requires a new environment setting, which ends up taking a lot of time." – P1

"I was also hoping that the review process would be open to everyone, and that I could review the code that other people submitted. (see day 17/10/2016 of the logbook). What happened in the review process were two people reviewing the deliverable of everyone... the code review process could be improved if there was a code review tool where I could see notes in the code of which part should be improved". – P1

About the competitive nature of TopCoder contest and the advantage that older platform members have over newcomers, the participants said:

" To what extent will new developers be encouraged to join the platform, since more experienced developers have a tendency to always win the challenges (there is a cycle of the same people always winning the challenges)?" – P8

"Tendency for the same participants to always win the challenges" – P14

Regarding the difficulty to communicate in another language, besides other barriers already evidenced in the study, the participant comments:

"Difficulties can happen, such as assembling the development environment, writing and reading in another language, or in the usability of a tool, which can be minimized with study and practice. All knowledge acquired in each project, be it at work or in the accomplishment of a course, adds to the next one." - P18

3.4.4 Discussion

The feedback from the exploratory case study helped us understand to what extent SW CS competitive model allows and requires collaboration during the task execution among members, contributing to the same software product development. Besides that, the study also helped characterizing the communication behavior on TopCoder. Furthermore, we investigated how developers collaborated with each other in a competitive work setting, and what the main collaboration challenges faced by them were in this context. As contribution, some preliminary suggestions to the collaboration challenges in SW CS under the crowd perspective were provided through data analysis from the exploratory case study. These suggestions should provide subsidies to improve the platform requirements to be more assertive in setting collaboration aspects, onboarding processes, and increase submission solutions in a satisfactory way. The unexpected collaboration between crowd competitors can be mentioned as good aspects in the SW CS onboard process. On the other hand, there is the need to improve the communication usability of the platform, adding a particular focus on improving the communication channels.

As limitations, our study concentrated in one crowdsourcing platform, and the strategy of offering to participants of the study only one option of challenge category on Topcoder (Development Challenge) for task selection led many of them to choose the same kind of task, which was, in this case, First to Finish (F2F). This recommendation also led many of them to carry out, even if in an individual way, the same task. Therefore, we cannot claim the generalizability of the results.

Finally, this empirical study, aimed to understand that collaboration assumes specific characteristics in SW CS contests, and that they should impact the involved parties in a positive and negative way. Afterwards, we analyzed the outcomes (coding of collaboration barriers) obtained from the qualitative studies interviews (study 1) and case study (study 3) with the literature review (study 2), which each barrier emerged from these studies was used as the input for the mapping and resulting in the collaboration barriers' model in SW CS presented in Figure 12, along 29 collaboration barriers. The analysis process was similar for all studies, where we found categories and codes identified during each study. The

categories represent the main aspects associated to collaboration in SW CS projects and these barriers were classified into five categories once them: social interaction, process management, cultural diversity, competition model, and task design. Regarding to codes, they represent the collaboration barriers emerged in each study and, they were refined, merged and rearranged to accommodating all the barriers evidenced from of the three studies executed in the exploratory research's phase. This step was need because we observed an overlap between some barriers and also, a group of barriers displayed similar proprieties and dimensions.

Therefore, the collaboration barriers' model presented in **Figure 12** reflects the final structure observed on SW CS collaboration barriers following the three qualitative studies.

Based on the categorization proposed in the collaboration barriers model in SW CS, we evaluate the *social interaction* category focusing on *communication issues* among crowd workers of the largest SW CS platform that operates in a competition model - TopCoder. In this way, it was possible to cover the other categories of the model as well, since we observed through the obtained results with the studies carried out in the exploratory phase, a strong dependence and interrelationship between cultural diversity, task design, and process management during the execution of software tasks in the SW CS projects.

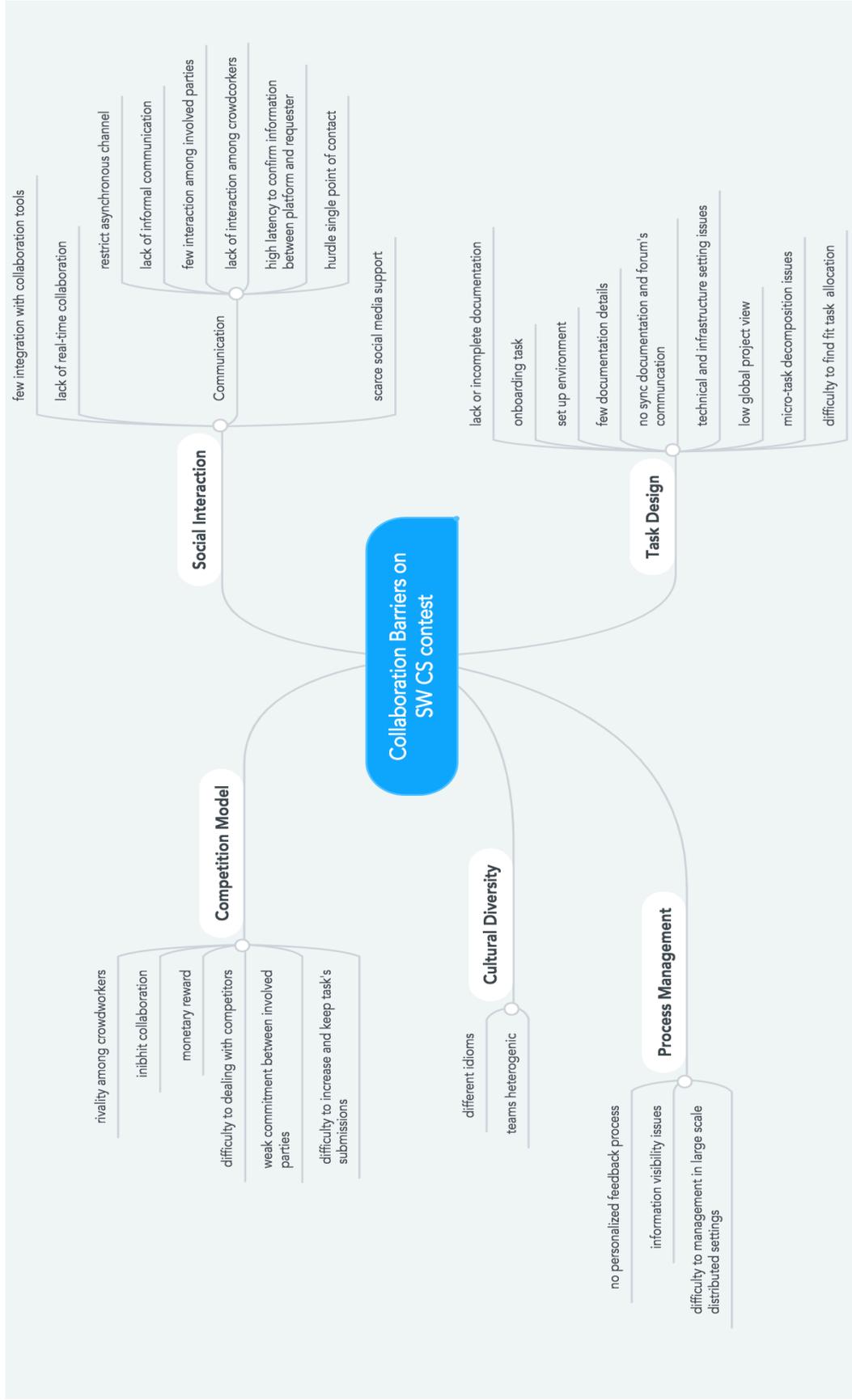


Figure 12 - Collaboration barriers model

Figure 13, present the barriers emerged from each study. We highlighted with different color each barrier per source. The two (2) ellipses in gray displays qualitative analysis interviews (study 1), the ellipse in red present 20 barriers identified by literature review (study 2) and, the exploratory case study (study 3) is represented by 16 yellow ellipses. In this final collaboration model, some barriers overlapped the studies, as show Figure 13, where the barriers are displayed with two colors. For example, in *Social Interaction* the barriers overlapped are: lack of informal communication, few interaction among involved parties and lack of interaction among crowdworkers.



Figure 13 - Model identified per study

4. COLLABORATION CHARACTERISTICS

4.1 Evaluatory Phase

In the evaluatory phase, we identified an analytical framework to measure collaboration characteristics among crowd workers, and the “fit” between these collaboration characteristics and the productivity (winners) by the crowd. This measure of fitness is called congruence [CAT06] and has been studied in both industrial [CAT06], [KWA11] and open-source [WAG05] software development projects. Using qualitative data from TopCoder’s platform forums that were analyzed using GT procedures [STR07], in this study, we found that patterns of collaboration among crowd workers influence on task performance, suggesting that crowd workers who communicate via forum is important for task’s completion in the TopCoder challenges, in other words, there is also congruence on SW CS for the task performance and winner of the challenge.

The second study was a survey with developers who have competed on TopCoder and it aimed to validate the results of the qualitative forum analysis. Questions from this survey focused on collaboration, task performance and communication channels used by crowd workers during task execution. We used correlations and qualitative approach for analysis of the questions [KIT02a], [KIT02b], [KIT02c], [KIT02d], [KIT03].

In the next sections, we provide more details about the design and analysis results of the empirical studies based on real-world data gathered on the Topcoder’s communication forums, and the collaboration in the SW CS survey with crowd workers who had participated in the development challenge on the same platform.

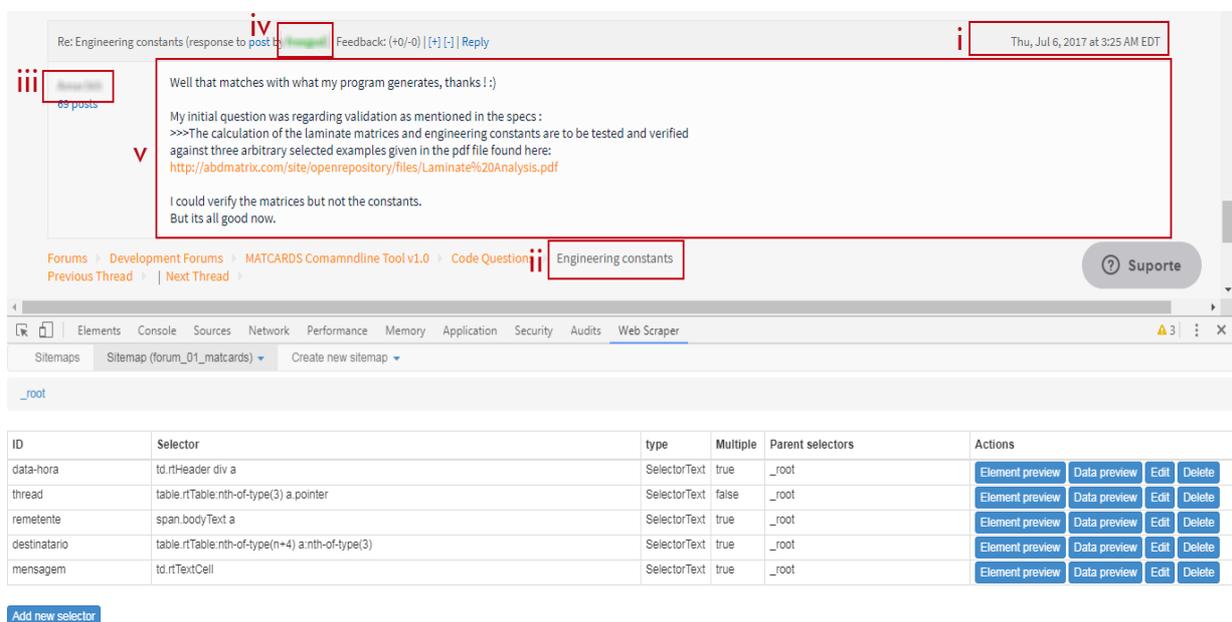
4.2. Qualitative analysis of communication’s forums

4.2.1 Settings and methodology

As mentioned, one of the objectives of this thesis was to analyze the collaboration between crowd workers in competition challenges on the TopCoder platform. This was done by qualitatively analyzing the messages exchanged between the crowd workers, here defined as coders, on the forums that are created for the challenges.

The TopCoder’s forums analysis were carried out in partnership with Ricardo Marinho during her Master course. In the partnership’s work, he analyzed *how much* crowd workers communicate in the SW CS forum’s challenges and it served as input for the extension of the messages analysis of TopCoder’s forum executed by the author of this thesis. Here, we analyze *what* crowd workers communicated about, i.e., the categories and topics identified in their messages. Thus, we presented the collaboration characteristics described in subsection 4.3 of this thesis.

To extract the data from the forums, a tool called Web Scraper²⁴ [MEL18] was used. This tool allows the creation of sitemaps to navigate the page and extract data. Unlike other tools that extract data only from HTML, Web Scraper can also extract data that is loaded or dynamically generated with JavaScript. Using different types of selectors, Web Scraper extracts various types of data - text, tables, images, links, among others. The data, once extracted, could be exported as CSV. It was necessary to register to participate in the development challenges, in tasks of the Code type, from July to August 2017, meeting the established criteria, as defined in **Table 12**. Thus, it was possible to access the data related to the messages among active coders in the discussions, available in the project area, called challenge forum. With this participation, data were extracted to return textual results, such as: date and time (i), thread (ii), sender (iii), recipient (iv), and finally the message itself (v), as shown in **Figure 14**



ID	Selector	type	Multiple	Parent selectors	Actions
data-hora	td.rHeader div a	SelectorText	true	_root	Element preview Data preview Edit Delete
thread	table.rTable:nth-of-type(3) a.pointer	SelectorText	false	_root	Element preview Data preview Edit Delete
remetente	span.bodyText a	SelectorText	true	_root	Element preview Data preview Edit Delete
destinatario	table.rTable:nth-of-type(n+4) a:nth-of-type(3)	SelectorText	true	_root	Element preview Data preview Edit Delete
mensagemem	td.rTableCell	SelectorText	true	_root	Element preview Data preview Edit Delete

Figure 14 -Example of selectors used for data collection

The discussion forums on TopCoder are normally used for sharing task documentation. Besides that, it is the channel where questions are asked and answered by platform moderators (copilots) and among all crowd participants (coders) registered in the challenge. The prevalent questions are related to task's documentation such as, communicating useful information (technical and managerial), removing requirements' misconceptions, setting deadlines, solving understanding problems on documentation and so on.

On the forum, there is a feature that allows users to click on the "feedback" link to mark posts with useful information, and a feature to manage forum "views" by the crowd participating in the competition. The number of feedbacks to each post is visible to all forum registers. We might contrast coders who are active forum members from those who are not.

²⁴ <http://webscraper.io/>

However, we cannot track coders' viewing log, so it is hard to see which coders combined knowledge from the forum without speaking up (i.e., posting a post). We consider these interactions via forum as acts of socialization and collaboration among the involved parties during a SW CS competition.

All coders that participated in task forum discussions represent the TopCoder members who had registered in one of the 25 challenges analyzed. Some coders may have registered for more than one challenge at a time.

Coders have different levels of participation during the challenges they perform. Yang et al. [YAN16] see crowd participants as a set of quitters, submitters, winners, and uninterested, that is:

- *Quitter*: includes crowd workers who did not make submissions to a task they once signed up for, by the given deadline;
- *Submitter*: includes workers who have submitted a task but did not win the competition;
- *Winner*: is a crowd worker who has submitted a piece of code for the stated task and was evaluated as the winner or runner-up among all submissions;
- *Uninterested*: comprises those workers who were active in the TopCoder platform but did not register for a task.

Despite the difference, all members are important to the success of the SW CS strategy. It is evident that, by making direct submission and winning a challenge to the software development at platform, winners and submitters are vital to SW CS development, but it is essential to consider that quitters and uninterested crowd workers are also the source of new submitter members, so supporting them is important to the long-term success of SW CS.

In this thesis, we employ the first three coder status/category, namely, quitter, submitter and winner [YAN16], as the labels on competitor outcomes that describe crowd participation in SW CS tasks. While the reasons for different crowd participants to quit of a competition may be complex, in this thesis we investigated the correlation of collaboration among crowd members and different levels of participation (task performance) of winners, submitters, and quitters during a task competition.

4.2.2 Data Collection

From several development challenges (SW CS tasks) hosted on Topcoder [TOP17], we selected a sample of challenges for analysis. The selection of the sample was based on the criteria as can see in the Table 12.

Table 12 - Summary of criteria definitions

Category	Metric	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 budget. The value the task requester is willing to pay
iv. Task phase	Registration and submission	Indicating the task status (registration, submission, review)
v. Task challenge	Development Challenges	Indicating the challenge area (design, development, and data science)
vi. Task category	Code	Capturing the different categories for tasks (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. [DUB16] study, where the authors analyzed July and August as the two months when the maximum number of tasks are posted on TopCoder. We have also restricted our sample to tasks with at least fifteen (15) registrants in the challenges with a minimum of financial reward (500 dollars). For these criteria (ii) and (iii) were considered the reasonable number to generate a forum discussion within medium reward to willing competitors.

Following the sampling criteria, our final sample consists of 25 challenges, more specifically, technical coding challenges (vi). Collecting detailed data by content post from communication forums is an arduous manually process. But, as far as we know, no previous work has reported content analysis from TopCoder's forum as a strategy of research and analysis in the SW CS area.

4.2.3 Data Analysis

The content of each message was analyzed using GT procedures [STR07] to identify categories and topics. By categories we mean the type of message being post including, public announcements, reported problems, tips, etc. (Table 13). Meanwhile, topics were used to better classify the message, i.e., the subjects being discussed in the messages. The

messages (conversations) codified into categories are present in Table 14, and the topics can be seen in Table 15 [MEL18].

The coding process was conducted separately by the author of this thesis and Ricardo (partnership's researcher) in his dissertation between December 2017 and February 2018. Then, they together reviewed every category and topic to reach an agreement about the final categories and topics to be used. The final coding scheme included the following categories: Public Announcement, Tips, Request for Help, Help Answers, Confirmation Request, Confirmation Response, Invitation Request, Invitation Response, Identified Problem, and Problem Response. As for the topics, they are: Access, Library, Connection, Deadline, Inputs, Style, Processing, Requirements, Outputs, Scorecard, and Units.

Table 13 - Examples from category coding

Categories	Messages
Public Announcement	<i>"[...] The purpose of this document should be to train the ML system to identify data in the schedule."</i>
Request for Help	<i>"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?"</i>
Problem Response	<i>"[...] You should first add corresponding class into .scss file, save that, and only then use it in .jsx. I should check, whether this behaviors can be improved."</i>

We defined ten categories that represent the message type "source-destination". For example, the forum thread can be posted by a copilot or coder (sender) and destined for a different copilot or other coders. The response(s) to this message can be given via reply by the copilot or by the coders themselves.

Table 14 – Message Categories

Coding 1 - Category	Description
i - Public Announcement	It has an informative aspect, it does not characterize a problem or solution. It usually starts a thread by the copilot. The Copilot provides technical information about the task through links and the documents created for the task in the forum.
ii - Tips	It has a solution aspect for a possible question or problem, in a way that it suggests to solve a certain request. The copilot or coders themselves pass on task specification details based on the forum participants' questions.

iii - Request for Help	They are usually open-ended questions, asking for an answer to solve and clarify a problem. Most of them are “W questions”.
vii – Help Answers	Request for Help Responses.
iv - Confirmation Request	They are usually closed questions, just to confirm the information already mentioned. Most of them are "Yes/No questions".
viii - Confirmation Response	Response of the Confirmation Request for the approval of using certain tools, libraries, etc. which are not formally described in the task specification.
v - Invitation Request	It has a requisition aspect for certain access on some platform. Most of them sends emails to users of any platform, requesting to be added. Request to provide certain file access, or private key.
ix - Invitation Response	It is a reply to the Invitation Response (for access or connection)
vi - Identified Problem	It has an informative aspect and it characterizes a problem. It does not request a response directly, nor does it present questions.
x - Problem Response	It is a Response to an Identified Problem (it points out inaccuracies in zip files, documentation, etc., associated with the task)

Table 15 – Message Topics

Coding 2 - Topic	Description
i - Access	It usually characterizes topics from code repositories, or from a given platform. Access to certain data, private key, and directories.
ii - Library	In this topic, technical aspects of several libraries / frameworks, APIs, plug-ins, tools, code repositories, components and code specific to each task are highlighted.
iii - Connection	It is usually characterized by issues that involve login and password to connect within the application itself.
iv - Deadline	Deadlines established for the delivery of solutions and the time / latency of return between a request or response are discussed.
v - Inputs	It is characterized by input variables to be established.

vi - Style	This topic discusses formatting styles, subjects related to the interface or application frontend.
vii - Processing	It is characterized by the compilation of codes/errors of execution in the application and in servers, and case tests.
viii - Requirements	It refers to the requirements' specification document, involving input and output issues. It also includes the artifacts and files used to support the documentation located on Github, Google Drive, and Dropbox.
ix - Output	It is characterized by comparing the results of an application.
x - Scorecard	In this subject, the relevant scores for the classification of the solution are discussed.
xi - Units	It considers units of measure of a given variable.

Figure 15 - Example of coding from messages in the forum

Figure 15 presents an example of coding procedure from an Excel screenshot. It contains an excerpt of messages exchanged among the involved participants in the TopCoder challenge. The columns describe date and time, thread, sender, recipient, message and applied coding. For instance, in the sender and recipient's column, we identified the cells in red as the copilots, green cells as the winning coder, and yellow and transparent represent, respectively, the submitter and quitter coders who communicated in the forum of the analyzed challenges.

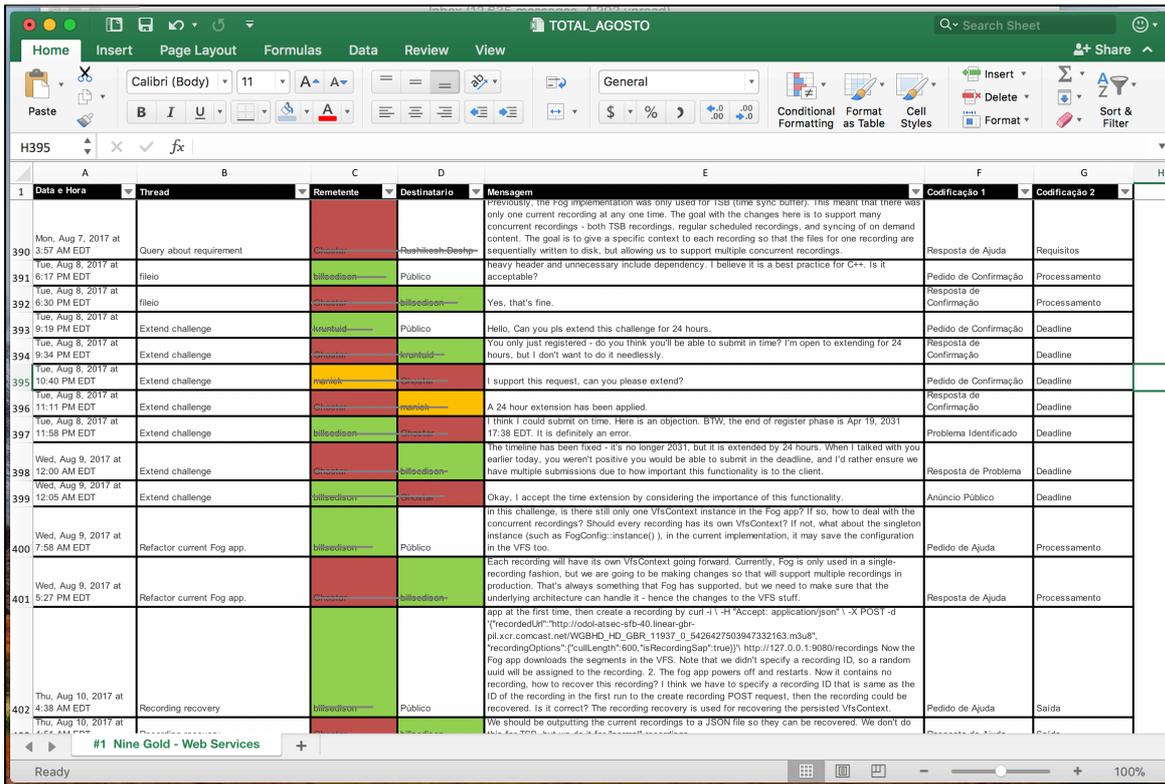


Figure 15 - Example of coding from messages in the forum

All forum messages were coded in a similar fashion, with categories being defined at a level of detail that could allow us to understand collaboration barriers participants face, and the characteristics of this collaboration, including who the participants are more likely to collaborate with, and how this collaboration correlated with participants' performance in the contests. During the next step, those categories were broken into topics so that eleven topics were encoded, as follows (Table 15): units, style, inputs, outputs, processing, connection, library, access, requirements, deadline, and scorecard.

Table 16 - Examples from topics coding

Topics	Messages
Requirements	<i>"In the specification it is given, the program shall use a standard directory as follows: i;- README.txt Should we need to give txt read me file or Markdown read me file?"</i>
Processing	<i>"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."</i>
Deadline	<i>"Is it possible to provide an additional extension of just 24h? I am definitely going to submit this"</i>
Library	<i>"Do you really need D3JS here? Why don't you want just render svg elements using the standard ReactJS?"</i>

Communication forums keep a set of information capturing the questions and answers that took place during TopCoder's registration and submission phases as mentioned in the section 2.7. The communications that occurred in the review and appeal phases that integrate the workflow of the platform are also directed to the forum, but in smaller number.

4.3 Initial Results

According the criteria presented in Table 12, our communication's analysis in the initial stage examined the messages exchanged among coders and co pilots during development SW CS challenges hosted on TopCoder as describe section 4.3.1. Afterwards, we restricted our forums' communication analysis just between coder and coder. The results about coder's communication, performance and pattern are detailed describe in the 4.4 and 4.5 sections.

4.3.1 Coders and Copilots collaboration

From the analysis conducted according to the methodology described, the following results were obtained. As mentioned, a subset of this analysis was published at [MEL18]. In general, 1,184 messages were collected, and, regarding the registration and submission phases criterion (iv), defined in Table 12, 1,053 messages were analyzed between July and August 2017 (496 messages sent by copilots and 557 messages sent by the crowd). These messages, in total, were sent by 120 active coders in the forums (11 copilots and 109 crowd members), distributed among 216 threads in 25 challenges of the category Development and subcategory Code during the period of analysis.

The average number of coders registered by challenge was 37,32 (at least 17 coders and at most 62 coders). Moreover, the average number of messages sent per challenge was 42.12 (a minimum of 2 messages and a maximum of 122 messages), while the average number of threads was 10.76 (a minimum of 2 threads and a maximum of 24 threads). The average total award is \$1,775 with a maximum of \$3.500 and a minimum award of \$900, and, on average, most of the tasks last 5.84 days from the first day of registration to the submissions deadline (minimum of 4 days and maximum of 10 days). 40% of the challenges have a number of registered coders less than or equal to 31, with the number of submissions less than or equal to 4. On the other hand, 50% of the total challenges received more than 34 registered coders. In short, it is possible to observe that the forums are somewhat active despite the short duration of most tasks. In addition, a large number of coders register to the challenges, but only a small part of them effectively submits a solution.

Table 17 describes the number of coders, threads and messages in each challenge, the duration in days, and the awards in dollars (\$).

From the messages sent via forum, the copilots presented a thread participation rate of more than 90%. This is an expected outcome, because as mentioned in Chapter 2, this is the copilot's role (messages are focused on solving the task without much informal communication). On the other hand, even in a competitive environment, crowd members can still construct a dialogue with a certain reciprocity in the exchange of information in each thread, given the presented average of coders per thread of 2.8 coders (minimum of 1 coder and maximum of 12 coders). One should also note that each thread had an average number of 10,76 messages (minimum=2 and maximum=24).

Table 17 - Challenge data

Challenge	Coders	Duration	Threads	Msgs.	Reward	1st position	2nd position	3rd position
DJ1	62	8 days	14	102	\$1500	\$1000	\$500	-
DJ2	43	5 days	7	21	\$1500	\$1000	\$500	-
DJ3	34	5 days	12	60	\$1800	\$1200	\$600	-
DJ4	42	6 days	5	18	\$1600	\$1100	\$500	-
DJ5	33	5 days	8	43	\$900	\$600	\$300	-
DJ6	52	7 days	13	23	\$3500	\$2000	\$1000	\$500
DJ7	27	5 days	4	8	\$1200	\$800	\$400	-
DJ8	43	6 days	20	76	\$1500	\$1000	\$500	-
DJ9	48	6 days	11	19	\$2800	\$1500	\$700	\$600
DA1	47	5 days	10	35	\$1200	\$775	\$425	-
DA2	48	9 days	24	115	\$3000	\$2000	\$1000	-
DA3	17	4 days	17	71	\$1500	\$1000	\$500	-
DA4	28	5 days	8	27	\$1200	\$800	\$400	-
DA5	26	5 days	3	7	\$1125	\$750	\$375	-
DA6	19	7 days	8	47	\$1200	\$800	\$400	-
DA7	43	5 days	15	32	\$2400	\$1600	\$800	-
DA8	33	5 days	6	22	\$1800	\$1200	\$600	-
DA9	26	5 days	9	29	\$1500	\$1000	\$500	-
DA10	20	6 days	9	22	\$2250	\$1500	\$750	-
DA11	50	10 days	17	36	\$2400	\$1500	\$600	\$300
DA12	24	4 days	2	2	\$1200	\$800	\$400	-
DA13	52	5 days	16	122	\$2100	\$1400	\$700	-
DA14	49	7 days	11	35	\$1600	\$1000	\$600	-
DA15	36	5 days	12	53	\$1800	\$1200	\$600	-
DA16	31	6 days	8	28	\$1800	\$1200	\$600	-
Total	933	146	269	1053	\$44375	\$28725	\$14250	\$1400
Average	37,32	5,84	10,76	42,12	\$1775	\$1149	\$570	\$466,6
Minimum	17	4	2	2	\$900	\$600	\$300	\$300
Maximum	62	10	24	122	\$3500	\$2000	\$1000	\$600

Figure 16 illustrates that the higher the number of messages sent, the lower the number of coders who send the message, that is, there is a concentration of messages sent by only a few coders during the challenges. As already mentioned, the copilot is the person who most sends messages in the forums.

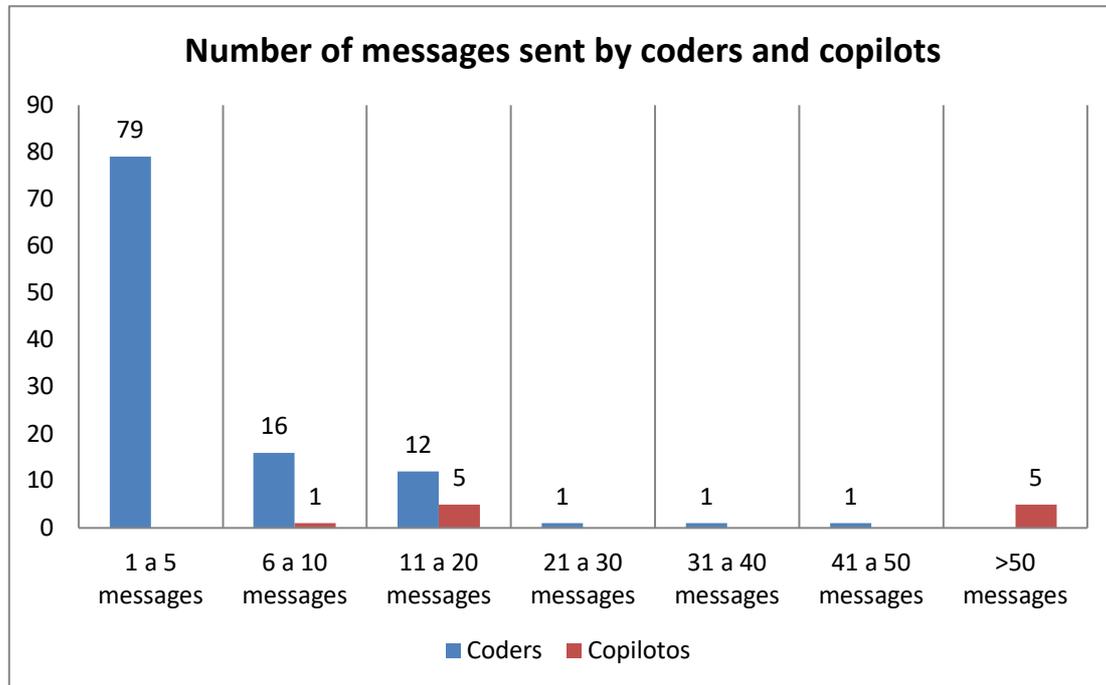


Figure 16 - Number of messages sent in forums between copilots and the crowd

It is possible to observe that the challenges begin with a high average of registered coders. However, at the end of the challenge, few solutions are submitted Table 18. In some cases (challenges DJ2, DJ3 and DA3), only one solution was submitted. This is evidenced by the number of messages sent in the forums of each challenge where few coders exchange messages about the tasks through classified message types. In contrast, it is possible to notice that 9 active coders submitted their solutions to challenge DJ6.

Most coders sent between 1 to 5 messages (**Table 18**). The coders who exchanged most messages with their copilots submitted solutions and obtained a good placement in their challenges.

Table 18 shows the number of messages sent by the coders that sent messages in the forum, and the number of messages sent by the winners (W1, W2, and W3) in order of first, second and third places on the task's classification. The winners sent messages via forum in 20 out of 25 challenges studied.

The column 'submitted solutions' represents the total number of solutions submitted in each challenge. Columns C1, C2, and C3 identify the three coders with the highest number of messages posted in the forum of each analyzed challenge. The ranking order of the challenge winners is identified in W1, W2, and W3.

Table 18 - Winner coders in the forum

Challenge	Coders in the forum	Submitted solutions	N° of send messages	C1	C2	C3	W1	W2	W3
DJ1	6	5	102	21*	16**	16*	16	10	-
DJ2	6	1	21	4	3	3	1	-	-
DJ3	15	1	60	15	6**	4	6	-	-
DJ4	4	2	18	6	3**	2**	2	3	-
DJ5	5	3	43	7	7**	6*	0	7	-
DJ6	9	7	23	11*	7	4	0	0	0
DJ7	4	2	8	1**	1**	1*	1	1	-
DJ8	14	4	76	7**	7	5*	7	2	-
DJ9	5	3	19	4**	2**	2	2	4	0
DA1	7	4	35	10	5**	1*	0	5	-
DA2	8	3	115	41**	9*	4**	41	4	-
DA3	8	1	71	12**	11	3	12	-	-
DA4	4	2	27	10**	2**	1	2	10	-
DA5	3	2	7	2	1	-	0	0	-
DA6	8	3	47	10**	5	3	10	0	-
DA7	6	4	32	8*	4*	3	1	-	-
DA8	4	3	22	5**	4**	4	4	5	-
DA9	5	4	29	10**	4*	-	10	-	-
DA10	7	3	22	8**	1**	1*	8	1	-
DA11	9	3	36	7**	5**	3**	3	5	7
DA12	1	2	2	-	-	-	0	0	-
DA13	45	3	122	21**	5**	4*	21	5	-
DA14	3	2	35	12**	7**	-	7	12	-
DA15	14	2	53	15*	2**	1	2	-	-
DA16	7	3	28	5**	4**	1	5	4	-

In Table 18, cells marked with (*) represent coders with a submitted solution to the challenge. In the cells highlighted with (**), there are coders who won the challenge. The markings with (-) do not present available awards in the challenge or there was no winner in this position. The cells with 0 represent the coders that won the challenge but did not exchange messages in the forum, that is, there was no active communication in the forum.

Regarding the ranking of the 25 challenges, we have the following setting:

- 25 challenges awarded the 1st place (W1)
- 25 challenges awarded the 2nd place (W2)
- 3 challenges awarded the 3rd place (W3)

Thus, 25 challenges offered awards only for the 1st and 2nd places, and only 3 challenges (DJ6, DJ9, and DA11), included the award for the 3rd place, i.e., TopCoder would select and pay for the three best submitted solutions as shown in

Table 17.

The DA12 challenge is considered an outlier of the set of forums analyzed since no messages were exchanged by the coders. Only the copilot used the forum to share task's documentation. The two winning submissions on W1 and W2 were from coders who did not communicate on the forum.

Analyzing the DA12 challenge, one can notice that the duration of the challenge was short, 4 days only between registering and submitting solutions. In addition, yet not the focus of this study, the technology involved in the challenge was about iOS development, which may have reflected on the poor communication and engagement of coders in submitting solutions, since this requires knowledge about a very specific technology feature in market.

4.4 Results about Coder's communication and performance

In this section we restricted our forums' communication analysis just between coder and coder on TopCoder's challenges. Our results reflect collaboration through communication exchanged among all coder who communicate during registration and submission tasks.

4.4.1 Winners who communicate vs Winners who did not communicate

Coders who communicated won 20 out of 25 challenges (80%) of which they registered.

a) Ranking of Winners (W1, W2, W3) who sent messages in the forums

In the 25 challenges that rewarded the 1st place (W1), the coders who communicated (sent messages in the forum) won 80% (20/25) of the challenges. As for the 25 challenges that awarded the 2nd place (W2), the coders who communicated won 60% of them (15/25). At last, in the 3 challenges that rewarded the 3rd place, the coders who communicated won only in one of them, representing a rate of 33.3% (1/3).

For the first place (**W1**) award, the 25 challenges received more than one submission, and, from this total number of submissions, it was possible to select the best solution W1 for the 25 challenges, i.e., for the 25 winning coders. **Table 19** details the number of challenges won by the group of coders who communicated more (C1, C2, and C3), and the challenges won by the coders who communicated with fewer messages in the forum (C4-n). In other words, C1 means the coder who sends the highest number of messages in a challenge; C2 is the second highest number of messages; and so on and so forth. This means that the coders who sent more messages were often those who won the challenges.

Table 19 - Winner's communication

Winners who communicate	W1	W2	W3	Total winners
C1, C2, C3	18	13	1	32
C4-n	2	2	0	4
Challenges won	20	15	1	
Total challenges	25	25	3	
%	80%	60%	33.30%	

The challenges DJ2, DJ3, and DA3 received only 1 (one) submitted solution, where it awarded the winner of the first place (W1) of the challenges (Table 19).

In the DA7 and DA9 challenges, four solutions were submitted, and in the DA15 challenge, two solutions were submitted. However, no submissions were selected as the winning ones in these challenges. This fact implies that, in these 6 (six) challenges, the submitted solutions did not reach the expected quality for the defined criteria and, therefore, were discarded.

b) Ranking of Winners (W1, W2, W3) who did not send messages in the forums

The coders who did not communicate in the challenges won 20% of the total number of challenges (5/25). The distribution of awards for the coders who did not communicate is presented below (Table 20):

Table 20 - Winners who did not communicate

Winner who did not communicate	W1	W2	W3	Total
Coders	5	4	2	11
Total challenges	25	25	3	
%	20%	16%	66.6%	

4.4.2. Winners group who communicated the most (C1, C2, C3) vs Winners who communicate (C4-n)

The coders who communicated the most – group C1, C2 and, C3 won 18 out of 20 W1 challenges (90%) of which they registered. In 86% (13/19) of W2 challenges, and 100% (1/1) of the W3 challenges (Table 21).

Table 21 – Winners who communicated

Group communication winners	W1	W2	W3	Total of group communication winners
C1	9	4	1	14
C2	7	8	0	15
C3	2	1	0	3
Total C1 C2 C3	18	13	1	32
C4-n	2	2	0	4
Challenges	18/20	13/19	1/1	
%	90%	86.6%	100%	

a) Winner Status from the Group of Those Who Communicated (C1, C2, C3)

Regarding the task performance of coders who communicated in groups C1, C2, and C3, the following results defined the coders as "winner", "submitter", and "quitter" status. As described in subchapter 4.2, winners represent the winning coders of the challenges, submitters include workers who submitted a solution but did not win the competition, and quitters included the coders who did not make submissions for the challenge.

- a) **Winners:** 32 coders communicated the most (C1, C2, C3) and submitted winning solutions with a rate of 88.8% (32/36), which represents the total proportion of the C1, C2, and C3 groups for coders who communicated in relation to the total coder that communicated and won in the 25 challenges.
- b) **Submitters:** with a rate of 11.9%, coders of the group that most communicated (C1, C2, and C3 groups) only submitted their solutions to the challenge task. The proportion of submitter coders was (13/109) (according to **Table 18**), which represents the total number of coders that most communicated (group C1, C2, C3) in relation to the total coders that communicated during the challenge.
- c) **Quitters:** the total of 23 coders that communicated the most of the groups C1, C2, and C3 did not submit their solutions. The rate of quitters was 21.1%, considering the relation of the total coders who communicated with the quitter coders of the group that most communicated (23/109).

It is possible to perceive a high rate of coders that communicated and won the challenges. This shows that the communication has a positive influence on coders' task performance.

4.4.3 Discussion

While the level of collaboration and mutual support can drastically decrease when community members become rivals, i.e., when they compete against other [HUT11], we evidenced that collaboration happens among crowd workers.

Coders communicate in an iterative process to collect information and share questions about tasks' requirements, access to files, and other aspects as we illustrated on Table 14. During the task course the co-pilot plays a significant role in social technical coordination through an asynchronous communication channel (the forum) by providing initial guidance and support for coders' concerns, removing misconceptions, getting the registered coders aware of deadline, clarifying requirements and, so on. The co pilot is especially important in an online collaboration, since there is no face-to-face interaction and the communication channel is limited to text [STO14a], [MAC17]. The dialogue among coders is task-oriented and focused on specification's technical and functional questions of the tasks.

We distinguish between coders who communicated from those who did not communicate via forums on TopCoder's challenges. This way, we introduce a method to identify productive coders through the variable communication extending the ranking of the winner, submitter, and quitter proposed in Yang et al. [YAN16]. Thus, for a coding analysis we have some combinations as shown in Figure 17.

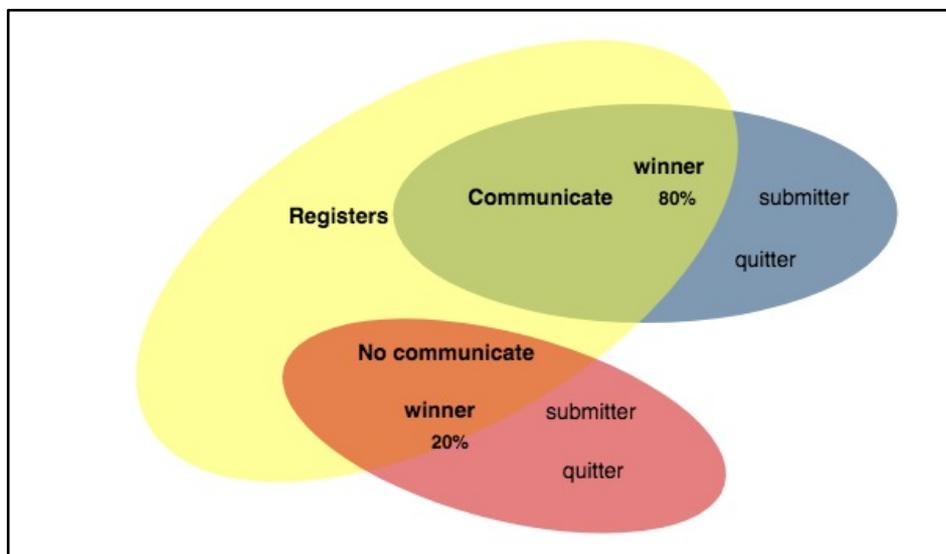


Figure 17 – Coder's classification

We can see that the great majority of coders who communicated won the challenges. It is possible to observe that communication contributed to the coders submitting their solutions to the challenges in which they were registered and thus, competing in the choice of the best solution.

From the analysis of the winners of the 25 analyzed challenges, the results suggest that collaboration characteristics have a correlation with coder's likelihood of completing the

task and submitting solutions that meet the demand of requesters who have utilized the TopCoder platform.

Of the total coders that submitted winning solutions, coders who did not communicate won in only 5 challenges. Thus, if we compare the challenge rate, according to the presented results, the coders who communicated had their solutions selected and awarded in 80% of the challenges compared to the rate of 20% of the challenges won by the coders who did not use the communication forum. In addition, it can be seen that the most productive coders that participated in the analyzed challenges were the coders who communicated more.

To understand whether communication affects productivity of winner coders, the number of forum posts of the three registered users that most sent messages individually in the forum (C1, C2, C3) in each competition was correlated with the ranking position (W1, W2, W3) of each winner coder in each competition.

To understand whether communication helps submission and would reduce the number of quitter, that is, if it would influence the coders to submit their solutions, the number of forum posts of the three registered users that most sent messages in the forum (C1, C2, C3) in each competition was correlated with those who were not in the ranking position (W1, W2, W3), but who had performed the task submission. Such information is obtained in the own tool that maintains the information of who the coders that have already sent their solutions are.

Other aspects can affect the level of collaboration in the communication forum both to ask questions and to receive answers from the crowd itself or the platform's mediator (copilot). One of the reasons for this can be given in relation to the time crowd workers dedicate for the accomplishment of the tasks. Developers can be in full-time employment, and the average duration of competitions (5 days) allows for weekend work to be a possibility. Otherwise, some developers could be working part-time to dedicate themselves to the competitions, and this issue can implicate in collaboration and consequently in submitted solutions, since many doubts occur related to understanding documentation and removing misconceptions on requirements. In many of these cases, crowd workers may register for more than one task in a challenge at a time and drop the ones they cannot complete before the submission deadlines. A task with many unreliable workers is subject to high risk of failure or cancellation [SAR17].

Prize value can also affect the level of collaboration between participants in the challenges. There may be a greater interest in competing for the prize and therefore, increasing the number of messages exchanges on the forums. As it can be seen in the rates in Table 13.

Depending on the complexity of technical attributes of the task, the level of collaboration may be greater for the cases in which the crowd is seeking to understand and clarify points about the task by producing a more active communication via the forum. On the other hand, collaboration among the crowd may be lower because of the unfamiliarity of the platform members with the technology required for the development of the solution.

4.5 Communication patterns from coders

In this section we restricted our results based on interpretation of the content messages exchanged by just coders who communicate on TopCoder's forums challenges.

4.5.1 Introduction

In order to identify how collaboration characteristics among crowd workers impact in the task performance (winner, submitter, quitter), productivity and quality of both SW CS competitors and contests are presented in the next sections.

Figure 18 shows the distribution of message categories in the challenge forums by coders and copilots in the months of July and August.

The 10 categories defined as previously mentioned were: Public Announcement, Tips, Request for help, Confirmation Request, Invitation Request, Identified Problem, Help answers, Confirmation Response, Invitation Response, and Problem Response.

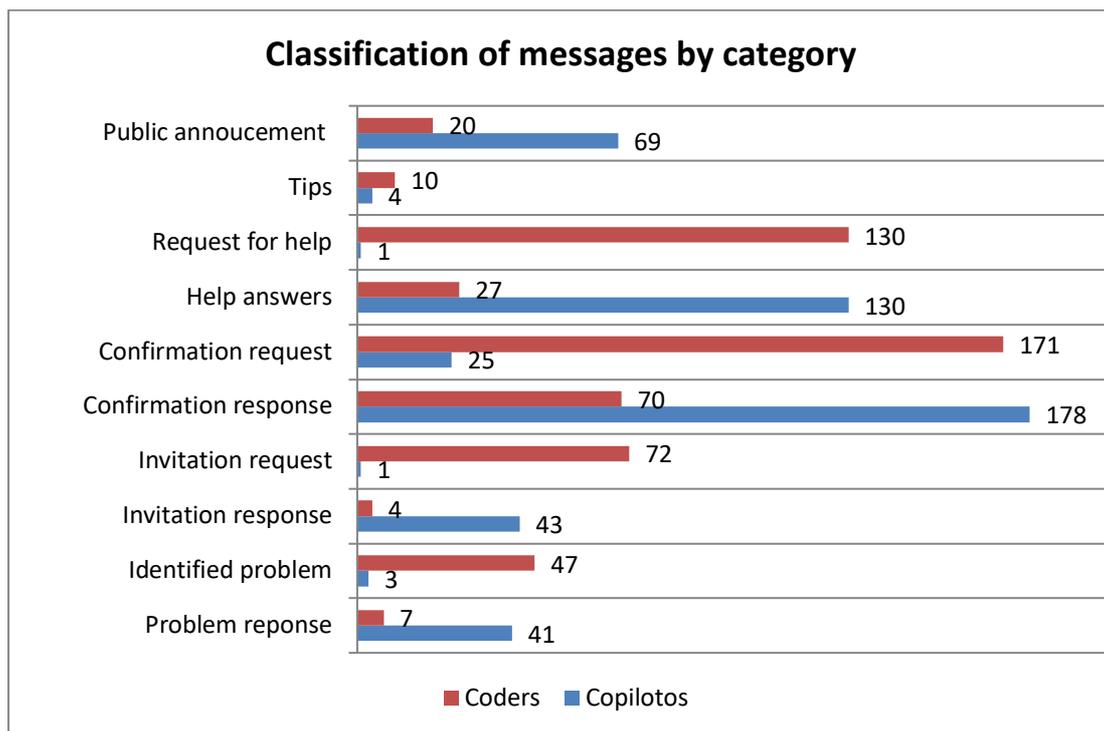


Figure 18 – Message category

The category “Confirmation Request” displays a large number of messages (**Figure 18**). In this category, a question is sent to check and ensure the understanding of the crowd participants on a given subject, usually relating to the task documentation to the requirement aspects, library, processing, among others, as it can be seen in Figure 19. Accordingly, the number of messages sent by the copilot for the category “Confirmation Response” is high, and it includes the response to crowd questions.

The topics that emerged in the second coding of the forums' message analysis are presented in **Figure 19**, summarized between coders and copilots.

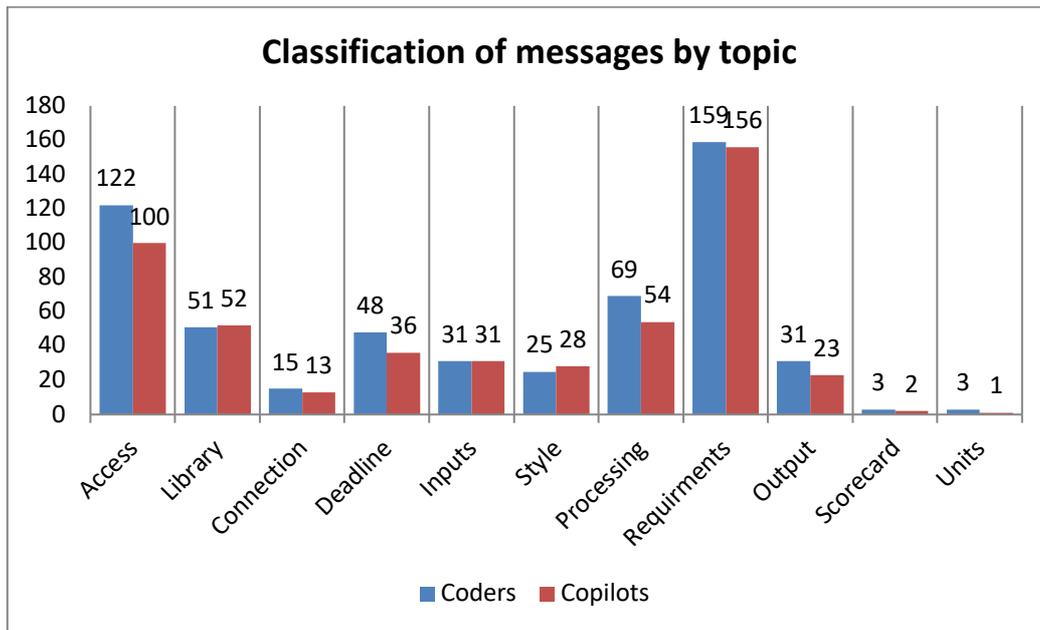


Figure 19 – Message topics in the forums

In this section, we present communication patterns (categories and topics) found for winners, submitters, and quitters who communicated in the challenges. We analyzed the relationship between the type of messages (category and topic) exchanged in the forum among coders and their status: winner, submitter, and quitter. A set of patterns was found for the messages sent by the coders during the analyzed challenges. In addition, it has been observed that coders offer technology-related and development tips, they identify specification problems, broadcast useful information about the task, keep the copilot aware of their interest in submitting a solution for the challenge, among several other aspects.

The number of coders who won tasks with similar communication patterns were illustrated in Figure 20 and Figure 21.

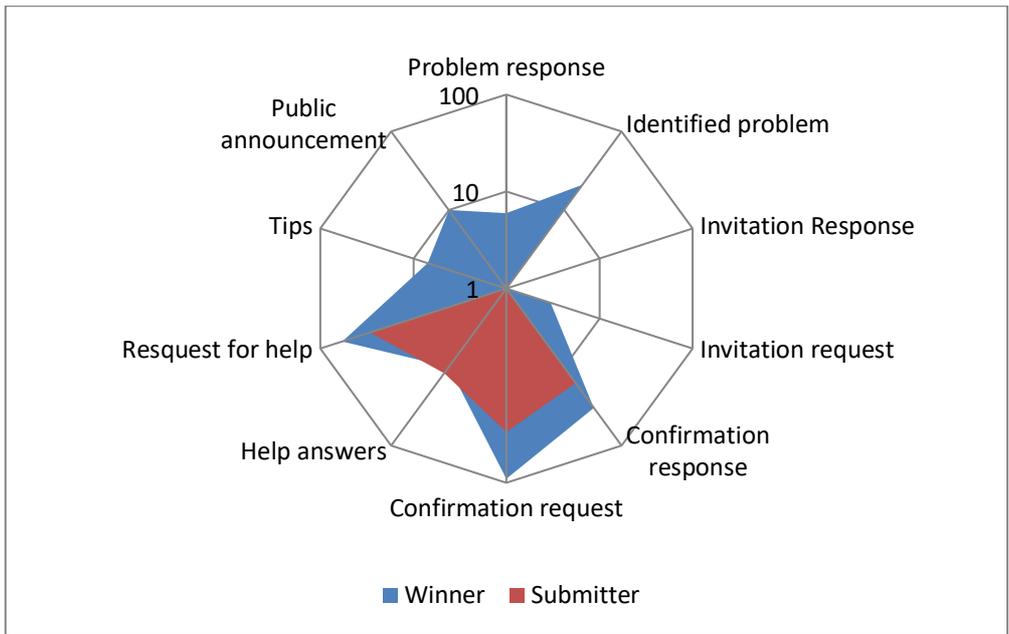


Figure 20 - Forum categories of winners and submitters

Regarding the category defined as Identified Problem related to the topic of Processing was highlighted as the highest record. In the category Confirmation Request, the questions related to the topics Output, Processing, Deadline, and Requirements were highlighted. The topics Processing and Requirements were highlights in the Request for Help category.

Regarding the Response topics, the category defined as Confirmation Response, related to the topics Input and Output, was highlighted as the largest record. In the category Help Answer, the answers related to Inputs, Processing, Outputs, and Requirements were highlighted. In the Tips and Problem Response categories, the incidence was higher in Processing and Requirements, respectively.

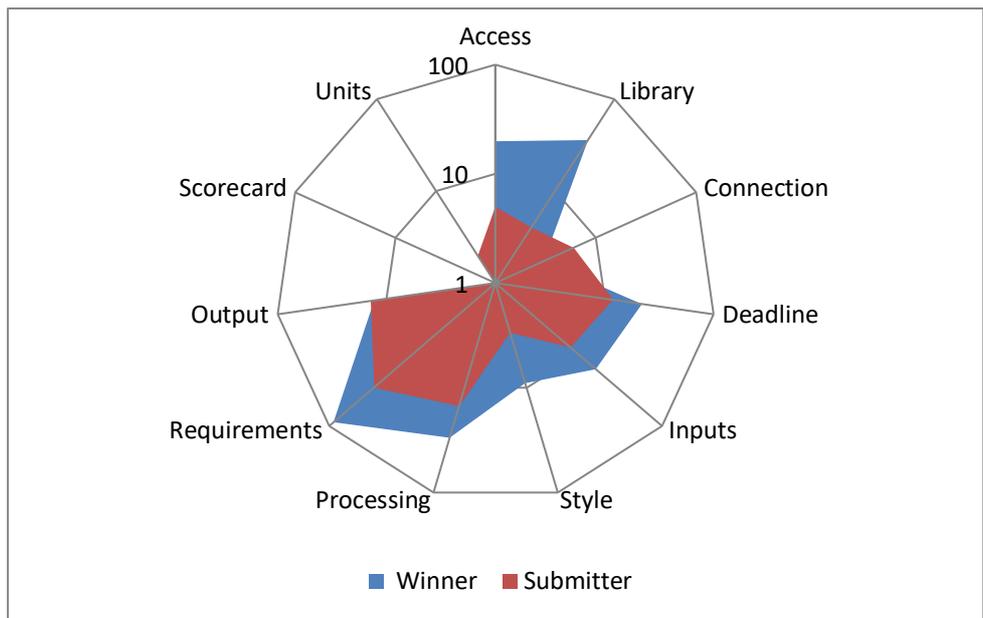


Figure 21 – Forums topics for winners and submitters

4.5.2 Common communication patterns for winner, submitter and quitter

The common communication patterns found for the status of winner, submitter, and quitter on the analyzed forums associated with categories and topics are described below.

For the message topics, coders presented a common pattern on Requirements, Processing, Deadline, Access. Meanwhile, commons pattern for the message categories 9 of 10 were identified: Tips, Identified problem, Request and help answers, Invitation request and response, Public announcement, and Confirmation request and response. Appendix G gives evidences of the commons pattern.

On the relative values of messages sent by coders it is possible to observe that 4.04 winners sent messages about *requirements*, where this average is obtained by total of requirement's messages sent (89/22) and by total of winners who communicated in the challenges. The same average was calculated for submitters and quitters thus, 2.2 submitters and 0.44 quitters exchanged messages about requirements of the task.

Questions and answers on *Help* are send when some information that helps and clarifies subjects about coding compilation, and comparisons of results' output, for example. Here again we see a clear collaboration among coders and the importance of communication to allow a coder to be able, individually, to find a solution to a problem during the development of the task solution

For this pattern some quotes are illustrated in **Table 22**. The quotes were extracted from two threads of different challenges.

Table 22 - Questions and Answers on Help – *Requirements*

Quotes from challenge 1	
Coder 1 quitter	<i>“Regarding the same response for no array situation: since the consequent initial URL will point to segment 1, shouldn't the playbackUrl stay the same regardless of what happened? If we specify manifests/<reclId>/1/manifest.m3u8 as initial, there will be no need to change that URL, or player will request recording by Id instead?”</i>
Coder winner	<i>Should it be treated as a deficiency if, when the recording is segmenting-enabled, (I think VOD and TSB recordings should not have segmenting enabled) the recording has a segments array even though there is only one segment? This does not interfere with anything and can possibly simplify some coding.</i>
Coder 2 quitter	<i>“I'm going with the spec (in the GitLab issue) for the review: "Normally, if a recording is only a single segment, we'll show the information we do now". Minor requirement.”</i>
Quotes from challenge 2	
Coder 1 submitter	<i>“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.)”</i>

Coder quitter	<i>"If you have a look at summary file, it says that for laminates we need to output engineering constants. Engineering constants (Laminate only)"</i>
Coder 1 submitter	<i>"Thank you (...) 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."</i>

Processing topics is evidenced with an average of 1.36 messages per winner, 1.15 messages per submitter, and 0.26 per quitters. The questions on *Processing* are illustrated among coders in the following quotes:

Table 23 – Questions on Identified problems – *Processing*

Quotes	
Coder submitter	<i>"I am so confused. the console says up to date. but nothing is updated. https://gyazo.com/04d99146ddfb4aa44b75d0350621fffa"</i>
Coder winner	<i>It's fine, just a little annoying to me, have to restart it manually.</i>

Table 24 - Questions on Identified problems – *Requirements*

Quotes from challenge 1	
Coder submitter	<i>"I am so confused. the console says up to date. but nothing is updated. https://gyazo.com/04d99146ddfb4aa44b75d0350621fffa"</i>
Coder winner	<i>It's fine, just a little annoying to me, have to restart it manually.</i>
Quotes from challenge 2	
Coder 1 winner	<i>"Why the contents of "Dashboard transitions" and "Profile effects" are identical? Thanks"</i>
Coder 2 winner	<i>"In screen, why we have the "tick" sign in step 1 section? I think step 1 in this screen is still not completed yet? Is it a design mistake? Please refer: http://i.imgur.com/E3T16pC.png"</i>
Coder submitter	<i>"I think we can safely ignore the sample files and use the pdf. I have validated my code against the numerical problems in the pdf (not the specs, the other one that was put up on google drive) and the formulas work pretty well against all the problems. (...)."</i>

For the *deadline* topic, most of the winner's messages (1.0 message) refer to the response that he/she will submit his/her solution to the challenge. During the challenge, the copilot makes a prediction of who is intending to submit his or her solution.

This situation is confirmed because the winner actually submits his solution. After all, (s)he is the winner.

Submitters and quitters sent respectively 0.9 message and 0.15 message for deadline. Such standard implies that submitters and quitters face time problems to complete their solutions and ask the copilot for an extension in the challenge deadline. In some cases, this request is answered by the copilot, and the challenge deadline is extended in a few hours or days, whereas in other situations the deadline is unchanged.

The main concern about deadline of the task can see in the quotes below.

Table 25 - Questions on Confirmation request - *Deadline*

Quotes	
Coder winner	<i>"I'm working hard on this challenge and plan to submit. Thanks."</i>
Coder 1 submitter	<i>"Since there are 3 prizes, no doubt there will be a few submissions, but an extension would give all of us time to improve quality of submission, and make sure it runs smoothly on AWS etc.."</i>
Coder 2 submitter	<i>"Hello, Can you pls extend this challenge for 24 hours."</i>
Coder quitter	<i>"I support this request, can you please extend?"</i>

The request of the *access* topic refers the request of coders to the copilot to release access to certain platforms for accessing code components, needs to start the task development. This topic was also a common communication pattern identified by winners and no winner (submitter and quitters). 0.9 messages are send per winner, 0.3 messages per submitter, and 1.03 message per quitter. Access was the most frequently topic identified for quitters (95 messages in total). This kind of pattern confirms that quitters only demand messages for access the code repositories and, they do not complete the task of challenge.

Table 26 - Questions and Answers on Invitation - *Access*

	Quotes
Coder winner	<i>"(...) says - You are not invited to this organization. Please contact org admin. My email: (...)@gmail.com username: (...)"</i>
Coder submitter	<i>"My handle is (...) Thanks"</i>
Coder quitter	<i>"My github username is (...)Thanks."</i>

The message classified as *Tips* prevails between the winning coders (7 messages in the total) and represents the collaboration between coders through the message of solutions to doubts or problems about the task and, in this way, can aid in their resolution. The tips were related to the topics: *library*, *inputs*, *access*, *requirements* and, mainly *processing*.

Only message of *Tips* about *processing* was sent by submitter.

Noticed also in the quitter's status two messages about *Tips* for *processing* and *inputs* topics were exchange during challenges.

Coders share useful information to perform the activity related for instance, on alternative tools to perform task and, can be verified in the following quotes in **Table 27**:

Table 27 - Tips on *Processing*

Quotes from challenge 1	
Coder submitter → coder winner	<i>"But just a heads up, the numerical problems aren't free from errors, you better have a calculator nearby:)"</i>
Coder winner → coder submitter	<i>"You should google about ABD matrix and read the Spec more carefully. Clearly, we can calculate the ABD matrix with the input data. The spec contains almost everything you need to know."</i>
Coder submitter → code winner	<i>"Yes. It's used to calculate Engineering constants and ABD matrix. You see we need to calculate Qk^* matrix. But it uses Tk matrix which contains many sin and cos. To calculate Tk, we need angles from "Ply Orientations". (...)"</i>
Quote from challenge 2	
Coder quitter → winner	<i>I use the Win10 Ubuntu Subsystem to for node and git... And to an extent apache/mysql if i need those as well. Apt-get works pretty well. https://msdn.microsoft.com/en-us/commandline/wsl/about A little off topic, but just in case people didn't know about it.. :)</i>
Quotes from challenge 3	
Coder winner (<i>Inputs Tips</i>)	<i>"iPad has a split view on designs, I propose we show the root folder on the left, and when opened show the contents on the right. Sounds ok?"</i>
Coder winner (<i>Access Tips</i>)	<i>That's still not good enough, files are returned per-directory, so we need to select one to be able to view files (otherwise it's cascading requests for each folder). So, in addition to that I propose selecting on the left (like messages) will show files for that folder (or no resources if empty). By default show files in root folder</i>

4.5.3 Winners Communication Patterns

Communication patterns are associated with winners from the first 3 positions (W1, W2, and W3), and with the coders who sent more messages in the forum (C1, C2, C3) suggesting that when coders communicate, task submission and winning increases.

A communicative coder can be more productive than an uncommunicative one according the results showed in subsection 4.2. We found that winning coders communicate

better with other participants in the challenge. We observed that coders who are winners are involved in all kinds of messages (categories and topics) as show **Figure 22**.

With a total of 240 messages, winners presented the highest concentration of distributed messages. We highlight the categories Problem response for requirements and processing as a differential for winners where they sent 6 messages associated with this category. Although, the winners present common patterns of communication with other coder, the winners are more active on forum and they contribute directly in several technical and non-technical topics during task's activities. The top five topics from winners are:

- Requirements → 89 messages
- Library → 36 messages
- Processing → 30 messages
- Deadline → 22 messages
- Access → 20 messages

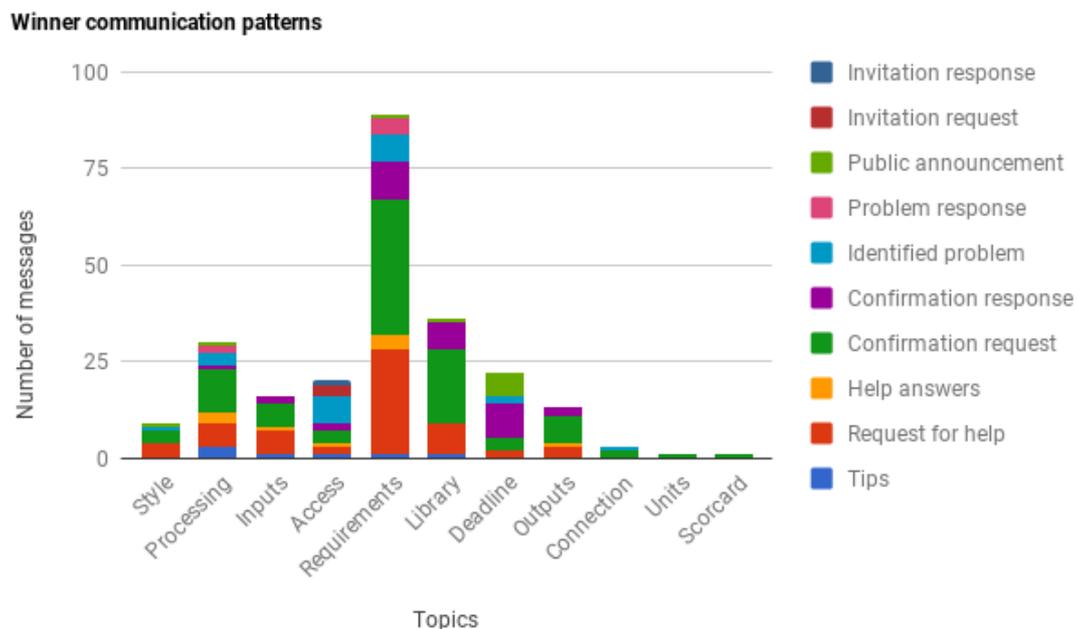


Figure 22 – Winner communication pattern by categories and topics

Not necessarily the volume of messages is important, that is, coders do not need to send more messages in the forum to become winners, but by sending messages about categories such as Problem response and Confirmation request that included most of the time, topics related on requirements, library and processing.

They end up being more assertive in the developed solution, and, therefore, meeting the expected quality criteria for the task. This way, all involved within SW CS projects are benefited, (crowd winners received financial reward for the dedicated effort, requesters got solutions to their problems, and the platform successfully completed the project between the crowd and requester).

The winner communicates through message exchanges in the category *Help* for different topics. This category happens when a coder asks questions for example, to clarify *style* questions (frontend components), technical coding details (*processing*, *inputs*, *library*

and *output*) and, mainly, to obtain help about task's requirements that are needed to deliver the solution as illustrates in **Figure 22**.

The categories *Confirmation Request* and *Confirmation Response* were the categories who winners most exchanged message, including all related topics. The predominant requesting confirmation messages on the various topics confirms that discussions on task documentation (goals, context, technology, tools, etc.) are crucial during development solution and, demands of collaboration and human-driven coordination.

The largest exchange of messages occurred on the requirements topic (45 messages), library (26 messages) and, processing (12 messages) as show **Figure 22**.

There can be many imperfect information in the documentation and winner coders ask to confirm and provide solution for document failures.

The winner communication pattern for the category *Identified Problems* occurred more frequently for the topics *requirements*, *libraries*, and *inputs*. Such pattern indicates that, through the reporting of requirement issues, the coder is reporting that there are inconsistencies in the documentation or some problems with incorrect variables presented in the documentation, and other artifacts made available for the task.

The identification and discussion of such problems via forum demonstrates that the winner coder has already acquired the knowledge/understanding of the task context, technologies, and details involved for its implementation. This aspect is confirmed by messages identified only for winner coders on *Problem response* category (**Table 28**).

The category *Problem response* is prevalent and unique in the winner communication pattern and occurred more frequently for the following topics: *requirements* and *processing*. Such pattern indicates that, through the reporting of requirement issues, the coder is actually reporting that there are inconsistencies in the documentation or some problems with incorrect variables presented in the documentation and other artifacts made available for the task.

The problem alert, on the other hand, points to the weakness of the documentation developed by the requester. The informative report of the coders for identified problems is a differential in the set of messages sent by the winner and can demonstrate a more advanced level in the construction of the solution that he is developing. Through problem messages, the copilots confirm and adjust the new information "on the fly", redirecting technical and functional decisions about the task. In some of these situations, the problem of non-synchronization of the initial document of specification with the new decisions discussed in the task forum is identified, bringing disadvantages to those who are not using the forum to develop their solution. On the other hand, both active and passive forum participants may be aware of the issues shared by competing coders.

Some quotes exemplify the notifying errors and inconsistency in the task documentation provided by requesters, among coders and copilot **Table 28**.

Table 28 – Winner Quotes from *Problems* response category

Problems response on <i>Requirements</i> - Challenge 1	
Quitter	<i>For this Challenge, the goal is extracting all tables that are a "schedule" or *might* contain Panelboard schedule data. In the SampleB-12.pdf file, a successful solution would extract five (5) (...) -- and/or provide a Confidence Score (for each table) indicating the probability it contains Schedule data.</i>
Coder 1 winner	<i>Okay. Thanks for the detailed info.</i>
Coder 2 winner	<i>My two cents opinion: we already have a xlsx file (PanelSchedule-TableExtraction-PDF_Legend) which contains the number of tables per each PDF file. This should give a direction to contestants for development and to reviewers to check how a submission fits with contest requirements.</i>
Coder 1 winner	<i>We can't be sure if that excel sheet is 100% error-free and is based on the latest criteria as communicated by MDuchek. And anyway, because the number of test pdfs will be fairly limited, I think it's better if the reviewers simply open the PDFs and verify things visually.</i>
Problems response on <i>Processing</i> – Challenge 2	
Submitter	<i>Yes quite clearly you can generate the ABD matrix from the inputs given. However, the particular ABD matrix printed in the sample summary file can't be generated from the data in the sample laminate input file(...)</i>
Winner	<i>In my calculation, the sample's data is overall accurate. I got the same result of Engineering constants and Qk matrix and A matrix. But there are some errors in B matrix and D matrix. I have said that B matrix should be 0 for symmetry laminate. I believe these errors come from float number calculation. It really makes sense and I explained that in my submission.</i>

Table 29 - Quotes winner from Identified problem on *Requirements*

Identified problem on <i>Requirements</i>	
Coder 1 winner	<i>"Why the contents of "Dashboard transitions" and "Profile effects" are identical? Thanks"</i>
Co pilot	<i>"Sorry this was a copy paste mistake in the spec. I've fixed it."</i>

Coder 2 winner	<i>"In screen, why we have the "tick" sign in step 1 section? I think step 1 in this screen is still not completed yet? Is it a design mistake? Please refer: http://i.imgur.com/E3T16pC.png"</i>
Co pilot	<i>"You are right on this. It should show the tick once fields for Step 1 have been filled."</i>

4.5.4. Communication Patterns of Submitters

The submitter who communicate via forum in competitive SW CS has different patterns of communication rather than winner. It was possible to notice by the reduced number of categories involved in the messages sent by the submitter. As **Figure 24** illustrates, the response and problem categories and invitation response were not identified among the submitter messages. With regard topics, submitters are involved in topics related to *style, processing, input, access, library, output, connection, and units*. With the largest number of posts are related to *requirement* topic (higher concentration of messages - 29 posts) and *deadline* (12 posts).

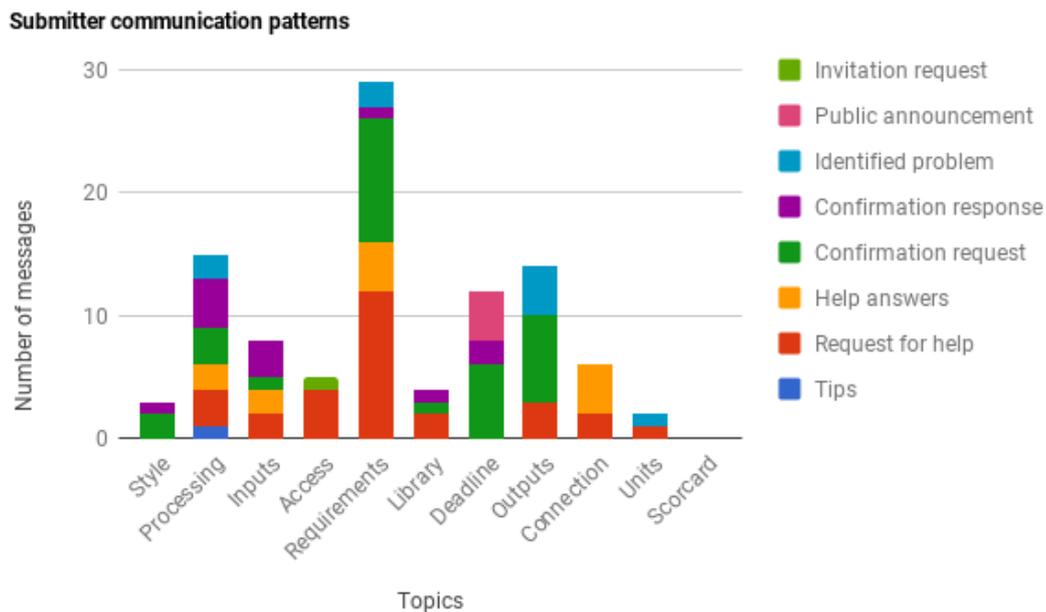


Figure 23 – Submitter communication pattern by categories and topics

With a total of 98 messages submitters collaborated through the categories and topics. The top five topics are:

- Requirements → 29 messages
- Processing → 15 messages
- Output → 14 messages
- Deadline → 12 messages
- Inputs → 8 messages

It was noted that the submitter coder most of the time sent messages related to the topic defined as *Deadline* as illustrated in quotes **Table 25**. According to its characterization, *Deadline* refers to messages that discuss the date and time set to submit the task. Such standard implies that submitters face time problems to complete their solutions and ask the copilot for an extension in the challenge deadline. In some cases, this request is answered by the copilot, and the challenge deadline is extended in a few hours or days, whereas in other situations the deadline is unchanged. In cases where the request for an extended deadline is accepted and such a discussion occurs among those involved, this may benefit the entire SW CS project indicating that communication is also favorable in the presented pattern. Therefore, there will be more and better solutions benefiting the requester and the platform.

For the Help category, submitter communication patterns were similar to the ones found for the other coders (winners and quitters), because it presented a high number of messages sent within this category for all coders. The topics identified in *Request for Help* were: *access, requirements connection, processing and inputs*.

Submitters reported only for one category on *Identified Problems* - questions, indicating a slightly difference from winner, which reports to both categories of problems (questions and answers). It is possible to infer that the coder that only submits is not attentive to the possible problems found in the tasks, and, therefore, does not realize that something is inconsistent or even incorrect regarding the technical and functional specification of the task. The following identified problem by submitters topics were found: *processing, requirements and, outputs*.

4.5.5 Communication Patterns of Quitters

Quitters who collaborate via forum is a communicative coder and s(he) is involved in many categories as well as the winner, however it concentrates their messages to the different topics as *requirements, deadline, processing* and mainly, *access* as seen in Figure 24.

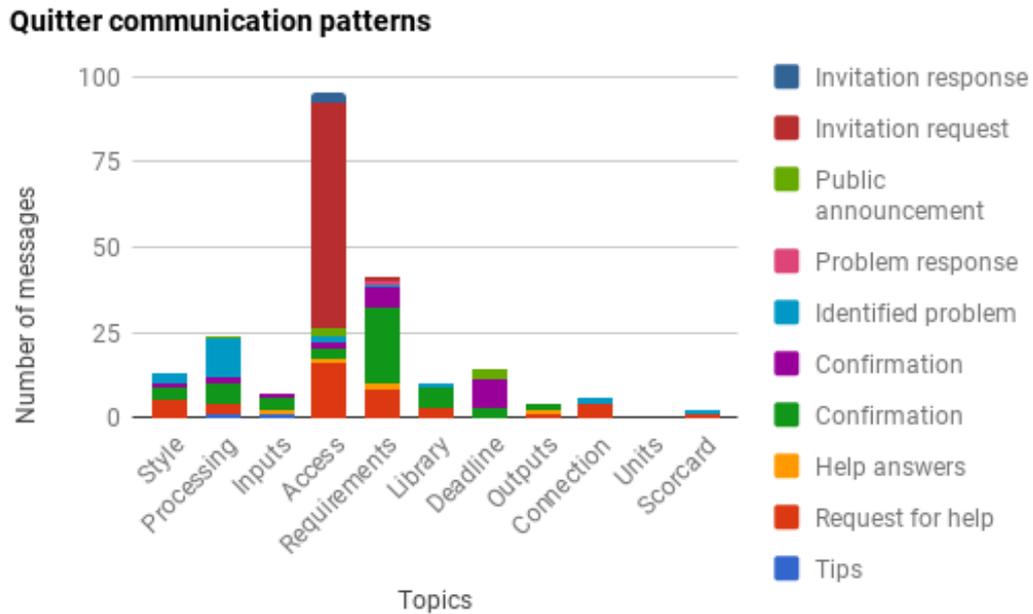


Figure 24 – Quitter communication patterns by categories and topics

With a total of 216 messages the coders quitters collaborated through of top five topics distributed in different categories:

- Access → 95 messages
- Requirements → 41 messages
- Processing → 24 messages
- Deadline → 14 messages
- Style → 13 messages

The most frequently category *Questions on Invitation* identified for quitters was related on *Access* topic (98% of the messages exchanges by them in this category). This kind of pattern confirms that quitters only demand messages for access the code repositories and, they do not complete the task of challenge.

The other frequently pattern evidenced was related to process, *requirements* and *deadline*. This communication pattern supports the huge concern in SW CS projects about problems on documentation [STO14a], [FIT15], [ZAN17], [MAC17].

Table 30 - Quotes from Quitters

Quotes from Confirmation request – <i>Requirements</i>	
Coder 1 quitter	<i>“For the scope of this challenge, it is safe to assume tables will remain on a single page & will not span multiple pages?”</i>
Coder 2 quitter	<i>“This will still work using Filters. It's clear from iPad design that tapping a user on left will filter messages on the right. I think the requirement is about iPhone only.</i>

4.5.6 Discussion

Collaboration Characteristics and Productivity

Who must collaborate with whom to get the task done? Congruence [CAT06] between collaboration and productivity illustrates how winners, particularly the most productive ones, are involved in the exchange of different types of information during task performance, achieving higher congruence.

Since communication in competitive SW CS is limited to written documentation via forums, the documentation definitions are required at every stage thus, one can observe that seeking information through the very narrow chat communication forum is quite frustrating and time-consuming. This means that identifying the most relevant information pertinent to the task at hand is particularly important for the crowd workers in the TopCoder's challenges.

Thus, we had effective ways of identifying detailed the factors that impact on the productivity of the crowd over task timeframe, we would be in a much better position to design software development crowdsourcing contests. That may lead to a significant increase in registrations and submissions solutions for the challenges.

We acknowledge that there is a difference in the communication patterns among winners, submitters and quitters. The first difference refers to the number of messages exchanged for each type of coder. The submitter coders presented a much smaller number of exchanged messages, 98 messages related to winner coders (240 messages), and quitters (216 messages). The second difference is attributed to the number of categories and themes that the coders are involved with. As presented in subsection 5.5.3 the winner coders were involved in communicating all 10 categories and 11 topics defined through analyzing the content of the messages. were involved in a smaller number of messages distributed through 8 of the 10 categories with a recurring communication pattern on the category Request for help and confirmation request of the topics associated with task requirements. In the quitters' group, it was observed the exchange of messages for request for help on access, confirmation request about requirements, and high number of messages for the category invitation request associated with the topic access. The significant communication pattern raised an interesting question: are all coders able to identify problems and, most importantly, to provide some sort of answer to the reported problems accordingly?

Our results suggest that congruence helps increase crowd workers submitted solutions and become a winner of the challenges. In addition, the results showed that winners are involved in several types of collaboration (categories and topics) through communication patterns and, they are more active via forum by better communicating with other competitors in ways that are more congruent with the work they perform. Moreover, the most productive competitors reach higher levels of congruence that the less productive ones. These results suggest that conventional views of competition vs. collaboration need to be change.

The communication patterns proposed in this thesis extends the standard measure of quantity /frequency of exchange messages in forum communication channel on the TopCoder platform by providing a finer-grain level of collaboration characteristics analysis and assessing the role of collaboration in competitive SW CS.

4.5.7 Limitations

The number of analyzed forums, only from the category development challenge and subcategory code. With regard to collecting data from the forums, it is worth mentioning that the challenges are "open calls", and once the participant is a member of the platform and registers in the challenges, it is possible to access the repository of forums of each TopCoder's challenge.

We did not visualize all the coders who did not communicated in the challenge and submit and no were winner and, those did not submit.

4.6 Survey Data

4.6.1 Introduction

In this section we present the survey study following our research design. We conducted an online survey with a group of developers who have recently participated in the TopCoder's challenge to assess their opinion about the influence of collaboration in task performance. Surveys can be used to compare users' attitudes, perceptions, and experiences across user segments, time, geographies, and other aspects (e.g., competing applications). Such data enable researchers to explore whether users' needs and experiences vary across geographies, assess application's strengths and weaknesses among competing technologies, and evaluate potential application improvements while supporting in the decision making between a variety of proposed designs [KIT02a], [KIT02b], [KIT02c], [KIT02d], [KIT03].

Our survey was structured based upon guidelines established by [KIT02a], [KIT02b], [KIT02c], [KIT02d], [KIT03], and included questions about basic demographics, and more importantly, collaboration and communication in competitive SW CS. In our case, we designed and carried out a survey to evaluate our hypothesis comparing:

- How useful would it be (or has it been) to communicate via forum for a crowd worker on the TopCoder challenges? This question is designed to assess whether collaboration impacts on task performance within the classification of crowd participants as winners, submitters and, quitters. Specifically, these questions aim to check whether the coder who communicates can be more productive with a solution and win a challenge.

- How much the collaboration via forum influences the productivity of the crowd workers during a TopCoder's challenge? This question is designed to investigate whether those who did not communicate can also win the challenges.
- What is the communication pattern that potentially assists crowd workers in the effectiveness to win a competition? This question is designed to investigate whether the coder winner sends certain messages that help in choosing their solution.

4.6.2 Data Collection

We designed and conducted an online survey, consisting of three steps. First, we invited the group of participants. We sent an invitation letter by email (Appendix E). Secondly, each participant answered the questions from the survey about communication and collaboration aspects on TopCoder. Lastly, we analyzed the answers based on the experience in performing SW CS tasks from the group of developers who participated in TopCoder challenges.

The survey was pretested with five colleague researchers who are familiar with competitive SW CS. After several rounds of adjustment, the survey was ready to be fielded to the entire sample.

We designed the survey with 20 questions divided into three sections. The first section asked questions about the TopCoder experience of the respondents in registering, submitting, and winning development challenges. The second section was carried out using Likert scale and open-ended questions about collaboration characteristics, about who the participants are more likely to collaborate with, and how this collaboration correlated with participants' performance in the contests, regarding the influence of communication in submitting and winning a challenge. Still in the second part of the survey, two questions were included related to self-determination with the use of other communication channels among the participants who compete in the platform. Finally, the third section collected demographic information from respondents.

The survey was distributed by email to a population of 51 software developers who had competed on TopCoder. We received 11 responses, i.e., the response rate was 21.5%. The data collection was conducted between January 23rd and February 15th 2018: a period of 4 weeks. A Google Forms was designed and used to collect data. For details about these forms, please see Appendix F.

We used inferential correlations [MÜL14] to assess whether the collaboration characteristics were most strongly associated with crowd winner who communicate in the SW CS challenges. In addition to analyzing the closed-ended responses, the review of open-ended comments contributed to a more holistic understanding of the survey data and revealed important insights that cannot otherwise be extracted from closed-ended responses [MÜL14].

4.6.3 Demographics information

In the total of entries from gender, of those who indicated, 60% of the respondents are male and 40% female. They all live in Brazil and are highly educated: 100% of them are graduate students (Master or PhD) and rather young (75% between ages of 23-39). They all are experienced developers with an average of 63% of them having between 5 and 10 years of software development experience.

Being the sample non-probabilistic and intentional, where the selection of respondents was for convenience from the selection of developers who had competed on TopCoder platform, the answer to the question “**How many development challenges have you participated (registration phase) on TopCoder?**”, unveiled that most respondents, 90% of the total, had participated in more than one TopCoder challenge, see **Figure 25**.

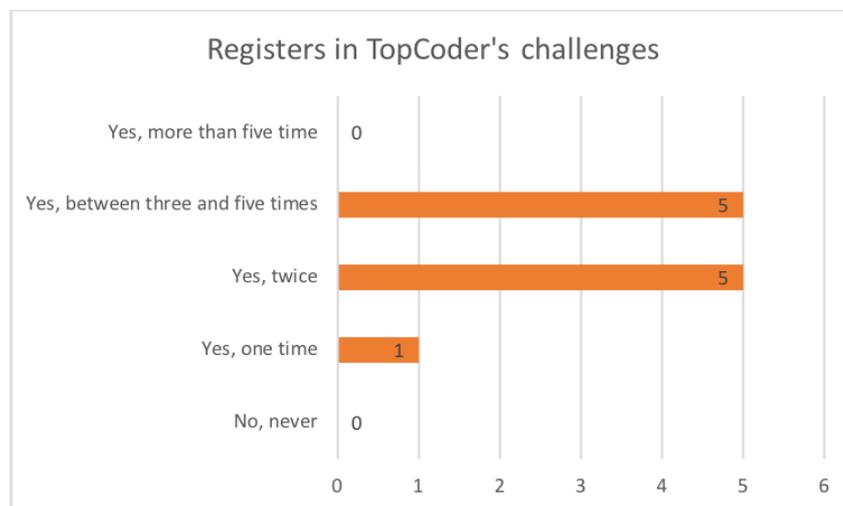


Figure 25 – Number of times participants registered

The question “**Have you ever submitted any solutions to challenges on TopCoder?**”, about task performance in terms of solution submission during the challenge, has had the following set of answers: Of the total respondents registered in the challenges, the data show that 36% of them did not submit any solution (**Figure 26**). For the participants who submitted their solutions, in turn, 64% of the total of solutions submitted to the challenges were performed at least once, twice, and between three and five times.

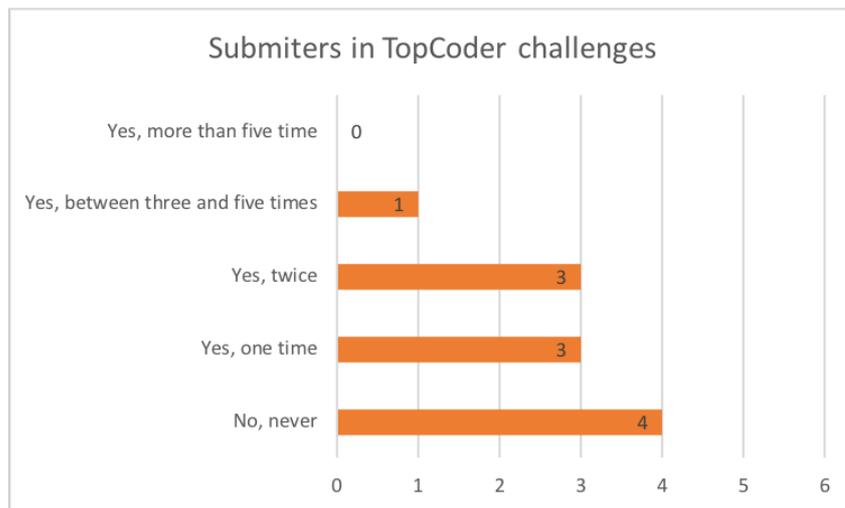


Figure 26 - Number of solutions submitted

For the total number of registered participants in the platform development challenge, it is observed that the rate of submissions exceeded the rate of non-submissions (quitters) for the challenges reported in the survey. **Figure 27** shows the rate of 64% of solutions sent to the task (including submitters and winners), and the rate of 36% of respondents who did not submit their solution to the challenge. For the analyzed population, only 1 (one) respondent won one of the challenges of which he competed.

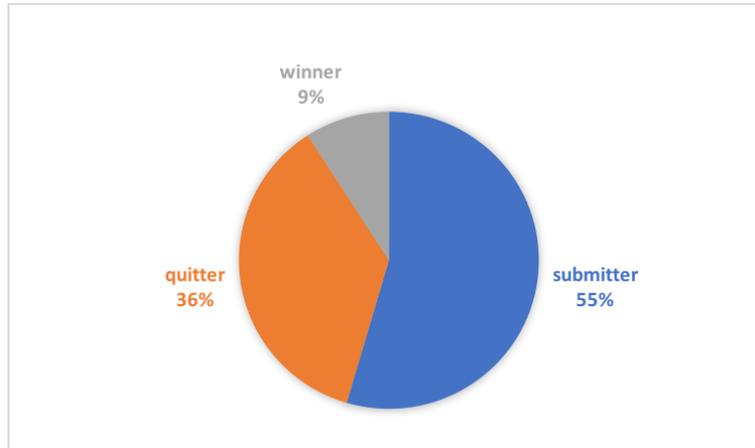


Figure 27 – Total of submitters and quitters

In general, these results mean that these respondents somewhat experienced with the TopCoder platform having participated in more than one challenge, having mostly submitted solutions and even won a competition. This all means that they can provide more insightful answers than first timers on the platform.

4.6.4 Results

This subsection is organized as follows. First, we present the results for the questions related to use of communication channel during task performance.

Correlation Communication vs Task Performance

Regarding the use of communication, a smaller number of respondents collaborated through the forum 45% (5/11) meanwhile 54% (6/11) of the competitors did *not* use the forum during the challenges they participated on TopCoder. **Figure 28** also shows the number of times that respondents who communicated or not, submitted their solutions. The results on task performance from participants who communicate are: winner 20% (1/5), submitter 60% (3/5) and, 20% (1/5) quitter.

Developers who communicated via forums were able to submit to at least one challenge in which they participated, which suggests, according to the characteristics of collaboration evidenced in this thesis, that the interaction between the participants on the forum contributes to the task performance as a whole (submitter and winner). Note also that the rate of quitters was 50% for developers who did not communicate compared with 20% of the developers who did communicate (Table 31).

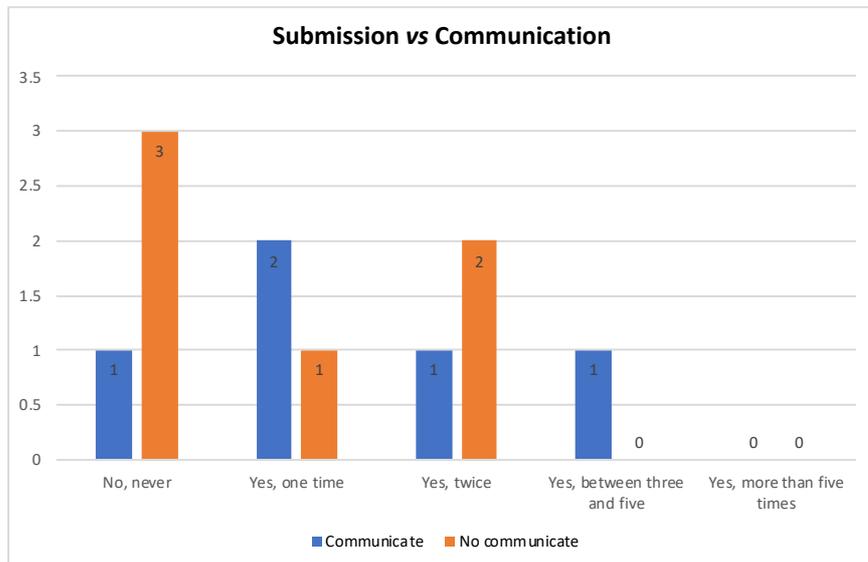


Figure 28 - Summative between submission and communication

Table 31 - Total number of participants in the communication forum

Do you use the TopCoder forums to communicate with the co pilot or other crowd members during a challenge?	Total	%
Yes. Participants who communicated	5	45%
No. Participants who did not communicate	6	55%
Total	11	100%

Note that the respondents who answered “No, never” for the question “Have you ever submitted any solutions to challenges on TopCoder?” are the quitters, i.e., coders who failed to submit their solutions. The number of quitters (who did not communicate) surpassed the number of coders who are quitters but communicated using the forums. In turn, respondents who communicated surpassed the number of times they were able to submit their solutions compared to those who did not communicate.

The use of forums enabled respondents with the status "submitter" to deliver their solutions at least once (two entries) and between three and five (one entry) times. In the “Yes, twice” entry, we had a slightly variation, participants who did not collaborate on forum submitted once more.

This data again suggests that there is correlation between communicating in the forum and not being a quitter, i.e., submitting a solution. This confirms the results presented on section 4.4.

Answers of questions for participants who communicate vs who did not communicate

Figure 29 reports the results of the questions related to Likert scale to prioritize the respondents' choices given as collaboration barriers and characteristics in competitive SW CS evidenced in this thesis. The questions focused on assessing the collaboration characteristics and task performance congruence.

We observed that 45% of the respondents *agreed* and *strongly agreed* with question 2, which is about engaging in communication with other crowd workers as something beneficial during TopCoder challenges. 36% of the informants reported *neutral* for this question.

We observed the same pattern (45%) regarding respondents agreeing and strongly agreeing about communication via TopCoder forums (sent or read posts) as something that helps crowd members to submit a task solution during competitions (question 4).

Question 6 asks about the potential of communication via forums to win a challenge. In this case, we perceive a concentration, with 45% responses of the neutral type. This can be explained by the high number of respondents who did not win a challenge on the platform (90%) and, therefore, could not evaluate whether the forum could or could not contribute to their chances of becoming a winner. The majority of the respondents (64%) reported that they did not use the challenge forum during task's activity. According our previous results about the characteristics of collaboration, an assessment of the survey confirms that coders who not communicate are less productive than coders who communicate during TopCoder challenges.

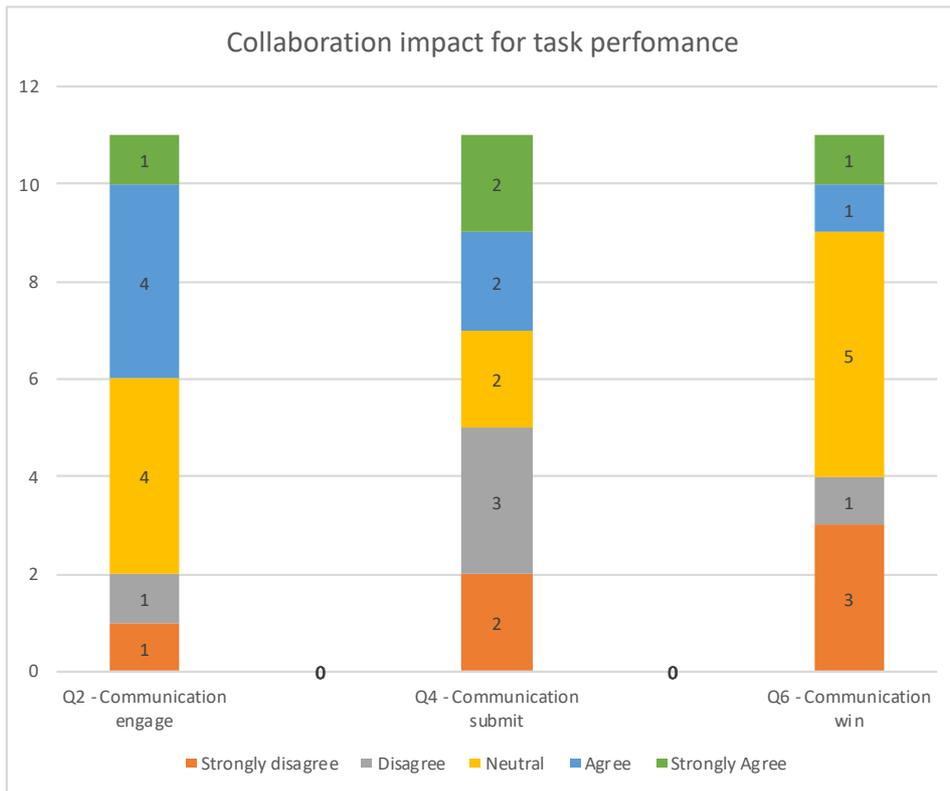


Figure 29 - Results of questions about communication vs task performance

Disagreements from the participants selected “strongly disagree and disagree” in their responses were found for questions 2, 4, and 6 (Figure 29). Analyzing the results, we noticed that one specific respondent (Participant 9) disagreed with all three questions; in his open-ended questions we found possible explanations. He mentioned that the platform did not answer on time, and when they did, other questions did not help, they were not satisfactory. He also mentioned that there were not enough explanations in the forums of the challenges he participated. We can see that quotes related to questions for Participant 9 (Table 32).

Table 32 - Quotes of survey's participant who did not communicate

	Answers for questions Q2, Q4, Q6
Participant 9	<i>“They did not respond in a timely manner and when they did, they did not clarify all the doubts” - Participant 9</i>
	<i>“There were not enough explanations in the forums” - Participant 9</i>
	<i>“If you cannot ask questions, imagine helping to win the competition.” - Participant 9</i>

A) Question 2 - Engaging in communication with other crowd members is beneficial during a TopCoder challenge?

We asked the participants to answer: Do you agree or disagree with the following statement: *“For me, engaging in communication with other crowd members is beneficial during a TopCoder challenge,* and to complete their answers with question: *“Regarding question 2, WHY do you agree/disagree? Can you think about an example or situation?”*

Table 33 presents the responses from the survey. As mentioned, one respondent disagreed and his answer was already discussed.

Answers for participants who communicate

We have a total of 45% (5/11) developers communicated during the challenges that they registered. The results on task performance from them are: winner 20% (1/5), submitter 60% (3/5) and, 20% (1/5) quitter.

The developers who communicated was able to submit to at least one challenge in which they participated, which suggests, according to the characteristics of collaboration evidenced in this thesis, that the interaction between the participants on the forum contributes to the task performance as a whole (submitter and winner). While the rate of quitters was 50% for developers who did not communicate compared with 20% to the collaborative developers.

Table 33 –Question 2 and quotes by participants who communicate

	Communication engaging in TopCoder's platform
Winner	<i>"In order to deliver the TopCoder challenge/task itself the communication with other crowd members should not be necessary beneficial. But for the personal development and knowledge exchange then yes."</i> – Participant 11
Submitter	<i>"clarifying technical questions or picking up tips / ideas for certain situations"</i> – Participant 2 <i>"So that everyone can reach the same level or as close to it as possible in relation to the task."</i> - Participant 5
Quitter	<i>"It can be useful in sharing experiences"</i> - Participant 7

The winner neither agrees nor disagrees about the fact that communication is beneficial in the development of the task.

However, one perceives the positive inclination given at the end of his answer to the question, where the forum centralizes information that can be useful for the construction of the individual knowledge of the competitors. It can be inferred that the exchange of knowledge through the communication in the forums not necessarily benefits participants collectively, that is, in some cases, for those who are only participating in the forum in a passive way (viewing and reading posts), the exchanged messages are not useful, in which case the participant may decide not to interact and keep his questions about a task.

As for the results for the submitters (Participant 2 and Participant 5) we highlight the impact from forums in anticipating issues and doubts about software development in the context of SW CS tasks. In the same way, Participant 7 pointed out the forum utility to sharing task information despite being a quitter.

Answers for participants who did not communicate

We have a total of 54% (6/11) developers who did not communicate during the challenges. The results on task performance of these group are: submitter 50% (3/6) and, quitter 50% (3/6).

The developers who did not communicate were able to submit once at least and at most twice in challenges which they participated. However, these participants did not win any challenge and the rate of quitters surpassed in 30% as compared to developers who did not communicate. The participants were inquired on the benefits of communication during the challenges (Table 34), in which the submitters answered:

Table 34 - Question 2 and quotes by participants who did not communicate

Communication engaging in TopCoder's platform	
Submitter	<i>"I had no reason to go the forum"</i> – Participant 1
	<i>'About business requirements I see people helping each other, but no technically.</i> - Participant 6
Quitter	<i>"I didn't have this opportunity during my experience."</i> – Participant 4
	<i>"I have not engaged in such activities."</i> – Participant 8
	<i>"I agree. However, I did not feel confident enough to talk to any of the other members. Instead, I just followed other people doubts to found out my solutions."</i> – Participant 10

An important point illustrated by Participant 6 is that s(he) accessed the forum and read some posts indicating a passive participation in the forum and confirming the collaboration among crowd members during task activity.

The quote from a quitter (Participant 10) with the same passive participation on the forum illustrates the agreement about the forum's benefits; however, the participant did not feel confident to collaborate during the challenge. This result is supported by literature in Gray et al. [GRA16], where the authors discuss workers with less English fluency or familiarity with the discussions via forum who are unable to fully take advantage of the opportunities to collaborate.

- b) Question 4 - Communication via TopCoder forums (sent or read posts) helps me to SUBMIT a task's solution?

To verify how the use of communication influenced participants' submitted their solution from TopCoder's challenges, we asked: Do you agree or disagree with the following statement: *"Communication via TopCoder forums (sent or read posts) helps me to SUBMIT a task's solution"*.

Answers for participants who communicate

Table 35 the submitters participants (Participant 2, 5 and 7) were not the winners with their solutions but they clearly considered activities associated with the forum important for task performance. This show that most part of the participants noticed the positive influence on forums during task execution. For Participant 7, the forum helped in mutual questions, but added an issue about time effort to understand the discussion associated with task. This issue can be attributed to collaboration barriers on difficulty to communicate in asynchronous channels and low global project view. Besides that, the short timeframe to deliver solutions during the SW CS contests.

Table 35 - Question 4 and quotes by participants who communicate

	Communication helps to submit
Winner	<i>"The forum posts may clarify some of my questions regarding the task."</i> - Participant 11
Submitter	<i>"In my case, it clarified a doubt about the deliverables of a challenge."</i> – Participant 2 <i>"It helps you to ask questions and have a better understanding of the task."</i> – Participant 5
Quitter	<i>"Help with questions that may not have been raised until viewed on the discussion forum. But sometimes they can bring even more doubts (demanding more time in the understanding)."</i> – Participant 7

Answers for participants who did not communicate

The participants who did not collaborate (in active way on the forum), both submitters and quitters, mentioned that they were benefited from the messages posted on the forum by other participants of the challenge. The quotes on **Table 36** from Participant 6 and Participant 10 illustrate that.

Table 36 – Question 4 and quotes by participants who did not communicate

	Communication helps to submit
Submitters	<i>"It is simple to submit a task"</i> – Participant 1
	<i>"I don't use forums"</i> – Participant 3
	<i>"Other members ask questions about the task and then when I read, sometimes it answers questions that might have had."</i> – Participant 6
Quitters	<i>"I didn't have this need during my experience."</i> – Participant 4
	<i>"I have not engaged in such activities."</i> – Participant 8
	<i>"By reading some of other members messages I was able to solve part of my problems (however, I never sent any solution)."</i> – Participant 10

- c) Question 6 – Communication via TopCoder forums (sent or read posts) helps me WIN a competition.

The last question about collaboration and productivity congruence we asked: Do you agree or disagree with the following statement: *"Communication via TopCoder forums (sent or read posts) helps me WIN a competition"*. The additional question 7 says: *Regarding question 6, WHY do you agree/disagree? Can you think about an example or situation?*

Answers for participants who communicate

The winner (Participant 11) mentions the forum had a crucial role in clarifying the details of the task and supported him/her to win. This result suggests that collaboration via forums contributes to become more productive, and, this way, results in a task solution that is acceptable and that meets what the requester needs (expectations) in competitive SW CS.

Participant 2 reported his/her feeling about the competitive nature of challenges and the contradiction in collaborating in that context.

Analyzing the quote presented by Participant 7 individually, his answer is related to the fact that he had not had the experience of winning a challenge. On the other hand, the participant mentioned in the previous questions (2 and 4) that the forum is the channel that enables visibility for crowd competitors, which is an important collaboration feature.

Table 37 - Question 6 and quotes by participants who communicate

Communication helps to win	
Winner	<i>"The forum posts may clarify some of my questions regarding the task and so support me to win the competition. But it is not a must, that is the reason I select 4 and not 5 points for the question 6. On the challenge that I won the company provide details on the Forum (not started by any member question). So, in this case, the forum supported the task development". - Participant 11</i>
Submitter	<i>"I do not believe it helps, and in my case, it did not help even because it was a competition." - Participant 2</i>
Quitter	<i>"I cannot answer if it would help to win a challenge or not." - Participant 7</i>

Answers for participants who did not communicate

Some participants did not answer the question 6, since the questions were not mandatory. While other participants repeated their answers from previous questions.

We believe that participants did not feel able to answer the questions about correlation between communication and win a challenge because they did not communicate via forum nor won a challenge.

Table 38 - Question 6 and quotes by participants who did not communicate

Communication helps to win	
Submitter	<i>"Never won"</i> – Participant 1
Quitter	<i>"I didn't win any competition"</i> – Participant 4

Participant 8 and participant 10 just repeated in this question their answers gave for questions 2 and 4.

Questions related on other communication channels

Still in relation to communication, respondents were asked: *"How often do you use each of the following communication channels to interact with crowd members in order to gather any information to apply in your task solution on TopCoder?"*

It was noted that respondents used other channels to communicate with TopCoder members. **Figure 30** exhibits the distribution of use of the following communication channels: 55% of respondents confirmed the use of TopCoder's forum, 36% indicated they used one of the code hosting services (GitHub, BitBucket²⁵, Google Code²⁶, or SourceForge²⁷) for communication among TopCoder members, and 54% answered they use one of the Q&A sites such as Stack Overflow and Quora²⁸. Most respondents used the TopCoder forum to communicate during the challenge, since the specific information about the task is located there (documentation or technical information discussed among the participants); however, it is important to highlight the use of collaborative code hosting tools and Q&A sites.

It can be inferred that the use of external channels of collaboration such as Q&A among TopCoder competitors may be used as a way to seek information to clarify doubts about the technologies requested in the tasks or to gain insights in other communities of software developers and that can be applied in the development of the solution to be submitted on TopCoder. There is a growing trend in the use of Q&A sites to discuss, share experiences, doubts, and knowledge among the software developer community [STO14a].

²⁵ <https://bitbucket.org/>

²⁶ <https://code.google.com/>

²⁷ <https://sourceforge.net/>

²⁸ <https://www.quora.com/>

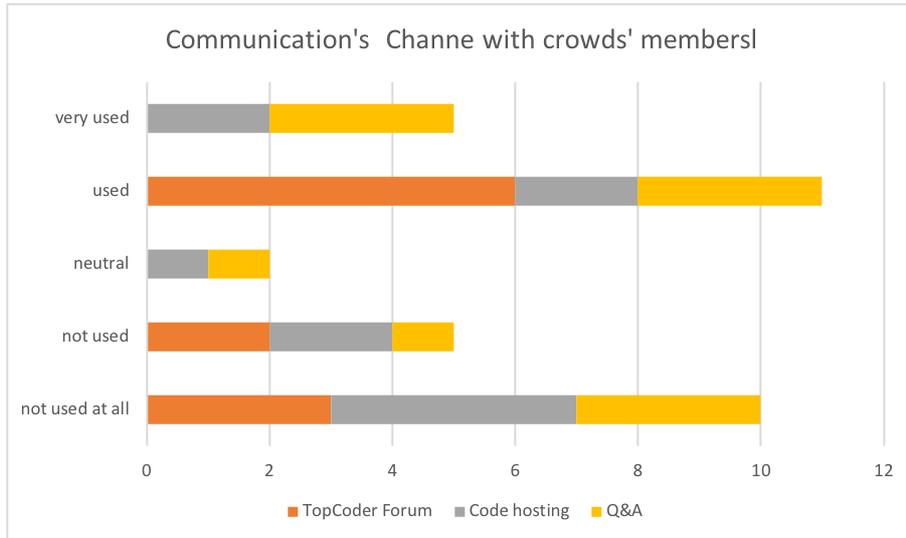


Figure 30 - Communication channel used between crowd

Regarding the question *“How often do you use each of the following communication channels to interact with OTHER DEVELOPERS (who are NOT participating in the challenge) in order to gather any information to apply in your task solution on TopCoder?”*, it was possible to observe (Figure 31), that respondents also make use of Q&A sites, including the microblog service (e.g., Twitter²⁹), to communicate externally with other developers and seek knowledge from other communities to assist in resolving the challenge tasks, as state by [BEG13], [STO14a].

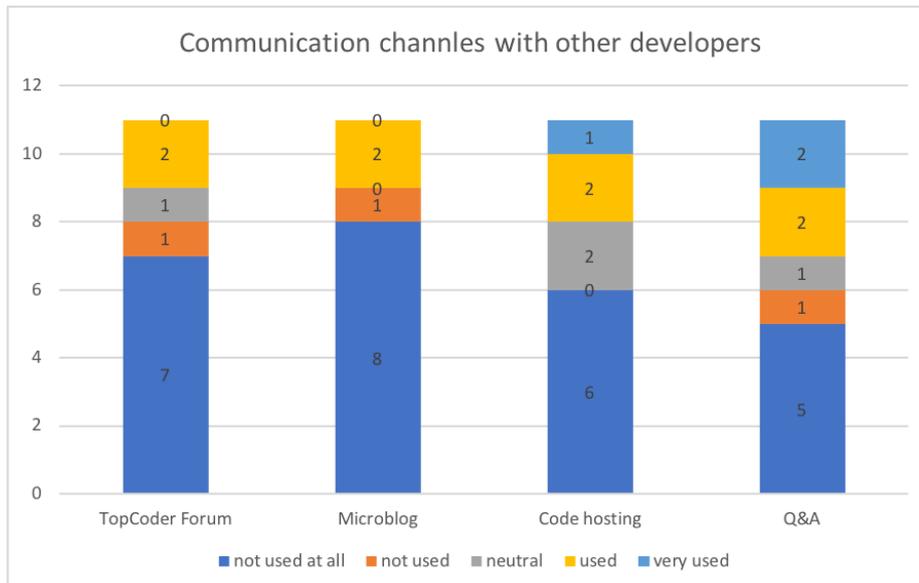


Figure 31 - Communication channel used between other developers

²⁹ <https://twitter.com/>

4.6.5 Discussion

Correlating Collaboration, Task Performance, and Productivity

The questions and the quotes presented to ground our findings about the collaboration impact, task performance and, productivity is positively correlated, and, in this case, the collaboration among participants during the challenge is an important driver for them to improve task performance and be more productive in relation to the other competitors.

It is possible to notice that the new roles which emerge in SW CS development such as copilot (platform), and the requester who demands the task, are not clear to the crowd. Considering that the interaction with crowd participants happened via asynchronous communication channels and is mediated by copilots (experienced members of the crowd community and platforms specialists) who interact with the requester [FIT15], this fact can justify the participants' response.

Regarding this situation, the copilot, by sending information via forum, is communicating and publicly distributing information that will be read by everyone who accesses the forum. Using the forum, it is possible to discuss files with problems, clarify doubts about the features that need to be delivered, etc. The discussions, however, tend to be discreet as to the possible solutions of the task, without compromising the competitor with relevant information about the strategies and decisions of their solution.

Among those who communicated and submitted a solution, one of them was the winner of a challenge. Even with a reduced sample, it is possible to notice that the communication has a positive effect for coders to complete the challenge and to be chosen as the winners with the best solution for the registered challenge.

On the other hand, in 100% of cases, those who did not communicate also did not win the challenge. Nevertheless, it is observed that, in some cases, the participants were able to submit their solutions without communicating in the forum, which does not guarantee they did not access it, i.e., they may have accessed the forum but passively. A passive participation refers to the registered coders that only follow the messages posted in the forum (Topcoder registers the views, as well as the feedback given, of the coders who report how useful or not the messages posted were or the answer to their question message). Such an action would imply saying that even by explicitly selecting 'No' for the question that says: "Do you use TopCoder forums to communicate with the copilot or other crowd members during a challenge?", they have been benefited from the messages posted in the forum and thus have managed to solve their solutions and submit them on time. This indicates that even through an implicit / indirect collaboration, it is still possible to be productive and send a solution.

Participants who did not communicate could not submit their solutions. In fact, they did not access the forum at any point of the course of the challenge and did not become aware of the messages and information that were being exchanged there.

Through the active or passive collaboration (reading or posting messages) in the forum, where such messages may have been posted (messages such as question/ request for help, confirmation or answers, tips, etc.) by the crowd or the copilot, the participants agree that the forum helps in the process of task submission.

Thus, the survey aimed to assess the potential benefits of collaboration among crowd members in terms of task performance, providing support for submitters complete the challenge and helping to reduce the amount of quitters. Moreover, the results from survey evaluated the congruence between collaboration and productivity in competitive SW CS.

4.6.6 Limitations

While our survey response and assess collaboration impact in task performance in competitive SW CS we had limited opportunities to recruit other geographical set of developers who had compete in TopCoder platform. It is important to mention that TopCoder platform keeps profile information of their pool the resources / members in a sort of "black box", where the community is basically identified by their nicknames. It is not obligatory that TopCoder's users mention any personal information in their profile (they can only associate the technologies of which they have skill), and rare users associate their Github profile or other websites. The anonymity of TopCoder users virtually makes the contact with both access via platform and external access to other networks impossible. Alternative forms of social engineering were used to try to connect with the community (through networks such as LinkedIn and Facebook), but without success. Also, Topcoder users' discussion forums have not been identified as it is common in OSS communities. Just a few isolated members mentioned about how to get started and win a Topcoder challenge, and sought help solving algorithms with Q&A sites such as Quora. In addition, email messages were exchanged with Dave Mesinger, Chief Architect and VP of Product of Topcoder to arrange a collaboration with the author of this thesis, and other students and researcher professors in SWCS of MuNDDoS research group; however, this collaboration did not happen on time before the conclusion of this study.

We also designed and partially performed an experiment in the online format of the competition like TopCoder, in collaboration with Prof. André van der Hoek from UCI (as mentioned in section 1.7), but, due to the low number of participants, the experiment was canceled, again affecting our empirical data collection in software development competitions.

Throughout a number of attempts, sending invitations to TopCoder registered coders within recent tasks via post forum on the development challenges, and also through sharing our survey in the general discussion TopCoder's forum (coders community) (Appendix F), all without answers, we noticed the distinctive culture of the "community", and the great difficulty of accessing and collecting data directly with a very specific sample of crowd workers.

5. DISCUSSION

The SW CS strategy taps global talents to work on software development challenges, but it also increases complexity to decide which development tasks are more suitable to be crowdsourced as well as setting and orchestrating unstable and undefined virtual workers. Besides that, SW CS platforms have a relevant importance providing directions for the task management (allocation and submission) [STOa14], [YAN16] [ZAN17], quality assurance [LAT15], [SAR17], motivation and remuneration [STO14a], [MAO13], [TAJ13], communication, and the coordination [PEN14], [STO14a], [MAC17] of processes and people in both technical and business levels.

Communication and collaboration among involved parties is a critical endeavor in software development and our results suggest it is also a critical feature of SW CS projects. It requires platforms to take the lead aiming to guarantee that members of the crowd (indirectly) work together in an effective manner. The majority of current SW CS platforms cannot meet the required collaboration mechanisms comprehensively [PEN14], [MAC16c], [MAC17]. In fact, recent tools that explore collaboration among crowd workers (like CrowdCode [LAT14]) have provided inspiring results regarding the quality of the submitted solutions.

Communication and collaboration barriers in SW CS hinders SW projects for all involved: crowd, requesters and platform owners. However, the organic nature of collaboration among crowd workers was surprisingly identified in CS applications through the analysis of different platforms by Gray et al. [GRA16] and Machado et al. [MAC17] such as LeadGenius³⁰ and TopCoder [TOP17]. The authors indicate that there are interactions and social relationships among crowd members even in a competition, and that the crowd uses collaborative strategies to articulate work on these platforms. In Gray et al. [GRA16] study, they have identified different forms of collaboration (sharing administrative overhead to reduce costs, seeking out job opportunity information to share with the crowd, and helping other crowd active participants to advance or finish a given task), in the moment of the task completion on CS platforms.

In our previous empirical studies and as a result of this thesis we argue that crowd workers present collaboration characteristics that are strongly correlated with delivering winning solutions in SW CS challenges. This is an important result that resembles traditional and distributed software development [CAT06], [KWA11]. It is important to note that requesters and SW CS platform owners assume that crowd participants do not communicate with each other. While each challenge is inherently competitive and much of the collaboration on TopCoder is a structured one, i.e., the TopCoder's process dictates how that collaboration takes place [NAG13], our results show that not only do crowd workers communicate but they are also willing to help each other. For instance, crowd workers do share tips among themselves.

Communication, as already mentioned, is a concern in SW CS [STO14a], [FIT15], [MAC16a], [MAC17] and, it is accentuated when the main channel to exchange information

³⁰ <https://www.leadgenius.com/>

is asynchronous. The authors in [BOU14a], discuss topics hindered in asynchronous communication environments where typically many topics are active with team members making contributions at the same time, possibly on different topics. In addition, long time lapses between communication events can lead to discontinuous and seemingly disjointed discussions. In [OSL00], they emphasized the role of synchronous interactions in providing rapid feedback among team members, and in supporting design and collaborative problem-solving.

The results are consistent with the barriers of collaboration identified in Chapter 3, related to unclear and misconception task's documentation and, non-synchronization of information that is exchanged during the forum, it means, immediately when these crowd questions are asked for the copilot, and he/she, in turn, answers or makes decisions directly in the forum that end up not being updated in the document and are only registered in the forum.

The collaboration barriers in the SW CS scenario creates a tension between the needs and the capabilities of distributed software development environments, and leads to misunderstanding, miscommunication, and coordination problems. While SW CS projects and platforms have been increased, we observed there are opportunities for improvement in collaboration among subsets of crowd workers and new workers.

With respect to unstructured collaboration, discussion forums enable participants to ask questions, and discuss the requirements with other participants. This discussion often adds additional details or reduces ambiguity in the contest specification. In this thesis, we showed that the crowd collaborates via forums through these discussions and, thanks to them, collaboration is correlated with task performance and productivity of the crowd.

Similarly, other researchers on SW CS [LAT15], [PEN14], [NAG13], [TAU17] suggest that the quality improvement of the solutions is associated with the crowd dynamics of collaboration. In this sense, any improvement of collaboration in the competitive SW CS context will need to be considered by platforms of on-demand services.

One important question in crowdsourcing studies is "How can the "long tail" of the crowd be mobilized to participate and submit their solutions in SW CS approaches?" That is, while Topcoder boasts more than 1.2 million members [TOP17], only a fraction of its members seems to be actively participating and submitting solutions to the challenges. Another question is "How effective is the competition-based approach to crowdsourcing compared to alternative and more collaborative approaches to crowdsourcing software development?" A clear understanding of these aspects can prevent the requesters who expect to receive quality solutions for their business from advertising competitions that are not attractive and that, consequently, might fail due to lack of submissions. Thus, when requesters are seeking to increase speed of software development through crowdsourcing, they need to be aware and choose collaboration strategies for the crowd and supported by the platform. Similarly, crowdsourcing software development platforms need to provide support to connect crowd workers during task execution and beyond.

The high task-quitting rate and low-quality solutions are huge concerns to competitive SW CS platforms [SAR7]. Thus, the high number of registrants in the challenge is not a guarantee of a high number of submissions. On the contrary, as other research [YAN16] also shows that the low number of submissions reveals an alarming factor in the SW CS projects. A number of empirical studies have attempted to relate the reasons that lead the platforms to receive a small number of solutions from crowd participants who had registered (specifically analyzing TopCoder data) such as: task allocation [MAO15a], [YAN16], unclear documentation and onboarding tasks [ZAN17], and pricing the task [MAO13]. Recently, studies involve the non-collaboration on a competition platform as a factor associated with social and technical barriers to complete SW CS tasks [MAC16b], [MAC17].

As the current baseline, the results extracted from TopCoder forums in this thesis highlight the important role of collaboration in both solution submission and winning the competition on the platform, i.e., task performance.

Consistent with previous empirical studies in this thesis, factors about communication during SW CS tasks are related to significant collaboration. Collaboration among coders facilitates one to share and gain insights into the best ways to drive the implementation of his/her solutions. Besides that, during collaboration among competitors, communication patterns evidenced some barriers mapped in the conceptual model of collaboration presented in Chapter 3 of this thesis through categories *social interaction, task design, and process management*. This confirms that the coders that collaborate in the forum seek to alleviate doubts about lack or incomplete documentation details and artifacts of the tasks, technical and infrastructure setting issues (*requirements, library, processing, etc.*), few integrations with collaboration tools (*access*), information visibility issues, latency of time between the question generated in the forum and the confirmation that the copilot must have with the requester and so on.

Table 39 illustrate the quotes about latency information between co pilot – requester and co pilot – crowd as one of the collaboration barriers identified in this thesis.

Table 39 - Example of quotes about latency information barrier

Confirmation response - Requirements	
Co pilot	<i>"These were all based on client feedback. I am checking with them on it. Stay tuned."</i>
Co pilot	<i>"I've escalated this to the client and asked them to fix it asap."</i>

There is a trend that suggests through the results in the evaluatory phase, forum analysis and data survey, that similar patterns of communication influence in the submission rate and the selection of the best solutions provided by crowd workers. In other words, the characteristics of collaboration among the crowd and the platform bring benefits for the performance of individuals in a more productive way, being the winner better rewarded in the contest.

Effectively does those who communicate submit a solution? Forum participants had high winning rates, whereas low participation in the forum had high quitting rate. The observed correlation implies that participants who were participatory via forum and were involved in different types of collaboration, given a particular task's dependency, tended to do well, i.e., better collaboration, better results.

Effectively do those who communicate win the challenges? In 80% of the cases, those who participated in the forum delivered solutions and ended up being chosen as the winners of the challenge. Thus, our results found that higher congruence is associated with better task performance it means, collaboration characteristics is congruent with crowd workers productivity. Similar approach was mentioned by Cataldo et al. [CAT06] for the coordination requirements domain. The authors found congruence between coordination requirements and coordination activities where that congruence helps to reduce the amount of time required to preform tasks.

SW CS in competitive model through the characteristics of collaboration between the participants, presented in this thesis, proved to be crucial to solve and answer several issues related to communication and coordination during the development of the tasks, as evidenced in Chapter 3 and Chapter 4. It was observed the importance of discussing information about the tasks collectively. Such discussion made it possible for the participants to acquire a sort of "appropriation" of knowledge on the task they were providing.

The results suggest that, even though the challenges are realized in a competition format, where the award is financial, there are different types of collaboration among coders. Thus, a study on crowd collaboration on TopCoder may suggest mechanisms that encourage coders to collaborate on the platform. We believe that this work will contribute to increase knowledge on crowdsourcing in the context of collaborative software development, since collaboration characteristics among the coders were presented, which it could be seen:

- a. Impact on the crowd in the understanding of the task and decision to carry on the challenge (quitter, submitter),
- b. Impact on the requester, receiving quality, diverse solutions, and fast delivery of the crowdsourcing project,
- c. Impact on the platform facing the productivity of the resource pool offered to organizations, and for the success of the software development strategy offered by the platforms.

Thus, our research can be used to provide design recommendations both to the platform software requirements - supporting tools to connect workers, building collaboration into their workflows, and designing the CS challenges, returning to the challenges of mediating collaboration. Some examples to provide greater collaboration among the participants can be cited: team formation, allowing crowd members to share solutions, and the reduction of the ranking visibility of coders who have already submitted solutions to the challenge.

Regarding the collaboration challenges recommendations for new software requirements for CS platforms and for different design of development contests in SW CS, we could mention analytical recommendations based on our TopCoder forums' analysis study, carried out in this thesis.

The recommendations display the difficulty to effectively guide online discussions about the task's requirements among crowd developers, platform and requesters and use a range of online tools to support discussion about task's cycle (assigning, understanding requirements, technical setting, coding, testing, delivering). We drafted a set of recommendations useful for requester (companies or individual requester) that want to obtain new (quality) or complementary (quantity) software solutions for their project's demands, for crowd developers who want to contribute and be productive in SW CS projects and, finally, for SW platforms that want to design effective online SW CS challenges.

Table 40 shows the recommendations to overcome collaboration barriers associated with each collaboration category. We observed an overlap between some recommendations with literature review study. For instance, lack of communication among crowd workers, scarce social media support, task decomposition issue.

Table 40 - SW CS collaboration recommendations

Collaboration categories	SW CS collaboration barriers	SW CS collaboration recommendations	
Social interaction	Lack of Informal communication	→ Conduct effective online feedback and reflection for crowd workers to alleviate fleeting relationship	
	No communication between crowd and requester	→ Support communication with other crowd developers to reduce misunderstandings and weak social context	
	Lack of real-time collaboration	→ Provide real time collaboration to avoid outdate information between forum's decisions and spec documentation	
	Restrict asynchronous communication channel		→ Provide a synchronous communication channel to minimize no direct communication among crowd and platform to perform the task
			→ Co pilots create a more regulary useful communication (on forums) with crowd workers to monitor and estimulate some discussions and clarify questions about task requierement details
	Few interactions among involved participants inside the platform		→ Incentivizing a "warm up" to discuss task's context among requester (who desired to participate) to anticipate crowd issues and doubts about documentation task's specification

Competition Model	Difficulty to dealing with competitors	→ To allow share solutions among crowd developers to encourage collaboration among crowd developers → Reduction of the ranking visibility of crowd workers who have already submitted solutions to the challenge to keep willing to task's submission
	Competition discouraging collaboration	→ Team formation to alleviate the competition and rivalry relation
Task Design	No synchronization of decisions taken in forum vs. documentation (and vice versa)	→ Greater visibility to tasks decisions' making update the documentation of task requirements changed in the forum
	Unclear and inconsistent documentation	→ Better divide and describe tasks due to high granularity and complexity to understanding → To provide task's documentation such as: downloadable files, screenshots / screens, images, diagrams, etc. as a complementary information
Process Management	Difficulty to management in large scale distributed settings	→ Manage producer-consumer relationship to reduce latency of information via asynchronous communication channel
	Information visibility issues	→ Provide mechanism to notify errors and inconsistencies posted in the task documentation to improve visibility and reduce gaps of tacit assumptions of the tasks
	Technical and infrastructure setting issues	→ Indicate and integrate tasks with other development tools and collaborative repository (Github, Stackoverflow, etc.) to facilitate the understanding of source and structure code and, reduce time-consuming

It is worth remembering that all suggestions presented refer to activities performed just in the "Code" of Development Challenge category in Topcoder, and were not validated.

The second research phase, detailed in Chapter 4, occurred specifically to evaluate the collaboration and communication difficulties during the registration and submission phases of the software tasks that the crowd faces during the challenges of TopCoder platform.

6. CONCLUSIONS AND FUTURE WORK

This research sought to contribute to the knowledge about SW CS in the development of collaborative software through a study on TopCoder. In addition, it aimed to stimulate studies on collaboration in this form of work and to allow greater understanding about the crowd workers in the SW CS platform.

Data were collected from different sources and, specifically the source on messages exchanged between crowd workers by content post from communication forums, as far as we know, there are no previous studies have reported as a strategy of research and analysis in the SW CS area.

Messages were coded into 10 categories and 11 topics. After analysis, we concluded that although this is a competition, there are different types of collaboration between coders. Continuous analysis of these data over a period of time may reveal the evolution of the increase in coders' participation in the forums each month, which is ongoing. It is noteworthy that the results sought to stimulate a greater contribution in empirical research in SW CS instead of focusing only on the construction of platforms and applications. In other words, understanding crowd collaboration, or at least part of it, can serve as a starting point for investigating factors that aim to increase collaboration on Topcoder [L113], and similar platforms.

The barriers and collaboration characteristics identified in the competitive environment of SW CS, emphasize that crowd workers collaborate to fulfill technical and social needs left by the platform, independent of the strategy for designing and creating software in which they act.

In the SW CS environment, particularly in competitive activities, there are potentially misunderstanding mismatches, scarce or overwhelming communication, and lack of social interactions to develop software activities that might impact on the success of the project. The lack of appropriate communication can result in difficulties to understand the task requirements at the time of deliver a solution, reducing the productivity and affecting the solutions quality.

We then explored the consequences of collaboration in terms of task completion during SW CS contests and the productivity of the crowd. Our analysis found that communication helps to keep crowd workers to perform tasks and, can reduce the amount of withdraw registers. In addition, the results show that crowd workers who have communication patterns are more productive with the work they perform, submitted solutions and consequently, winning a SW CS challenge. Moreover, the most productive workers reach higher levels of collaboration than the less productive ones.

These results suggest that preventing collaboration from the point of view of communication in competitive SW CS is negative to get the largest number of solutions and increase speed of software development through crowdsourcing.

Furthermore, we bring a study on how developers collaborate on TopCoder, who the participants that collaborate the most are during task completion, and which collaboration

characteristics related to these participants are. Regarding the communication patterns, we found that crowd workers who collaborate in ways that are more congruent with the work they perform. Moreover, the most productive crowd worker reaches higher level of congruence than the less productive ones.

The collaboration barriers mentioned in this thesis can, at first, disturb SW CS platforms that display a strongly competitive nature, limiting and inhibiting collaboration among their participants.

If a while ago, software development was looking for solutions to deal with the collaboration challenges imposed by physical distance, what we observe now is that distance is no longer a factor that prevents collaboration. What has become clear, though, is how competitive platforms in the context of SW CS are re-creating collaboration barriers that have already been overcome with coordination and communication mechanisms over the past years.

The idea that reducing dependency between tasks and software components would also reduce the need for collaboration among participants in a software project was not found in the context of SW CS. The results obtained in this thesis suggest that even in a competitive and “isolated” environment, software engineering project work is a highly collaborative activity, and promises to continue in this way.

In the Chapter 3 and 4, we reported the results for answering the three research questions. The answers are briefly described in the following.

RQ1. Which collaboration barriers do the crowd face when performing tasks in a competitive SW CS environment? By analyzing different qualitative and empirical sources such as interviews, literature review and case study we obtained a set of collaboration barriers faced by crowd workers in competitive SW CS. In addition, we provide the collaboration model in SW CS.

RQ2. Which current collaboration characteristics are present in competitive SW CS? Based on the quantitative (statistics of forums’ messages among crowd – crowd and crowd-platform) and qualitative analysis (interpretation of the content of the forums’ messages among crowd and crowd) we evidenced different communication patterns from the crowds’ role such as winners, submitters, and quitters from SW CS challenges.

RQ3. How might the collaboration impact in crowd productivity? We analyzed the crowd developers who communicated during task performing and shows that the most productive developers are those with higher congruence, i.e., they communicate with the developers with whom they have dependencies due to the code they are changing.

This thesis encourages academics, practitioners, and community members to refine our ways of thinking about issues on collaboration in SW CS projects.

Limitations

As any other empirical work, we have limitations and topics that remain for future work. As one limitation, our study concentrated in one SW CS platform. Although it is the most used platform, we cannot say that the behavior identified here would replicate in other platforms. Moreover, our approach does not consider other Topcoder competitions such as Specification, Architecture, Design, Development, Assembly, and Test Suites. These challenges could present a different communication behavior and, therefore, different results for the productivity aspects of crowd workers, as well as for the patterns of communication encountered. New coding of categories and topics for communication patterns could also be identified in these challenges.

The number of analyzed forums and the manual form of analysis can also be considered a limitation. However, qualitative analyzes face an initial thread off for the recognition of desirable and emerging domains. The bias question of the categories of communication found was alleviated by double checking with another researcher.

Barriers and collaboration characteristics impact on the quality of solutions and the productivity of the participants involved in competitive SW CS. We presented examples of these barriers and characteristics and illustrated who the participants are more likely to collaborate with. These results have expanded our understanding on how collaboration takes place in competitive SW CS, but there is still much work left to be done. A possible extension for this study is to conduct a series of studies to:

- **Mining platform's forum repositories to confirm the barriers and characteristics.** Some of the identified barriers can be further analyzed using software-mining techniques. A possible future direction could be to use these techniques to verify which barriers a given project presents.
- **Investigating on task's descriptions and technologies** required from the tasks to have a relationship in terms of collaboration levels/characteristics and barriers.
- **Mining sentimental analysis.** An interesting future research direction could be to use mining techniques and natural language processing to explore sentimental analysis from exchanges of messages in the communication challenges like a forum in SW CS projects.
- **Adding new attributes.** In the analysis of the barriers and characteristics of collaboration, the experience of the platform participants could be analyzed, evaluating their relationship with the productivity between experienced and beginner participants.
- **Another look at the barriers.** The barriers and characteristics of collaboration can be evaluated specifically from the perspective of requesters and the platform, enabling a different perspective on the findings.

- **Investigating different challenge categories.** Consider other challenge categories (data science) and subcategories of TopCoder, including other SW CS platforms, to verify the relation between the existing and new characteristics and barriers to collaboration.
- **Advance to other SW CS competitive workflow phases.** To extend the study of collaboration to other TopCoder platform workflow phases such as: review and appeal.

REFERENCES

- [ADE12] Adepetu, A., Ahmed, K., Abd, Y.A. "CrowdREquire: A Requirements Engineering Crowdsourcing Platform". In: Proceeding of the Association for the Advancement of Artificial Intelligence Spring Symposium: Wisdom of the Crowd, 2012, pp. 2-7.
- [ÅGE15] Ågerfalk, P. J.; Fitzgerald, B.; Stol, K. J. "Software Sourcing in the Age of Open: Leveraging the Unknown Workforce". Springer Briefs in Computer Science, 2015, 71p.
- [ARC10] Archak, N. "Money, glory and cheap talk: analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on TopCoder.com." In: Proceedings of the 19th International Conference on World Wide Web, 2010, pp. 21-30.
- [BAR13] Barzilay, O., Treude, C.; Zagalsky, A. "Facilitating crowd sourced software engineering via stack overflow". In: Finding Source Code on the Web for Remix and Reuse, Springer Internacional Publishing, 2013, chap. 14, pp. 289-308.
- [BEG13] Begel, A.; Bosch, J.; Storey, M.A. "Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder". *IEEE Software*, vol. 30-1, Jan 2013, pp.52-66.
- [BOU14a] Boudreau, K.; Gaule, P.; Lakhani, K.R.; Riedl, C.; Woolley, A.W. "From crowds to collaborators: Initiating effort & catalyzing interactions among online creative workers", Harvard Business School Working Paper, 2014, pp.14-060.
- [BOU14b] Boughzala, I.; De Vreede, T.; Nguyen, C.; De Vreede, G.J. "Towards a maturity model for the assessment of ideation in crowdsourcing projects". In: Proceedings of the 47th Hawaii International Conference on System Sciences, 2014, pp. 483-490.
- [BRA08] Brabham, D. C. "Crowdsourcing as a model for problem solving: An introduction and cases". *Convergence*, vol. 14-1, Feb 2008, pp. 75-90.
- [CAR01] Carmel, E.; Agarwal, R. "Tactical approaches for alleviating distance in global software development". *IEEE Software*, vol.18-2, Mar 2001, pp.22-29.
- [CAR99] Carmel, E. "Global software teams: collaborating across borders and time zones." Prentice Hall PTR,1999, 269p.
- [CAT06] Cataldo, M.; Wagstrom, P.A.; Herbsleb, J.D.; Carley, K.M. "Identification of coordination requirements: implications for the Design of collaboration and awareness tools". In: Proceedings of the 20th Anniversary Conference on Computer Supported Cooperative Work, 2006, pp. 353-362.
- [CRO17] Crowston, K. and Shamshurin, I. "Core-periphery communication and the success of free/libre open source software projects". *Journal of Internet Services and Applications*, vol. 8-1, Jul 2017, pp.1-10.

- [DOA11] Doan, A.; Ramakrishnan, R.; Halevy, A. Y. "Crowdsourcing systems on the world-wide web". *Communications of the ACM*, vol. 54-4, Apr 2011, pp. 86-96.
- [DUB16] Dubey, A.; Abhinav, K.; Taneja, S.; Viridi, G.; Dwarakanath, A.; Kass, A.; Kuriakose, M.S. "Dynamics of software development crowdsourcing". In: *Proceedings of the 11th International Conference on Global Software Engineering*, 2016, pp. 49-58.
- [DWA15] Dwarakanath, A.; Chintala, U., Viridi, G., Kass, A., Chandran, A., Sengupta, S.; Paul, S. "CrowdBuild: a methodology for enterprise software development using crowdsourcing". In: *Proceedings of the 2nd International Workshop on CrowdSourcing in Software Engineering*, 2015, pp. 8-14.
- [EST12] Estellés-Arolas, E.; González-Ladrón-De-Guevara, F. "Towards an integrated crowdsourcing definition". *Journal of Information Science*, vol. 38-2, Mar 2012, pp. 189–200.
- [FAR11] Faraj, S.; Jarvenpaa, S.L.; Majchrzak, A. "Knowledge collaboration in online communities". *Organization Science*, vol. 22-5, Feb 2011, pp.1224-1239.
- [FIT15] Fitzgerald, B.; Stol, K.J. "The dos and dont's of crowdsourcing software development". In: *Proceedings of the 41st International Conference on Current Trends in Theory and Practice of Informatics*, 2015, pp. 58-64.
- [GNY11] Gnyawali, D.R.; Park, B.J.R. "Co-opetition between giants: Collaboration with competitors for technological innovation". *Research Policy*, vol.40-5, Jun 2011, pp.650-663.
- [GOL11a] Goldman, M., Little, G.; Miller, R.C. "Real-time collaborative coding in a web IDE." In: *Proceedings of the 24th Symposium on User Interface Software and Technology*, 2011, pp. 155-164.
- [GOL11b] Goldman, M. "Role-based interfaces for collaborative software development." In: *Proceedings of the 24th Symposium on User Interface Software and Technology*, 2011, pp. 23-26.
- [GON11] Gonçalves, M.K.; de Souza, C.R.; González, V.M. "Collaboration, Information Seeking and Communication: An Observational Study of Software Developers' Work Practices". *Journal of Universal Computer Science*, vol.17-14, Jan 2011, pp.1913-1930
- [GRA16] Gray, M.L.; Suri, S.; Ali, S.S.; Kulkarni, D. "The crowd is a collaborative network". In: *Proceedings of the 19th Conference on Computer-Supported Cooperative Work & Social Computing*, 2016, pp. 134-147.
- [GUA15] Guaiani, F.; Muccini, H. "Crowd and laboratory testing can they co-exist? an exploratory study". In: *Proceedings of the 2nd International Workshop on CrowdSourcing in Software Engineering*, 2015, pp. 32-37.
- [HER01] Herbsleb, J. D.; Moitra, D. "Global software development". *IEEE Software*, vol.18-2, Mar 2001, pp.16-20.

- [HER99] Herbsleb, J.D.; Grinter, R.E. "Architectures, coordination, and distance: Conway's law and beyond". *IEEE Software*, vol.16-5, Sep 1999, pp.63-70.
- [HOD10] Hoda, R.; Noble, J.; Marshall, S. "Using grounded theory to study the human aspects of software engineering". In: Proceedings of the 10th Human Aspects of Software Engineering, 2010, pp.5.
- [HOP96] Hoppen, N. "Um guia para avaliação de artigos de pesquisas em sistemas de informação", *RevistaEeletrônica de Administração*, vol. 2-2, Nov 1996, pp. 31-52.
- [Hoß14] Hoßfeld, T.; Keimel, C.; Hirth, M.; Gardlo, B.; Habgit, J.; Diepoldi, K.; Tran-Gial, P. "Best Practices for QoE crowdtesting: QoE assessment with crowdsourcing". *IEEE Transactions on Multimedia*, vol.16-2, Feb 2014, pp.541-558.
- [HOS14a] Hosseini, M.; Phalp, K.; Taylor, J.; Ali, R. "The four pillars of crowdsourcing: A reference model". In: Proceedings of the 8th International Conference, Research Challenges in Information Science, 2014, pp. 1-12.
- [HOS14b] Hosseini, M.; Phalp, K.; Taylor, J.; Ali, R. "Towards crowdsourcing for requirements engineering". In: Proceedings of the 20th International Conference on Requirements Engineering Foundation for Software Quality, 2014, pp. 43-69.
- [HOS14c] Hossfeld, T., Keimel, C.; Timmerer, C. "Crowdsourcing quality-of-experience assessments". *Computer*, vol. 47-9, Sep 2014, pp.98-102.
- [HOS15] Hosseini, M.; Shahri, A.; Phalp, K.; Taylor, J.; Ali, R. and Dalpiaz, F. "Configuring crowdsourcing for requirements elicitation". In: Proceedings of the 9th International Conference on Requirements Engineering Foundation for Software Quality, Research Challenges in Information Science, 2015, pp. 133-138.
- [HOW06] Howe, J. "The rise of crowdsourcing". *Wired Magazine*, vol.14-6, Jun 2006, pp.1-4.
- [HOW08] Howe, J. "Crowdsourcing: "How the power of the crowd is driving the future of business". Random House Business, 2008, 304p.
- [HUT11] Hutter, K.; Hautz, J.; Füller, J.; Mueller, J.; Matzler, K. "Communitition: The tension between competition and collaboration in community-based design contests". *Creativity and Innovation Management*, vol. 20-1, Mar 2011, pp.3-21.
- [JOH11] Johns, T.; Laubscher, R. J.; Malone, T. W. "The age of hyper specialization". *Harvard Business Review*, vol. 89-8, Jul 2011, pp.56.
- [KAG13] Kaganer, E.; Carmel, E.; Hirschheim, R.; Olsen, T. "Managing the human cloud". *MIT Sloan Management Review*, vol. 54-2, Dec 2013, pp.23-32.
- [KAL15] Kalliamvakou, E.; Damian, D.; Blincoe, K.; Singer, L.; German, D.M. "Open source-style collaborative development practices in commercial projects using github". In: Proceedings of the 37th International Conference on Software Engineering, 2015, pp. 574-585.

- [KAZ09] Kazman, R.; Chen, H. M. "The metropolis model a new logic for development of crowdsourced systems". *Communications of the ACM*, vol.52-7, Jul 2009, pp.76-84.
- [KAZ10] Kazman, R., Chen, H.M. "The metropolis model and its implications for the engineering of software ecosystems. In: Proceedings of the 10th Foundation of Software Engineering workshop on Future of software engineering research, 2010, pp.187-190.
- [KIT02a] Kitchenham, B.; Pfleeger, S.L. "Principles of survey research: part 2: designing a survey". *ACM Special Interest Group on Software Engineering Notes*, vol. 27-5, Jan 2002, pp.18-20.
- [KIT02b] Kitchenham, B.; Pfleeger, S.L. "Principles of survey research: part 3: constructing a survey instrument". *ACM Special Interest Group on Software Engineering Notes*, vol. 27-5, Jan 2002, pp. 20-24.
- [KIT02c] Kitchenham, B.; Pfleeger, S.L. "Principles of survey research: part 4: questionnaire evaluation". *ACM Special Interest Group on Software Engineering Notes*, vol. 27-5, Jan 2002, pp.20-23.
- [KIT02d] Kitchenham, B.; Pfleeger, S.L. "Principles of survey research: part 5: populations and samples". *ACM Special Interest Group on Software Engineering Notes*, vol. 27-5, Jan 2002, pp.17-20.
- [KIT08] Kittur, A.; from the LR Chi, E.; Suh, B. "Crowdsourcing user studies with Mechanical Turk". In: Proceedings of the 26th Conference on Human factors in computing systems, 2008, pp. 453– 456.
- [KIT10] Kittur, A. "Crowdsourcing, collaboration and creativity". *XRDS: crossroads, The ACM Magazine for Students*, vol.17-2, Dec 2010, pp.22-26.
- [KIT13] Kittur, A.; Nickerson, V. J.; Bernstein, M.; Gerber, E.; Shaw, A.; Zimmerman, J.; J. Horton, "The future of crowd work". In: Proceedings of the 16th Conference on Computer-Supported Cooperative Work, 2013, pp.1301-1318.
- [KOT05] Kotlarsky, J.; Oshri, I. "Social ties, knowledge sharing and successful collaboration in globally distributed system development projects". *European Journal of Information Systems*, vol.14-1, Mar 2005, pp.37-48.
- [KWA11] Kwan, I.; Schroter, A.; Damian, D. "Does socio-technical congruence have an effect on software build success? a study of coordination in a software project". *IEEE Transactions on Software Engineering*, vol. 37-3, May 2011, pp.307-324.
- [LAK10] Lakhani, D.; Garvin, D.; Lonstein, E. "TopCoder (a): developing software through crowdsourcing". *Harvard Bussines School Case*, vol. 610-32, May 2010, p.20.

- [LAS15] Lasecki, W.S., Kim, J., Rafter, N., Sen, O., Bigham, J.P.; Bernstein, M.S. "Apparition: Crowdsourced user interfaces that come to life as you sketch them". In: Proceedings of the 33rd Conference on Human Factors in Computing Systems, 2015, pp. 1925-1934.
- [LAT13] LaToza, T. D.; Towne, W. B.; van der Hoek, A.; Herbsleb, J. D. "Crowd development". In: Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering, 2013, pp. 85-88.
- [LAT14] LaToza, T. D.; Towne, W. B.; Adriano, C. M.; Van Der Hoek, A. "Microtask programming: Building software with a crowd". In: Proceedings of the 27th Symposium on User Interface Software and Technology, 2014, pp. 43-54.
- [LAT15a] LaToza, T.D.; Chen, M.; Jiang, L.; Zhao, M.; Van Der Hoek, A. "Borrowing from the crowd: A study of recombination in software design competitions". In: Proceedings of the 37th International Conference on Software Engineering, 2015, pp. 551-562.
- [LAT15b] LaToza, T.D., Di Lecce, A., Ricci, F., Towne, W.B.; Van Der Hoek, A. "Ask the crowd: Scaffolding coordination and knowledge sharing in microtask programming". In: Proceedings of the Visual Languages and Human-Centric Computing, 2015 pp. 23-27.
- [LAT15c] LaToza, T.D. and Van Der Hoek, A. "A vision of crowd development". In: Proceedings of the 37th International Conference on Software Engineering, 2015, pp. 563-566.
- [LAT16] LaToza, D. T.; van der Hoek, A. "Crowdsourcing Software Engineering: Models, Motivations, and Challenges.", *IEEE Software*, vol.33-1, Jan 2016, pp. 74-80.
- [LEE15] Lee, C.P.; Paine, D. "From the matrix to a model of coordinated action (MoCA): A conceptual framework of and for CSCW". In: Proceedings of the 18th Conference on Computer-Supported Cooperative Work & Social Computing, 2015, pp. 179-194.
- [LI13] Li, K. "Analysis of the Key Factors for Software Quality in Crowdsourcing Development: An Empirical Study on TopCoder.com". In: Proceedings of the 37th Computer Software and Applications Conference, 2013, pp. 812–817.
- [LI15] Li, W.; Tsai, T. W.; Wu, W. "Crowdsourcing for Large-Scale Software Development Crowdsourcing". *Springer*, vol. 10-1, May 2015, pp. 3-23.
- [MAC14] Machado, L.; Pereira, G.; Prikladnicki, R.; Carmel, E.; de Souza, C. R. "Crowdsourcing in the Brazilian IT industry: what we know and what we don't know". In: Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies, 2014, pp.7-12.
- [MAC16a] Machado, L., Kroll, J., Prikladnicki, R., de Souza, C. R. and Carmel, E. "Software Crowdsourcing Challenges in the Brazilian IT Industry". In: Proceedings of the 18th International Conference on Enterprise Information Systems, 2016, pp. 482-489.

- [MAC16b] Machado, L., Meneguzzi, F., Prikladnicki, R., Carmel, E.; de Souza, C. "Task Allocation for Crowdsourcing using AI Planning". In: Proceedings of the 3th International Workshop on Crowdsourcing in Software Engineering, 2016, pp. 36-40.
- [MAC16c] Machado, L.; Kroll, J.; Marczak, S.; Prikladnicki, R. "Breaking Collaboration Barriers through Communication Practices in Software Crowdsourcing". In: Proceedings of 11th International Conference on Global Software Engineering, 2016, pp. 44-48.
- [MAC17] Machado, L. S.; Zanatta, A. L.; Marczak, S.; Prikladnicki, R. "The Good, the Bad and the Ugly: An Onboard Journey in Software Crowdsourcing Competitive Model". In: Proceedings of the 4th International Workshop on CrowdSourcing in Software Engineering, 2017, pp. 2-8.
- [MAO13] Mao, K.; Yang, Y.; Li, M.; Harman, M. "Pricing crowdsourcing-based software development tasks". In: Proceedings of the 35th International Conference on Software Engineering, 2013, pp. 1205-1208.
- [MAO15a] Mao, K.; Yang, Y.; Wang, Q.; Jia, Y.; Harman, M. "Developer Recommendation for Crowdsourced Software Development Tasks". In: Proceedings of the 9th Symposium Service-Oriented System Engineering, 2015, pp. 347-356.
- [MAO15b] Mao, K.; Licia, C.; Harman, M.; Yue, J. "A Survey of the Use of Crowdsourcing in Software Engineering". Research Note, University College London, 2015, pp.1- 30
- [MEH14] Mehta, D. "An Insight into Software Crowd Sourcing: How Crowd can transform the Business Model for Technology Service Providers". *International Journal of Computer Applications*, vol. 101-12, Jan 2014, pp.34-40.
- [MEL18] Melo, R.R.M.; Machado. L.; Prikladnicki. R.; Souza, C.R.B. "Um Estudo Qualitativo sobre o Crowdsourcing: Análise da Colaboração na plataforma TopCoder". In: Proceedings of XXI Ibero American Conference on Software Engineering, 2018. (*in press*)
- [MÜL14] Müller, H., Sedley, A. and Ferrall-Nunge, E., "Survey research in HCI". In: *Ways of Knowing in HCI*, Springer, 2014, chap. 10, pp. 229-266.
- [NAG12] Nag, S.; Heffan, I.; Saenz-Otero, A.; Lydon, M. "SPHERES Zero Robotics software development: Lessons on crowdsourcing and collaborative competition". In: Proceedings of the Aerospace Conference, 2012, pp. 1-17.
- [NAG13] Nag, S., Katz, J.G. and Saenz-Otero, A., "Collaborative gaming and competition for CS-STEM education using SPHERES Zero Robotics". *Acta Astronautica*, vol. 83, Feb 2013, pp.145-174.

- [NAP13] Naparat, D.; Finnegan, P., "Crowdsourcing Software Requirements and Development: A Mechanism-based Exploration of 'Opensourcing'". In: Proceedings of the 9th Americas Conference on Information Systems, 2013, pp.7.
- [NAY14] Nayebi, M.; Ruhe, G. "An open innovation approach in support of product release decisions". In: Proceedings of 7th the International Workshop on Cooperative and Human Aspects of Software Engineering, 2014, pp. 64-71.
- [OLS00] Olson, G. M.; Olson, J. S. "Distance matters". *Human-Computer Interaction*, vol. 15-2, Sep 2000, pp.139–178.
- [OLS13] Olson, D. L.; Rosacker, K. "Crowdsourcing and open source software participation". *Service Business*, vol.4, Dec 2013, pp. 499-511.
- [OXF17] Oxford Dictionary. Access in: <https://en.oxforddictionaries.com/>. October 2017.
- [PEN14] Peng, X.; Ali Babar, M.; Ebert, C. "Collaborative Software Development Platforms for Crowdsourcing". *IEEE Software*, vol. 31-2, Mar 2014, pp.30–36.
- [PIM12] Pimentel, M.; Fuks, H. "Sistemas colaborativos". Elsevier Editora, 2012, 375p.
- [PRI14] Prikladnicki, R.; Machado, L.; Carmel, E.; de Souza, C. R. B. "Brazil Software Crowdsourcing: A First Step in a Multi-year Study". In: Proceedings of the 1st International Workshop on Crowdsourcing in Software Engineering, 2014, pp. 1-4.
- [RIE17] Riedl, C.; Woolley, A.W. "Teams vs. Crowds: A Field Test of the Relative Contribution of Incentives, Member Ability, and Emergent Collaboration to Crowd-Based Problem Solving Performance". *Academy of Management Discoveries*, vol. 3-4, Dec 2017, pp.382-403.
- [SAR15] Saremi, R.L.; Yang, Y. "Dynamic simulation of software workers and task completion". In: Proceedings of the 2nd International Workshop on CrowdSourcing in Software Engineering, 2015, pp. 17-23.
- [SAR17] Saremi, R.L.; Yang, Y.; Ruhe, G.; Messinger, D. "Leveraging crowdsourcing for team elasticity: an empirical evaluation at TopCoder". In: Proceedings of the Software Engineering in Practice Track, 2017, pp. 103-112.
- [SAX13] Saxton, G. D.; Oh, O.; Kishore, R. "Rules of crowdsourcing: Models, issues, and systems of control." *Information Systems Management*, Jan 2013, vol. 30-1, pp. 2-20.
- [SCH09] Schenk, E.; Guittard, C. "Crowdsourcing: What can be Outsourced to the Crowd, and Why". In: Proceedings of the Workshop on Open Source Innovation, 2009, pp.3.
- [SCH11] Schenk, E.; Guittard, C. "Towards a characterization of crowdsourcing practices". *Journal of Innovation Economics & Management*, vol.1, Nov 2011, pp.93-107.

- [SCH12] Schiller, T.W.; Ernst, M.D. "Reducing the barriers to writing verified specifications". *ACM Special Interest Group on Programming Languages Notices*, vol.47-10, Nov 2012, pp.95-112.
- [SOM11] Sommerville, I., "Software Engineering". Addison-Wesley, 2011, 792p.
- [STO14a] Stol, K.-J.; Fitzgerald, B. "Two's company, three's a crowd: a case study of crowdsourcing software development". In: Proceedings of the 36th International Conference on Software Engineering, 2014, pp.187.
- [STO14b] Storey, M.A.; Singer, L.; Cleary, B.; Figueira Filho, F. and Zagalsky, A. "The (r) evolution of social media in software engineering". In: Proceedings of the on Future of Software Engineering, 2014, pp. 100-116.
- [STO17] Stol, K.J.; Caglayan, B.; Fitzgerald, B., "Competition-Based Crowdsourcing Software Development: A Multi-Method Study from a Customer Perspective". *IEEE Transactions on Software Engineering*, 2017, 19p.
- [STR07] Strauss A.; Corbin, J. M. "Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory". SAGE Publications, 2007, 312p.
- [TAJ13] Tajedin, H.; Nevo, D. "Determinants of success in crowdsourcing software development". In: Proceedings of the 51st Conference on Computers and People Research, 2013, pp.173-178.
- [TAJ14] Tajedin, H.; Nevo, D. "Value-adding intermediaries in software crowdsourcing". In: Proceedings of the 47th Hawaii International Conference on System Sciences, 2014, pp. 1396-1405.
- [TAU17] Tausczik, Y.; Wang, P. "To Share, or Not to Share? Community-Level Collaboration in Open Innovation Contests". In: Proceedings of the *Human Computer Interaction*, vol.1-2, Nov 2017, pp.100-123.
- [TOP17] TopCoder, 2017. Access in: <https://www.topcoder.com>. July 2017.
- [TSA14] Tsai, W.-t.; Wu, W.; Huhns, M. N. "Cloud-Based Software Crowdsourcing". *IEEE Internet Computing*, vol.18-3, May 2014, pp.78–83.
- [WAG05] Wagstrom, P., Herbsleb, J., Carley, K. "A social network approach to free/open source software simulation". In: Proceedings of the 1st International Conference on Open Source Systems, 2005, pp. 16-23.
- [WHI07] Whitehead, J. "Collaboration in software engineering: A roadmap". In: Proceedings of Future of Software Engineering, 2007, pp. 214-225.
- [WHI10] Whitehead, J., Mistrík, I., Grundy, J. and Van der Hoek, A. "Collaborative software engineering: concepts and techniques". *Collaborative Software Engineering*, Springer, 2010, pp. 1-30.

- [WOH13] Wohlin, C; Prikladnicki, R. "Systematic literature reviews in software engineering". In: *Information and Software Technology*, vol.55-6, Jun 2013, pp. 919–920.
- [WOH14] Wohlin, Claes. "Guidelines for snowballing in systematic literature studies and a replication in software engineering". In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, 2014, pp.10.
- [WU13] Wu, W.; Li, W.; Tsa, W. "An evaluation framework for software crowdsourcing". *Frontiers of Computer Science*, vol.7-5, Oct 2013, pp.694-709.
- [YAN15] Yang, Y.; Saremi, R. "Award vs. worker behaviors in competitive crowdsourcing tasks". In: Proceedings of the 9th International Symposium on Empirical Software Engineering and Measurement, 2015, pp. 1-10.
- [YAN16] Yang, Y.; Karim, M.R.; Saremi, R.; Ruhe, G. "Who should take this task? Dynamic decision support for crowd workers". In: Proceedings of the 10th International Symposium on Empirical Software Engineering and Measurement, 2016, p. 8.
- [YIN13] Yin., R. K. "Case Study Research: Design and Methods", series Applied Social Research Methods, 2013, 259p.
- [ZAN16] Zanatta, A. L.; Machado, L.; Pereira,G.; Prikladnicki, R.; Carmel, E. "Software Crowdsourcing Platforms". *IEEE Software*, vol 33-6, Nov 2016, pp.112-116.
- [ZAN17] Zanatta, A. L.; Steinmacher, I.; Machado, L. S.; Souza, C.; Prikladnicki, R. "Barriers Faced by Newcomers to Software-Crowdsourcing Projects". *IEEE Software*, vol.34, Mar 2017, pp. 37-43.
- [ZHA14] Zhao, Y.; Zhu, Q. "Evaluation on crowdsourcing research: Current status and future direction". *Information Systems Frontiers*, vol.16-3, Jul 2014, pp. 417-434.
- [ZHA15] Zhao, M.; van der Hoek, A. "A brief perspective on microtask crowdsourcing workflows for interface design". In: Proceedings of the 2nd International Workshop on Crowdsourcing in Software Engineering, 2015, pp. 45-46.
- [ZHO17] Zhou, W.; Yan, W.; Zhang, X. "Collaboration for success in crowdsourced innovation projects: knowledge creation, team diversity, and tacit coordination". In: Proceedings of the 50th Hawaii International Conference on System Sciences, 2017, pp.381-390.

APPENDIX A

First data collection Interview Guide

Actors	Aspects	Questions
Crowd Requester Platform	CS Initiatives	Do you now CS?
		Tell us about your experience?
		Are you doing micro tasking specifically? Or are you doing more complex task projects?
Requester Crowd	CS platforms	Which platforms have you been used?
		What the number of workers in the crowd?
Requester Platform Crowd	CS Tasks and Projects	Quality envuring: What have you done to achieve and inspect for quality in the solutions submitted?
		Control transparency project management: How do you manage day-to-day tasks?
		Are you a union member? How many hours p week do you work? Are you full-time; part-time? Is this a second job?
Requester	CS payment	Is the company encouraging/discouraging the use of paid CS?
Platform Requester	Business impact	Current CS picture: Number of workers in the crowd, number of the clients?
		By what measure was it successful? What has made this challenge a success?
Crowd Requester Platform	Future	What is the CS scenario for the next three years?

APPENDIX B

Literature Review – Data extraction

We identified how SW CS publications reporting barriers are distributed over the years (see Figure 4). We can observe that it first appears in studies in 2009, and just a few other studies appeared from then until 2012. The last 4 years contain the major number of relevant publications for this study. It makes sense since SW CS has been increasingly adopted in the software industry over the last few years.

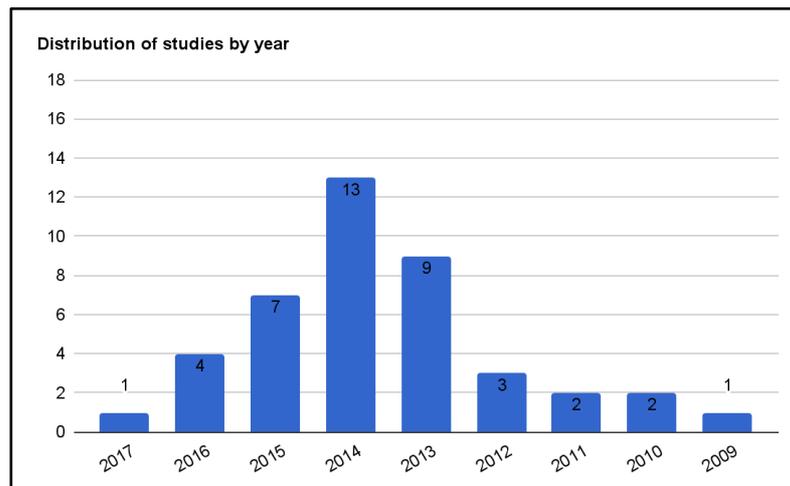


Figure 4. Number of studies by year.

While reading the papers in full, we also classified based on the research approach. We adopted the classification scheme provided by Petersen et al. (2008) to identify the study type:

- *Evaluation research*: techniques and solutions implemented in practice, where an evaluation of the technique was conducted.
- *Solution proposal*: a proposal of a solution for a problem. This solution can be an extension of an existing technique.
- *Validation research*: techniques or solutions investigated that have not yet been implemented in practice.
- *Philosophical papers*: presents new things by structuring the field in the form of a taxonomy or conceptual framework.
- Personal experience papers (1 study): reports the practical experience in a specific topic.
- *Opinion papers*: the personal opinion of somebody on the usefulness of a certain technique, methodology or topic.

Figure 5 shows the classification of the papers based on the categories proposed by Petersen et al. (2008). The majority of the papers report evaluation research (35 studies - 79%) as the type of study conducted, literature review (12 studies - 27%) and, case study (10 studies - 22%) as the method to evidence the problems. Some of the studies that conducted an experiment and case study also, conduct both an interview. An interesting fact here is that few studies report interviews (5 studies) and, only one report ethnograph to collect data.

It is possible to see the lack of studies conducting qualitative analysis by interviews with SW CS actors (platform, crowd and requester) as supporting the existence of collaboration problems.

There is still room available for studies based on interviews, content analysis from repositories of platforms communication's channel and ethnograph.

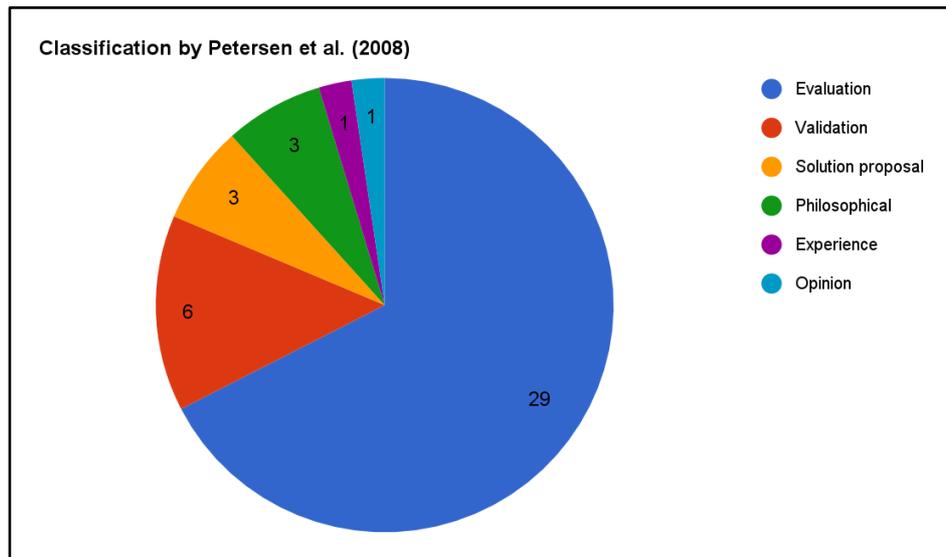


Figure 5. Classification followed by **Petersen et al. (2008)**.

We can also observe that the data analyzed by the studies are predominantly gathered from TopCoder and Amazon Mechanical Turk (AMT) software platforms. Other data used by the studies include tools or environment to simulate CS platforms and, it's were development by own authors of the papers. Regards to research methods, the papers reported a mix-method study to evidence the problems such as surveys, interviews, case study, and controlled experiments.

APPENDIX C

Case Study_ Report 2_TopCoder Open questions

PUCRS – Faculdade de Informática – PPGCC
Desenvolvimento Colaborativo de Software – Profa. Sabrina Marczak

Software Crowdsourcing: Entrega 2 – Experiência com a Plataforma TopCoder

Nome Completo: <indique aqui seu nome>

Instruções Gerais

As questões abaixo devem ser preenchidas INDIVIDUALMENTE baseadas na sua EXPERIÊNCIA PESSOAL de uso da plataforma TopCoder. Ou seja, na TAREFA (ou tarefas) que você realizou e relatou na Entrega 1 do projeto. Você deve considerar as questões e respondê-las SEM discussões prévias com os colegas de aula. Qualquer similaridade, total ou parcial, de conteúdo identificadas como “plágio de ideias” entre dois ou mais colegas terão como resultado a anulação da avaliação dos colegas envolvidos. Ainda, responda às questões de maneira crítica. Reflita sobre o que a questão indaga, reveja suas atividades, anotações, etc e criticamente expresse sua opinião. Caso você tenha realizado mais de uma tarefa, por favor, referencie aspectos relacionados a cada uma das mesmas quando se aplicar.

Questões

1. Explique como se deu o processo de seleção da tarefa que você realizou na plataforma. Por exemplo, como você decidiu por qual tarefa realizar? Você decidiu por uma tarefa e por algum motivo acabou realizando uma outra? Qual a razão? Nota: Toda e qualquer informação relevante para o entendimento do motivo da seleção da tarefa realizada é de interesse e deve ser relatado.

<Insira aqui a sua resposta>

2. Que aspectos você considerou interessante na sua experiência em *software crowdsourcing* com a plataforma TopCoder?

<Insira aqui a sua resposta>

3. Colaboração é considerado um aspecto importante durante o processo de desenvolvimento de *software*, conforme foi discutido na disciplina. Como você percebeu a colaboração em *software crowdsourcing* na plataforma TopCoder?
 - Entre os membros da plataforma
 - Entre a plataforma (ou cliente) e os membros da plataforma

<Insira aqui a sua resposta>

4. Você acredita que a plataforma utilizada suporta a colaboração? Justifique sua resposta.

<Insira aqui a sua resposta>

- 5. Na sua opinião, quais as dificuldades para colaboração você enfrentou na plataforma ou durante a participação da tarefa? Se você acredita que não encontrou nenhuma dificuldade, explique sua razão por ter esta crença.**

<Insira aqui a sua resposta>

- 6. Quais as suas sugestões para minimizar as dificuldades de colaboração encontradas na plataforma ou na participação de uma tarefa? Nota: A questão não precisa ser respondida caso você tenha respondido que não enfrentou dificuldades na Questão 5.**

<Insira aqui a sua resposta>

- 7. Quais são as atividades/etapas, na sua opinião, que mais necessitam de suporte durante a participação na plataforma?**

<Insira aqui a sua resposta>

- 8. Que outros tipos de dificuldades foram encontradas por você durante a participação na plataforma? As dificuldades podem ser de ordem pessoal e/ou técnica, tanto na utilização da plataforma como na realização da tarefa.**
- Na utilização da plataforma
 - Na realização da tarefa

<Insira aqui a sua resposta>

- 9. Que sugestões você propõe para minimizar estas dificuldades? Nota: A questão não precisa ser respondida caso você tenha respondido que não enfrentou dificuldades na Questão 8.**

<Insira aqui a sua resposta>

APPENDIX D

Confidentiality Term - SW CS Case Study



Faculdade de Informática /PUCRS
 Programa de Pós-Graduação em Ciência da Computação
 Avenida Ipiranga, 6681 – Prédio 32 - 90619-900 – Porto Alegre – RS

Termo de Consentimento Livre e Esclarecido

A PUCRS, por meio do Grupo de Pesquisa MUNDDOS, da área de Engenharia de Software da Faculdade de Informática, agradece aos participantes por sua contribuição ao entendimento de problemas reais enfrentados por equipes de desenvolvimento de software, os quais são de valia para o avanço da geração de conhecimento. O objetivo deste estudo, além de oferecer a experiência prática, foi identificar e entender como a colaboração se manifesta e quais são as principais barreiras enfrentadas por usuários novatos na realização de atividades de software crowdsourcing.

Gostaríamos de poder compartilhar os resultados com outros grupos de pesquisa e/ou comunidade acadêmica que também estão trabalhando com seus alunos na discussão sobre a visão de futuro da engenharia de software ao se realizar projetos práticos experimentando a realização de tarefas do mundo real.

No caso da sua concordância de uso dos resultados das atividades conduzidas para tais fins, suas percepções já compartilhadas serão usadas para fomentar a reflexão sobre o tema usufruindo de recursos de análise científica dos dados e a posterior divulgação dos resultados. Cabe ressaltar que:

1. **O anonimato dos participantes será preservado em todo e qualquer documento divulgado** em foros científicos (tais como conferências, periódicos, livros e assemelhados) ou pedagógicos (tais como apostilas de cursos, *slides* de apresentações, e assemelhados).
2. **Todo participante terá acesso às cópias destes documentos após a publicação dos mesmos.**
3. Todo participante que se sentir constrangido ou incomodado com o uso de seus resultados para o projeto de pesquisa pode solicitar o não uso de seus dados a qualquer momento e estará fazendo um favor se registrar por escrito as razões ou sensações que o levaram a esta atitude. Os dados serão então obrigatoriamente removidos.
4. Todo participante tem direito de expressar por escrito qualquer restrição ou condição adicional que lhe pareça aplicar-se aos itens acima enumerados (1, 2 e 3). A equipe conduzindo a pesquisa se compromete a observá-las com rigor e entende que, na ausência de tal manifestação, o participante concorda que sejam o comportamento ético da equipe somente as condições impressas no presente documento.
5. A equipe tem direito de utilizar os dados das atividades, mantidas as condições acima mencionadas, para quaisquer fins acadêmicos, pedagógicos e/ou de desenvolvimento contemplados por seus membros.

[a ser preenchido pelo entrevistador]
Forma: _____ Data: __/__/____
Condições especiais (caso não haja condições especiais, escreva "nenhuma"):

<input type="checkbox"/> continua no verso

Por favor, indique sua posição em relação aos termos acima:
<input type="checkbox"/> Estou de pleno acordo com os termos acima.
<input type="checkbox"/> Em anexo registro condições adicionais para este teste.

Assinatura do participante

Assinatura do responsável (caso o participante seja menor de idade)

Assinatura do pesquisador

Nome do Participante:

Pesquisador Responsável: – Sabrina Marczak, Faculdade de Informática - PUCRS

APPENDIX E

Emails VP TopCoder Contact

2/19/2018

Gmail - Survey on TopCoder - research



Leticia Machado <leticia.smachado@gmail.com>

Survey on TopCoder - research

5 messages

Leticia Machado <leticia.smachado@gmail.com>

Mon, Jan 22, 2018 at 12:52 PM

To: Dave Messinger <[REDACTED]>

Cc: Rafael Prikładnicki <[REDACTED]>, Cleidson de Souza <[REDACTED]>

Dear Dave,

A while ago Prof. Erran Carmel from American University met you at Software Engineering conference (ICSE) in Texas and we exchanged some emails about software crowdsourcing.

After that, I briefly talked to LaMora, the contact that you introduced us, but we did not have the opportunity to explain our ideas for collaborating with TopCoder. I'd like to resume this contact now.

In my PhD I've been exploring how communication and collaboration among coders during the challenges can improve the quantity and quality of submissions. This is confirmed by other studies from other researchers from UC, Irvine, Stevens Institute of Technology and Tausczik and Wang from University of Maryland.

In this way, I am wondering whether you could give me (and my advisors) permission to distribute a web survey among coders participating in challenges. The idea is distribute /sent a web survey/ to a potentially large TopCoder's members and to hear from the community itself. Results of this survey would be used in my dissertation, but more importantly, they can provide you insights about ways to increase /boost the number of workers who submit solutions to challenges instead of only registering for tasks.

We also are open to explore other issues that might interest TopCoder. In other words, in our survey we could include additional questions that you might find relevant.

The survey responses will not contain any identifying information about TopCoder's members. All research data collected will be stored securely and confidentially.

What do you think? Do you want to schedule a conversation so that I can better explain my ideas?

Thanks in advance,

 Leticia Santos Machado
 Ph.D. student in Computer Science - PUCRS - www.pucrs.br
<http://munddos.com.br/projects/>
 Assistant Professor at UNISINOS - www.unisinos.br - leticiasa@unisinos.br
<http://lattes.cnpq.br/2910882986298754>

Email Group Survey Participants

Prezado aluno,

Você está recebendo este email pois foi aluno da Profa. Sabrina Marczak no semestre passado.

Estamos finalizando a pesquisa de doutorado da aluna Leticia Machado em Software Crowdsourcing e estamos interessados em entender como a comunicação e a colaboração entre os desenvolvedores durante os desafios da plataforma TopCoder podem melhorar a quantidade e qualidade de soluções submetidas.

Nós precisamos da tua ajuda e ficaremos muito grato se você puder nos ajudar a entender isso melhor respondendo as questões disponíveis aqui: <https://goo.gl/forms/rpdGcGNjgV8JsWW52>.

Esta é uma pesquisa puramente acadêmica, sem interesses comerciais. Nós iremos compartilhar os resultados publicamente para que todos possam se beneficiar com eles mas, eles serão tratados de forma anônima e confidencial.

A Leticia é orientada por mim e a tua participação neste estudo é totalmente voluntária.

Agradeço desde já a tua participação!

Qualquer dúvida podem entrar em contato direto com a Leticia (leticia.machado.001@acad.pucrs.br).

Obrigado.

Prof. Rafael Prikladnicki

Invitation post in the general TopCoder's forum

https://apps.topcoder.com/forums/?module=Thread&threadID=912721&start=0&mc=3#2244960

Apps Miscelânea Blogs Professional Crowd Leituras PUCRS EUA Convert ft to cm - C... São Francisco - Ond... GitHub - Rhapsod/s... Other Bookmarks

Forums

Select a Forum

Search | Watch Thread | My Post History | My Watches | User Settings

View: Flat (newest first) | Threaded | Tree

Previous Thread | Next Thread

Forums > Development Forums > PseudoVet - Convert Aging Algorithm Into REST API Python Challenge v1.0 > Code Questions > Help

Help | Feedback: (+0/-0) | [+ [-] | Reply | Edit

Sun, Feb 11, 2018 at 2:40 PM EST

 **leticiars**
5 posts

Hello, I am a PhD student from Brazil. I am conducting a research aiming to finding how communication among developers can improve the solution's submission rate. My goal is to hear from the developer's community itself, i.e., from you who has participated in the TopCoder's challenges. I need your help answering some questions available here: <https://goo.gl/forms/AoowpLhiGCcW9kkj2>. You will not take more than 10 minutes, and the survey responses does not contain any identifying information about you unless a specific question in the survey has asked for this. Thanks in advance,

Re: Thread "Help" in category "PseudoVet - Convert Aging Algorithm Into REST API Python Challenge v1.0" has been updated by leticiars (response to post by leticiars) | Sun, Feb 11, 2018 at 9:13 PM EST

Feedback: (+0/-0) | [+ [-] | Reply

 8339 posts

Please do not spam the challenge forums with such inputs. You can post in the general Topcoder forums for this.

Suporte

APPENDIX F

Survey Google forms

Communication and Collaboration in Crowdsourcing Software Development

Hello!

We're researchers from Brazil in the Pontifical Catholic University and we're interested in how software developers communicate and collaborate during a TopCoder competition.

We'd be grateful if you could help us understand this better by filling the survey below. It should only take about 10-12 minutes of your time. Note this is a purely academic research project with no commercial interests. We will OPENLY PUBLISH the results so everyone can benefit from them, but will ANONYMIZE everything before doing so. We will handle your response confidentially.

Thanks a lot for participating!

Leticia Santos Machado (PUCRS University) leticia.machado.001@acad.pucrs.br

Rafael Prikladnicki (PUCRS University) rafael.prikladnicki@pucrs.br

Cleidson de Souza (Federal University of Pará) - cleidson.desouza@acm.org

* Required

Basics

1. **1. What is your experience in software development? ***

Mark only one oval.

- 1 year or less
- Between 2 and 5 years
- Between 5 and 10 years
- Between 10 and 20 years
- More than 20 years

2. **2. How many development challenges have you participated (registration phase) on TopCoder? ***

Mark only one oval.

- No, never
- Yes, once
- Yes, twice
- Yes, between three and five times
- Yes, more than five times

3. **3. Have you ever submitted any solution for challenges in the TopCoder? ***

Mark only one oval.

- No, never
- Yes, one time
- Yes, twice
- Yes, between three and five times
- Yes, more than five times

4. **4. Have you ever won any challenge on TopCoder? ***

Mark only one oval.

- No, never
- Yes, one time
- Yes, twice
- Yes, between three and five times
- Yes, more than five times

Communication

5. **1. Do you use the TopCoder forums to communicate with the co pilot or other crowd members during a challenge? ***

Mark only one oval.

- Yes
- No

6. **2. Do you agree or disagree with the following statement: "For me, engaging in communication with other crowd members is beneficial during a TopCoder challenge". ***

Mark only one oval.

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	strongly agree				

7. **3. Regarding question 2, WHY do you agree/disagree? Can you think about an example or situation?**

8. 4. Do you agree or disagree with the following statement: "Communication via TopCoder's forums (send ou read posts) helps me to SUBMIT a task's solution". *

Mark only one oval.

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	strongly agree				

9. 5. Regarding question 4, WHY do you agree/disagree? Can you think about an example or situation?

10. 6. Do you agree or disagree with the following statement: "Communication via TopCoder's forums (send ou read posts) helps me to WIN a competition". *

Mark only one oval.

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	strongly agree				

11. 7. Regarding question 6, WHY do you agree/disagree? Can you think about an example or situation?

12. **8. How often do you use each of the following communication channels to interact with CROWDS' MEMBERS in order to gather any information to apply in your task solution on TopCoder? ***

Mark only one oval per row.

	Not used at all	Not used	Neutral	Used	Very used
TopCoder Forums	<input type="radio"/>				
Microblog (Twitter, Tumblr,...)	<input type="radio"/>				
Code hosting sites (GitHub, BitBucket, Google Code, SourceForge)	<input type="radio"/>				
Question & Answer Sites (Stack Overflow, Quora,...)	<input type="radio"/>				
Discussion Groups (Mailing Lists, Google Groups,...)	<input type="radio"/>				
Private Discussions (Email,...)	<input type="radio"/>				
Public Chat (IRC, Slack...)	<input type="radio"/>				
Private Chat (IM, Skype, Google Chat,...)	<input type="radio"/>				
Professional Networking Sites (LinkedIn, Xing,...)	<input type="radio"/>				
Social Network Sites (Facebook, Google Plus, Instagram...)	<input type="radio"/>				

13. **9. How often do you use each of the following communication channels to interact with OTHER DEVELOPERS (who are NOT participating in the challenge) in order to gather any information to apply in your task solution on TopCoder? ***

Mark only one oval per row.

	Not used at all	Not used	Neutral	Used	Very used
TopCoder Forums	<input type="radio"/>				
Microblog (Twitter, Tumblr, ...)	<input type="radio"/>				
Code hosting sites (GitHub, BitBucket, Google Code, SourceForge, ...)	<input type="radio"/>				
Question & Answer Sites (Stack Overflow, Quora, ...)	<input type="radio"/>				
Discussion Groups (Mailing Lists, Google Groups,...)	<input type="radio"/>				
Private Discussions (Email, ...)	<input type="radio"/>				
Public Chat (IRC, Slack,...)	<input type="radio"/>				
Private Chat (IM, Skype Chat, Google Chat, ...)	<input type="radio"/>				
Professional Networking Sites (LinkedIn, Xing, ...)	<input type="radio"/>				
Social Network Sites (Facebook, Google Plus, ...)	<input type="radio"/>				

Closing

Thank you for your participation! We are almost there.

14. **1. Would you like to receive an email when we publish the results of our survey?**

Mark only one oval.

- Yes, please!
- Nope!

15. **2. Would you be up for a short voice interview (Skype, Hangouts,...) so we can learn more about your response?**

Mark only one oval.

Yes, I'd do that!

Nope!

16. **3. What is your email address? We will only email you if you checked one of the two options above.**

17. **4. How old are you?**

18. **5. Which country do you live in? ***

19. **6. What is your gender?**

20. **7. Thanks so much for getting this far! Any questions, comments or concerns you'd like to tell us about?**
