IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 4, 2020, accepted March 11, 2020, date of publication March 23, 2020, date of current version April 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2982619

Crowdsourcing in Software Development: Empirical Support for Configuring Contests

STAMATIA BIBI[1], IOANNIS ZOZAS[1], APOSTOLOS AMPATZOGLOU[2], PANAGIOTIS G. SARIGIANNIDIS[1], GEORGE KALAMPOKIS[1], AND IOANNIS STAMELOS[3]

[1]Department of Electrical and Computer Engineering, University of Western Macedonia, 50100 Kozani, Greece
[2]Department of Applied Informatics, University of Macedonia, 54636 Thessaloniki, Greece
[3]Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece

Corresponding author: Stamatia Bibi (sbibi@uowm.gr)

This work was supported by the European Union under the Erasmus KA2 grants and the project ARRANGE-ICT, Grant 2018-1-BG-01-KA203-048023.

ABSTRACT Despite the extensive adoption of crowdsourcing for the timely, cost-effective, and high-quality completion of software development tasks, a large number of crowdsourced challenges are not able to acquire a winning solution, on time, and within the desired cost and quality thresholds. A possible reason for this is that we currently lack a systematic approach that would aid software managers during the process of designing software development tasks that will be crowdsourced. This paper attempts to extend the current knowledge on designing crowdsourced software development tasks, by empirically answering the following management questions: (a) what type of projects should be crowdsourced; (b) why should one crowdsource—in terms of acquired benefits; (c) where should one crowdsource—in terms of application domain; (d) when to crowdsource—referring to the time period of the year; (e) who will win or participate in the contest; and (f) how to crowdsource (define contest duration, prize, type of contest etc.) to acquire the maximum benefits—depending on the goal of crowdsourcing. To answer the aforementioned questions, we have performed a case study on 2,209 software development tasks crowdsourced through TopCoder platform. The results suggest that there are significant differences in the level to which crowdsourcing goals are reached, across different software development activities. Based on this observation we suggest that software managers should prioritize the goals of crowdsourcing, decide carefully upon the activity to be crowdsourced and then define the settings of the task.

INDEX TERMS Crowdsourcing, software development, success factors, crowd factors, cost, duration.

I. INTRODUCTION

The term crowdsourcing combines two words: crowd and outsourcing. Formally, Howe [12]—who first coined the term in 2006—defined crowdsourcing as ''the act of taking a job traditionally performed by a designated agent and outsourcing it to an undefined, generally large group of people in the form of an open call''. In this context crowdsourcing is a new business model that enables the co-creation between a ''*provider*'' that is the one that sets the details of the problem, the ''*supplier*'' / *''crowd''* that suggests a solution and the *''host''* who offers the crowdsourcing platform, enabled by Web 2.0 [32], presenting the problem formulated by the provider to the crowd. Crowdsourced Software Engineering (CSE) gained an increasing interest by both industry and academia, appearing to be a promising approach for completing specific software engineering tasks. CSE as a means to leverage the power of the crowd over task completion is not a new research topic, considering that the development paradigm of Open Source Software [17] is popular from the late 90s, but as a formulated contest-based software development process, has been an emerging hot trend only the last years [5]. According to Latoza and Van Der Hoek [17] *''contests''* as a crowdsourcing model, are similar to the outsourcing model, in the sense that a client requests work and pays for its completion, but differs as it treats the crowd as contestants rather than collaborators. When formulating a contest, the provider suggests the task to

The associate editor coordinating the review of this manuscript and approving it for publication was Chintan Amrit.

be crowdsourced, that may cover requirements, architecture, user interface design, implementation, and testing, that can be completed in a number of days. Contestants (the crowd) provide each one of them a competing solution; from which a winning one is selected and apprized. Current evidence suggests that crowdsourcing contests are more successful when associated with the completion of micro-tasks [37], [7], yet there are some examples where large, innovation projects are successfully crowdsourced [12].

Despite the dominating feeling that crowdsourcing is a low risk alternative to outsourcing tasks, the possibility of failure is non-negligible [20]. According to Dahlander and Piezunka [5] only 10% of crowdsourced contests are able to attract the desired amount of contributions, while in 50% of the contests there is no contribution at all. The success of crowdsourced software development is closely related to the reliability of the participating crowd-workers [1] and their experience [37] and therefore it is important for contest providers to attract the suitable contributors [7]. According to Weidema *et al.* [37] many participants find the crowdsourced tasks to be difficult despite the fact they are of limited scope. A possible reason for this perception might be that the tasks crowdsourced are poorly oriented [10], or even the fact that the tasks are not suitable for being crowdsourced [5]. Another problem that crowdsourcing contests deal with is the quality of the submitted solutions [37] and the fact that the overall costs of contests are underestimated [10]. Summarizing, crowdsourcing contests apart from failing directly (i.e., they are not able to acquire a winning solution), they very frequently fail indirectly. In particular, if a solution comes late, it might not be relevant or useful for the contest provider; if it is of low quality then, it might not be ready for being brought into market; or if it is overpriced, it might not be beneficial decision to not build the product in-house.

Therefore, it is of paramount importance to carefully make decisions during the planning phase of a crowdsourcing project [14]. Focusing on the decision phase and inspired by the 5W+1H model [11] to improve project management efficiencies in this paper we aim to provide assistance to a contest provider in order to help him clearly answering the following questions:

- **What** to crowd-source? To answer this question, we will carefully examine the different tasks that can be crowdsourced and their performance with respect to solution acquisition, the required time, the cost, and the quality of the solution acquired. Traditional software engineering activities that can be considered for crowdsourcing are design, development and testing tasks. Also more specialized activities such as cognitive tasks (i.e. Artificial Intelligence solutions) can be considered. Such an answer will help the provider decide upon the specific task that can be assigned for crowdsourcing.
- **Where** to crowd-source? This question extends the previous one, emphasizing on the application domain of the software engineering challenges that can be crowdsourced. Our interest here is to find the types of the applications that are more likely to be successful when assigned to an online community. Such application domains can be scientific applications, media applications, civil engineering applications or even business applications among others. Similarly, this question will help the provider realize whether the application domain of the task planned to be crowd-sourced has the potentials to be successful.
- **Why** to crowd-source? The answer to this question is related to the goals that a provider is expecting to achieve while adopting crowd-sourcing solutions. Apart from the delivery of the end-product/artifact crowdsourcing goals may include cost savings, quick solution acquisition (time reduction), solution diversity (that includes more than one possible solutions) and increased quality. A provider needs to clarify which of these goals are the most important in each contest and focus so as to carefully set the configuration parameters of the contest.
- **When** to crowd-source? In this question we will examine the distributions of challenges over the calendar year and draw conclusions on the activity of each month, the completion of successful contests and the crowd participation. Our target is to help the contest provider, define the optimal time period to launch a new challenge so as to maximize the likelihood of achieving crowdsourcing goals.
- **Who** is the crowd? The answer to this question is related to the profiles of the winners and the participants of the contest, their competencies and their relative performance. Also the profiles of participants and the ''quitters'' of the contests (low reliability) will be analyzed. Such information will be useful for contest providers to better orient the contests based on the experience and expertise of the community of 'crowd workers' they are expecting.
- **How** to crowd-source? In this question we examine the relevant factors that orient a crowdsourcing contest, in-cluding the configuration parameters in the problem statement of the challenge. Such parameters can be the type of the contest, the prize, the duration and the number of winners. Obviously all the 5W questions are related to the current question and thus a contest provider needs to carefully answer all of them to be able to set a new contest that has the potentials to succeed. The decision making phase is further exemplified in Fig.1.

This study explores the ''How'' question by taking into account the ''Why'' question in the sense that a manager should consider the level to which crowdsourcing goals are achieved across varying software development tasks (''What''), different application domains (''Where''), time periods that the contest will be announced (''When'') and the profiles of 'crowd' workers (''Who'').

For the purpose of this study we have performed an exploratory case study on 2,209 projects, crowdsourced by the popular TopCoder platform during 2018. The data involve a variety of different software development challenges that

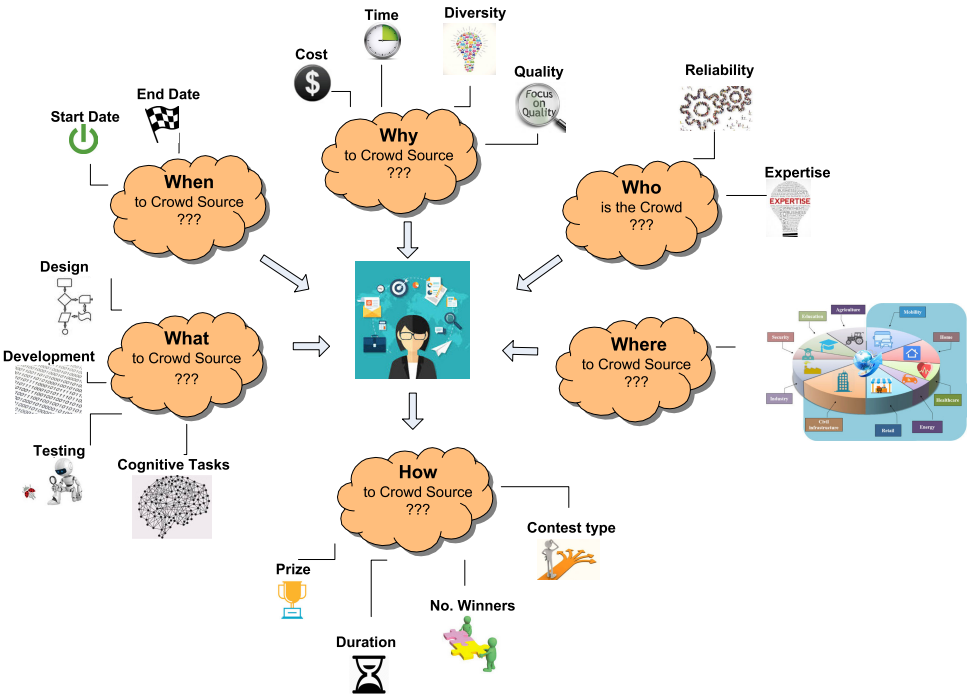

FIGURE 1. Decision making phase for crowdsourcing software.

were crowdsourced through different types of contests and apprized by specific monetary rewards. The challenges were further classified to different types of tasks and application domains. We investigated if there are statistically significant differences among the success indicators of different types of challenges, examined the distribution of contest types and the level to which each type is preferred when different challenges are crowd sourced. All the results were summarized in evidence-based models that provide assistance to contest providers when deciding upon the activity to be crowd sourced and the settings of the chal-lenge, with respect to the crowdsourcing goals.

The rest of the paper is organized as follows: In Section II we present an overview of related work and in Section III we present the case study design, in Section IV we provide the results, organized by research question, and discuss them in Section V. In Section VI we present the threats to validity of our study, and in Section VII, we conclude the paper.

II. RELATED WORK

Crowdsourced Software Engineering (CSE), as a development model, has been an emerging trend in the last decade [2], facing new types of management challenges determining its operational success. In this section we will refer to the different CSE management challenges that have been tackled by the related work by taking into consideration the perspective of the 5W+1H model. In particular we will present works related to the questions:

- Why to crowdsource?
- What to crowdsource?
- Where to crowdsource?
- Who will be the crowd?
- When to crowdsource?
- How to crowdsource?

A. WHY TO CROWDSOURCE?

According to Stol *et al.* [28], the main benefits acquired from crowdsourcing software are:

Cost reduction that is achieved by exploiting lower development costs in certain regions [28] and by offering new types of compensation models such as experience gain, and recognition based systems [16] instead of typical monetary prizes. Another reason that supports the claims of cost reduction is that crowdsourcing model is a form of outsourcing and therefore it inherits the typical benefits of outsourcing such as the avoidance of insourcing costs (i.e hire overheads, know how acquisition, application of new processes) [18], [28], [31].

Time reduction that is achieved due to the parallelization of decomposed tasks [17], [27] and the potential to access a pool of experienced developers [14], [16], [31] that can increase productivity and accelerate development. Additionally according to Stol *et al.* [28], motivated geographically distributed workers are willing to work during weekends, exploiting the different time zones so as to achieve a full 24h productive day.

Higher quality is also a potential benefit acquired from crowdsourcing [4]. According to Latoza and Van Der Hoek [17] crowdsourcing, as a process, promotes the generation of alternative solutions to a given problem.

From these solutions the provider may select or even combine the most appropriate ones, obtaining at the end higher quality solutions. This argument is further supported by Stol *et al.* [28] who emphasize that broad participation of the crowd provides access to experienced developers that self-select the task based on their skills and therefore attempt to offer the highest quality solution in order to win.

Crowd innovation is the last but the most important motivation to crowdsourcing tasks. The ''wisdom of the crowd'' [12], the democratization of participation [17], the open creativity that many times outmatches the fixed mindset that exists within individual companies [28], [30] is the main driver for selecting and designing crowdsourced software development contests. It seems that crowd participation is among the most important goals of crowdsourcing since engaging the right ''crowd'' the contest provider can obtain all the aforementioned benefits regarding time and cost reduction along with quality assurance. Solution diversity obtained by broad participation is also another benefit leveraging from crowdsourcing [38].

B. WHAT TO CROWDSOURCE?

Sarı *et al.* [26] mention that crowdsourcing has been applied to various process areas in software engineering with coding tasks being the most popular one. Task selection of the software development activities crowdsourced is a great challenge for stakeholders as micro-tasks are more easy to complete according to Latoza and Van Der Hoek [17], but bigger tasks may under the right environment leverage the most the power of the crowd. Stol *et al.* [28] mention that effective task decomposition can help the organization leverage the most the power of crowdsourced development.

Thuan *et al.* [34] offer a theoretical framework to support decision making regarding the type of task to be crowdsourced considering four types of tasks based on their properties: internet tasks, interactive tasks, sensitive tasks and partitioned tasks concluding that internet, interactive tasks are easier to crowdsource. As for task management, Dissanayake *et al.* [6] explored task division practices in team based competitions by analyzing data from the Kaggle platform, finding that team leader's social capital and team expert's intellectual capital affect the performance of a team that is accelerated in contests that are less competitive. Though in this study the authors do not refer to the specific types of tasks crowdsourced but rather examine the efficacy of task division practices. Similarly Yu *et al.* [40] explored task assignment and division in collaborative crowdsourced development.

The level of success of crowdsourced micro-task completion, and in particular software interface design tasks, was studied in [37]. The authors conducted experiments with Amazon Mechanical Turk workers and noted that it is feasible for the crowd to generate a large number of alternative solutions, though their quality is highly differentiated. Yang *et al.* [39] proposed a methodology, based on ranking, to recommend tasks to crowd workers taking into consideration, among other factors, the average submission quality on similar tasks and the overall submission rate of each crowd worker.

We can observe that despite the fact that task selection and decomposition is considered a very important success factor when crowdsourcing software [6], [10], [17], [18], [31], [33], we cannot find any study that compares the efficiency of crowdsourcing different types of software engineering tasks in terms of the success indicators described in section II-A.

C. WHERE TO CROWDSOURCE?

The ''Where'' question may refer to two things: (a) the platform in which a software development contest will be hosted and (b) the application domain where the solution derived from the crowdsourced contest is deployed. Regarding the first interpretation according to Mao *et al.* [21] and Wu *et al.* [38] there are several online platforms available that can currently host software development challenges. Several of these platforms support all types of software development tasks[1] (TopCoder, Bountify) while others support specific tasks [2] such as testing and mobile development (uTest, TestBirds). According to Mao *et al.* [21] TopCoder is a pioneer for practicing crowdsourced software engineering and the dominant platform when selecting the medium to host contents. Additionally this platform is used in the majority of research studies performed for determining success factors of crowdsourced software development [1], [2], [19], [36], [38], [39]. However, we were not able to identify any study that compares the efficacy of different platforms. Regarding the second interpretation of the where question, we were not able to find empirical evidence on the application domain that is more popular for crowdsourcing contests.

D. WHO WILL BE THE CROWD?

Several studies have explored the profile and the characteristics of the ''crowd'' participating in software development contests. The crowd motivation and incentives are recognized as the most important factors from the crowd perspective affecting CSE success [17], [18], [22]. A contributor can be highly motivated to participate in a contest because of the recognition received [19], or the monetary prize [21] or the acquisition of experience [29]. The crowd size necessary to tackle a problem is appointed by Latoza and Van Der Hoek [17], as an important dimension of CSE. According to Tajedin and Nevo [33] the size of the crowd also has a positive effect on the crowd composition that affects CSE success considering the fact that when more people are attracted to a project the chances of receiving innovative, diversified solutions are increased. Crowd reliability is recorded also as an important factor by Mehta [22] and Yang *et al.* [39] referring to the level to which registered contributors submit, at the end, a solution and whether this solution is of the expected quality. Crowd experience is part

[1] www.topcoder.com, www.bountify.co
[2] www.utest.com, www.appstori.com

of the CSE success according to Li [19], as it is a factor that can affect the quality of the end product solution.

E. WHEN TO CROWDSOURCE?

Numerous studies can be found in literature mentioning the importance of selecting the right period to crowdsource a project [30]. Stol *et al.* [30] appoint that providers need to schedule a contest during the right period so as to ensure that sufficient number of workers are available when needed. While there may be extensive expertise within the crowd, it might not be available at the moment when it is needed [14].

Despite the aforementioned, we were not able to find any case studies answering the *When* question directly. Li *et al.* [19] approximated time by introducing a variable representing the contest platform ''traffic'', measured as the number of other projects posted when the present project is being crowdsourced. Li *et al.* [19] examined the influence of platform traffic on the final quality of the crowdsourced project. He concluded that the quality of a crowdsourced project is increased when it is posted in a prosperous period (i.e. a number of other challenges are communicated through the platform).

F. HOW TO CROWDSOURCE?

In this section, we present an overview of the experimental research performed on *how* to crowdsource software development. We mainly refer to the settings and environmental parameters that orient a contest and affect its performance in terms of the quality of the solution acquired and the crowd participation and engagement. We should mention that we were not able to find any studies examining the contest parameters that affect CSE success in terms of cost and time.

Archak [2] was one of the first to explore efficient crowdsourcing mechanisms with respect to *quality factors.* The findings of his study highlighted the influence of special factors like payment and project requirements over the final quality of the delivered project. Li in [19] tested 23 software quality factors based on platform and project nature and performed an experimental analysis on crowdsourced reusable projects concluding that among the four aspects that affect CSE quality and should be considered when designing contests are the *time period* when a project is posted, the *size of the project*, the *participation of experienced developers* and the level to which the *design documents* of the project are of high quality. Sohibani *et al.* [27] recorded quality factors based on the findings from questionnaires answered by participants of 5 different crowdsourcing platforms YouTube, Amazon Mechanical Turk, Wikipedia, Rally Fighter and Kick Starter appointing as well that the crowd company workers experience is very important. Wang *et al.* [36] suggested a new quality metric for assessing crowdsourced software development projects, the effort level. The effort level is calculated as a bi-product of 5 parameters: duration, payment, specification length and number of links, and technology requirements.

Crowd participation was found to be influenced by the clarity of the description of the associated tasks [35]. Tasks with unclear objective description, without specifying required technologies or environment setup instructions, discourage developers from selecting them. Crowd participation was also examined by Stol *et al.* in [28] who concluded that the duration and the prize of contests do not significantly affect crowd participation. On the other hand, the number of competitions that run in parallel within a project has a significant negative effect on the crowd's interest in a competition. Alelyani and Yang [1] explored *crowd reliability* by investigating possible connections between the nature of the crowd and crowdsourced tasks. Data regarding workforce reliability (in terms of complete task submission and thus participation), task registration speed, task completion duration, skills (on programming language knowledge list) and challenge types (on the nature and rewards of each contest) were analyzed and signalized as catalyst factors for success. Similarly, Dwarakanath *et al.* [8] assessed the *crowd trustworthiness* based on submission quality, timeliness and ownership concluding that task requirements, user efficacy and reputation strongly influence the trustworthiness of the crowd. Karim *et al.* in [13] proposed a recommendation system that can help mainly crowd workers (and as a side effect providers) on taking over the appropriate tasks based on technology requirements and their skills. The aforementioned factors are also explored by Saremi *et al.* [25] who explored team reliability, velocity and reliability.

G. CONTRIBUTIONS OF THE STUDY

A summary of the studies experimenting on the success of crowdsourced development is presented in Table 1. For instance, study [18] investigates if ''Quality'' is a parameter that affects ''why crowdsourcing is performed'' and ''how the contest shall be setup''. The sign ''X'' corresponds to investigation performed by the current study. Based on Table 1, and the aforementioned discussion, we can observe that the majority of these studies focus on contestants' behavior and quality assessment. In this study we go beyond current literature, since this study:

- ***Focuses on management issues that can be controlled and monitored early***. In particular, we focus on aspects that are formulated early while designing and setting up the crowdsourced activity. Special emphasis is placed on factors that can affect the success of the CSE operation.
- ***Provides a model that can aid contest providers*** on deciding upon important aspects of challenges' settings, based on the specific goal of crowdsourcing.
- ***Investigates duration and cost*** of crowdsourcing challenges, as factors for answering all 5W+1H management questions.
- Examines the ***where to crowdsource question***, from the perspective of the ***application domain***.
- Examines the ***relation between the time period*** when a contest is announced, ***to the potential success*** in terms of cost, duration, solution acquisition and quality.

TABLE 1. Case studies on CSE challenges.

Success Factors					
Cost	Duration	Quality	Crowd		
x	x	x, [19], [37]	x, [8], [25], [37]	Why?	Management questions
x	x	x	x, [6]	What?	
x	x	x, [2]	x, [1], [6]	Who?	
x	x	x	x	When?	
x	x	x	x	Where?	
x	x	x, [2], [19]	x, [39]	How?	

III. STUDY DESIGN

In this section we present the protocol of our case study that has been designed according to the guidelines of Runeson *et al.* [23]. The main goal of the case study is to provide empirical evidence on the success potentials of crowdsourcing software engineering tasks. The factors of interest include the parameters that the contest provider should configure, while setting up a new contest, whereas a CSE project is considered successful if it is able: (a) to acquire a winning solution, (b) to retain the cost under acceptable thresholds, (c) to maximize quality, and (d) to minimize the duration of the contest. To achieve this goal, we conducted a case study on the 2,209 contests in the TopCoder platform performed during 2018. The reason for conducting a case study is that our goal is to investigate the phenomenon of crowdsourcing in its real context. The cases and units of analysis in this study are presented in Section III-B.

A. OBJECTIVES AND RESEARCH QUESTIONS

The overall goal of this case study, formulated according to the ***Goal-Question-Metric*** approach [3] is to *analyze data from crowdsourced software development*; **for the purpose of***overcoming software management challenges*; **with respect to** *achieving contest success in terms of solution acquisition, cost, quality, duration;* **from the viewpoint of** *the contest provider*; **in the context of** *applying the crowdsourcing software development approach in design, development, quality assurance and cognitive tasks.* In order to achieve the aforementioned goal, we set six research questions driven by the 5W+1H model defined in Section I:

[RQ1] Why should a contest provider setup a competition?

RQ1 focuses on the goals of crowdsourcing. Based on the literature software development crowdsourcing, apart from the delivery of the product per se, provides additional benefits, e.g., decreased development cost, decreased development time and increased quality. Therefore, in this research question we investigate why contest providers' crowdsource with respect to the success indicators that are: delivery of a solution, reduced costs, time efficiency, quality. We answer this RQ by investigating the values of the aforementioned success indicators per type of software engineering challenge.

[RQ2] What types of tasks should be crowdsourced?

RQ2 aims to investigate the types of crowdsourced software engineering tasks that are the most common in the Top-Coder platform and the level to which these tasks achieve the crowdsourcing goals regarding solution delivery, reduced costs, time efficiency, quality and Overall Success. In this RQ we differentiate from RQ1 by examining whether the performance of each contest, with respect to the four success indicators, is within the success thresholds defined in Section III-D. Being aware of this information the contest provider can have access to the accumulated experience derived from the TopCoder community regarding the specific types of tasks.

[RQ3] Where can the crowdsourced projects be exploited?

RQ3 digs further into the findings of RQ1 by placing emphasis in the application domain (e.g., business applications, scientific applications, etc.). The obtained benefit from answering this research question is the same as in RQ1, we note that the extraction of the application domain cannot be obtained automatically from the TopCoder API, but was extracted manually.

[RQ4] Who is going to participate and win the competition?

RQ4 focuses on investigating the profile of the competition participants along with the winner(s) profile. In particular, we obtain information on the experience of the participants/ winners and their reliability. This information can be of paramount importance for contest providers: e.g., if the nature of the contest is relevant only for highly experienced developers, demanding high prize rates, then the configuration of the contest should not be of a low prize.

[RQ5] When is the right time to crowdsource a project?

RQ5 attempts to investigate trends on which periods of the year are the most active for competitions in terms of opening new contests, submitting solutions, receiving successful submissions, etc. The answer to this research question can guide the contest provider on which month the contest should be set, so as to achieve maximized success opportunities.

[RQ6] How do the different crowdsourcing success factors and indicators vary across the different types of software development activities crowdsourced?

RQ6 focuses on exploring how contest configuration factors (i.e., contest type, cost, duration, and number of winners) and project factors (i.e., application domain, challenge type) can affect the success of the contest. We note that a contest is considered successful not only if the end product is acquired, but also if the solution is cost- and time-efficient, and of high quality. The answer to this research question can be very useful for contest providers, in the sense that they will gain awareness on how to improve the success rates for their products.

B. CASE SELECTION

This paper reports a holistic multiple-case study since cases match the units of analysis. In particular, the ***context*** of the

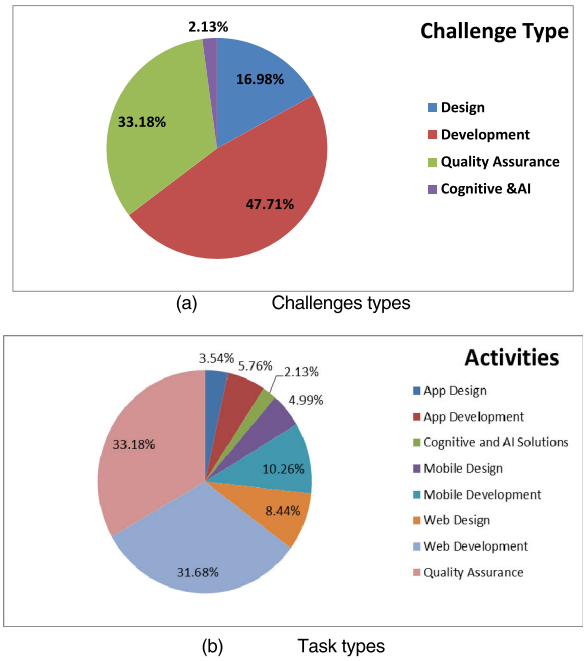

FIGURE 2. Types of challenges and tasks crowdsourced during 2018.

case study is crowdsourced software development, whereas as ***cases*** and ***units of analysis*** we consider crowdsourced competitions hosted by the TopCoder platform. TopCoder platform is a pioneer for practicing Crowdsourced Software Engineering (CSE), systematically used for empirical research the last years [2], [39], [35].

The platform supports four types of software challenges: Design, Development, and Quality Assurance and Cognitive & AI challenges. Each challenge hosted by TopCoder platform belongs to the above four types and is performed through an online competition where crowd developers compete with each other. Qualified winning solutions, i.e., those that receive the highest score through a review process, receive the award, which is usually a monetary prize. In the cases where the contest type is of type "First-to-Finish" the quickest contestant(s) with a qualified solution is (are) the one(s) that receive(s) the award. For the purposes of this study we selected all the contests that were announced and completed the last calendar year 2018, considering that this is a representative sample of crowdsourcing trends on software engineering. Fig.2a presents the distribution of challenges across the four different types. According to TopCoder classification each challenge can be further classified into eight specific types of tasks. The different types of tasks crowdsourced along with the frequency of each type are depicted in Fig.2b.

C. DATA COLLECTION

For every competition performed during 2018 on TopCoder platform we recorded or calculated the variables presented in Table 2. The majority of the variables are directly extracted from Topcoder platform API. In the case where further calculation is required the method to retrieve the variables cores is presented in the 3rd column of Table 2. In total 26 variables are recorded for each competition. The anonymized data set is publicly available.[3]

In Fig. 3 we present an example of a contest crowd sourced in TopCoder platform. The metrics that are directly derived from the description provided in the platform, are presented with a green font. The metrics that are derived from the values of the rest of the variables, or through text mining, are highlighted with red font. The calculation process of success indicator metrics (*V24- V28*) is presented in section III-D. The tool used to collect data based on TopCoder API is hosted in Github.[4] Specifically the location metric of the contest is indirectly derived by retrieving the main country where the provider company is situated, while the location of the contest winner (*V5*) is provided directly by TopCoder platform by parsing the page of the particular participant. Fig. 4 provides an overall view of the data collection and analysis process followed to serve the goal of this study.

D. DATA ANALYSIS

In this section we present the data analysis performed to define the success thresholds for each indicator (solution acquisition, cost, duration, and quality) and answer the six research questions defined in section III-A. To reach a conclusion whether the crowdsourced software engineering challenges can be considered successful we followed the steps described in Fig.4:

Step 1: To define success in terms of solution acquisition (Solution Acquisition$_{\text{binary}}$, V24) we examined the value of the Number of Winning Solutions (V21) variable and compared it to the value of Number of Winners (V12). If the value of variable V21 is smaller than the value of variable V12, then the contest is considered unsuccessful since it has not received the desired number of winning solutions. A failure to this criterion might be either due to lack of sufficient participation, or complete lack of participation.

Step 2: To define the success thresholds for the cost, duration and quality indicators we initially classified all challenges based on the value of variable V2 (Challenge Type) into the following classes: Design, Development, Quality Assurance and Cognitive and AI challenges. We selected to classify the challenges into groups so as to ensure data homogeneity within classes that will help us reach representative conclusions within each type of challenge. Then for each class of challenges we developed the boxplots for the four metrics related to the three success indicators, Prize (V10), Duration (V11), and Quality (V13) and examined the corresponding boxplot diagrams for each indicator (see Fig.5). The graphical representation of boxplots helped us identify outliers and extreme outliers for each indicator. Next, as a threshold for each success indicator, we have set the 75% quartile depicted in the boxplots. The quartile values helped us define the upper values (Prize, Duration) or the lower

[3] https://users.uowm.gr/sbibi/topcoder.xls
[4] http://github.com/zozas/topcoder

TABLE 2. Success factors and metrics.

	Variable	Metric	Values/ Calculation Method
Task Decomposition	V1	Competition Title	The title of the challenge
	V2	Challenge Type	The type of challenge as assigned by the provider: *Design, Development, Quality Assurance, Quality & AI solutions*
	V3	Type of Task	The specific type of task crowdsourced as assigned by the provider: *Application Design, Mobile Design, Web Design, Application Development, Mobile Development, Web Development, Quality Assurance, Cognitive& AI solutions*
	V4	Application Domain	The application domain of the crowdsourced challenge: *Commerce, Social Media, Entertainment, Finance, Industry, Media, Services , Other*
	V5	Contest Location	The location of the contest provider, i.e the country where the company is situated.
Project	V6	Type of Contest	Refers to the type of contest performed in order to support a challenge. *Round Match (RM) contest*: a deadline competition that has a pre-defined end date, after which no other submissions are allowed. Some of the RM competitions have two submission rounds, during the first round all submissions are tested according to several checkpoints and only the ones that meet the criteria pass to the second and final round. *First-to-Finish (F2F)* contests: submissions are reviewed when they arrive and the first(s) one that fulfill(s) quality requirements win(s) *Marathon Match* (MM) contest: is a competition that runs for an extended period of time and competitors tackle algorithmic challenges both real world and theoretical ones.
	V7	Start Date	Start date of the competition
	V8	End Date	End date of the competition
	V9	Date of the submitted winning solution.	This variable is recorded only for First-to-Finish contests and refers to the date in which the winning solution was submitted.
	V10	Prize	The monetary reward offered to the winning solution. In some cases, more than one winning solutions are accepted. In that case the total amount of the winning prizes is recorded.
	V11	Duration	This variable refers to the duration of the contest: For RMs and MM contests *duration= V7-V6* For F2F contests *duration = V8-V6*
	V12	Nof. Winners	The number of winners as defined by the contest provider
	V13	Quality	The quality score that winning solution received
	V14	Participants name	The nick names of the registrants to the particular challenge
Crowd	V15	No. of registrants	The number of the registered participants to the specific challenge
	V16	No. of submissions	The number of solutions submitted to the specific challenge
	V17	Participant rating	The rating of the participant, as assessed by the Topcoder, is based on the *Elo*[5] *rating system [9]* which is a method for calculating the relative skill levels of players in competitor-versus-competitor games.
	V18	Quit rate	Represent the quittance rate of the participants of a contest. It is calculated as following: *(V14-V15)/ V14.*
	V19	Avg. Rating	The average Participant Rating (V17) of all registered competitors to the contest.
	V20	Avg. Winner rating	Average winner rate of all the winners of a particular type of task.
	V21	Nof Winning solutions	The number of winning solutions for the particular contest
	V22	Avg. Nof Submissions	Average number of submissions for a particular type of task
	V23	Winner's Location	The location (country) where the winner is situated
	V24	Solution Acquisition$_{binary}$	This variable is binary and refers to whether the particular contest acquired the desired number of winning solutions. 0- when there was no sufficient number of winning solutions 1- when there was sufficient number of winning solutions
	V25	Prize Success$_{binary}$	This variable is binary and refers to whether the particular contest is within cost thresholds (see Section III-D). 0- when the contest is above the related thresholds 1- when the contest is below the related thresholds

TABLE 2. *(Continued.)* **Success factors and metrics.**

Variable		Metric	Values/ Calculation Method
Success Indicators	V26	Duration $Success_{binary}$	This variable is binary and refers to whether the particular contest is within duration thresholds (see Section III-D). 0- when the contest is above the related thresholds 1- when the contest is below the related thresholds
	V27	Quality $Success_{binary}$	This variable is binary and refers to whether the particular contest is within quality thresholds (see Section III-D). 0- when the contest is above the related thresholds 1- when the contest is below the related thresholds
	V28	Overall $Success_{binary}$	This variable is binary and refers to whether the particular contest has achieved success with respect to V24-V27 indicators. 0- when at least one of the four indicators has a zero value 1- when all of the four indicators have a value of one

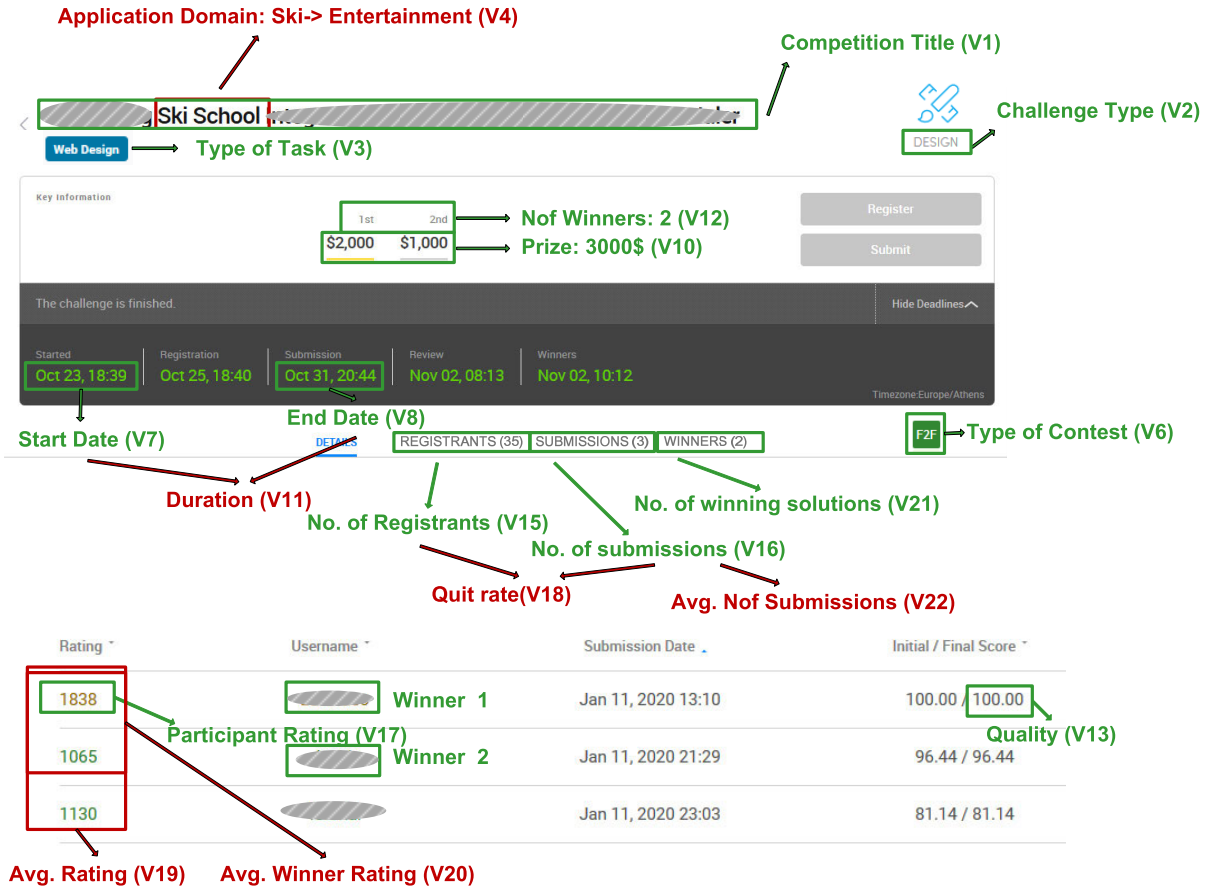

FIGURE 3. Contest information representation in TopCoder platform.

values (Quality) of the corresponding metric beyond/ above which any challenge can be considered successful.

The final list of threshold values for each success indicator for each challenge type is presented in Table 3. Each row of the table corresponds to a specific challenge type, whereas the columns correspond to the success indicators. The cells present the conditions under which a contest is considered successful: e.g., a design contest that ended up with a cost lower to 2.600 is considered successful. We note that the use of different thresholds for different types of challenges is necessary since the level task granularity and difficulty varies across them. Therefore, a Development project would be characterized as "*Successful*" if it lasts 6 days, but a project with exactly the same duration is considered as "*Failed*" if it involves Quality Assurance tasks.

Step 3: As a next step we created four new binary variables, Prize $Success_{binary}$ (V25), Duration $Success_{binary}$ (V26), Quality $Success_{binary}$ (V27) receiving the value 0 if

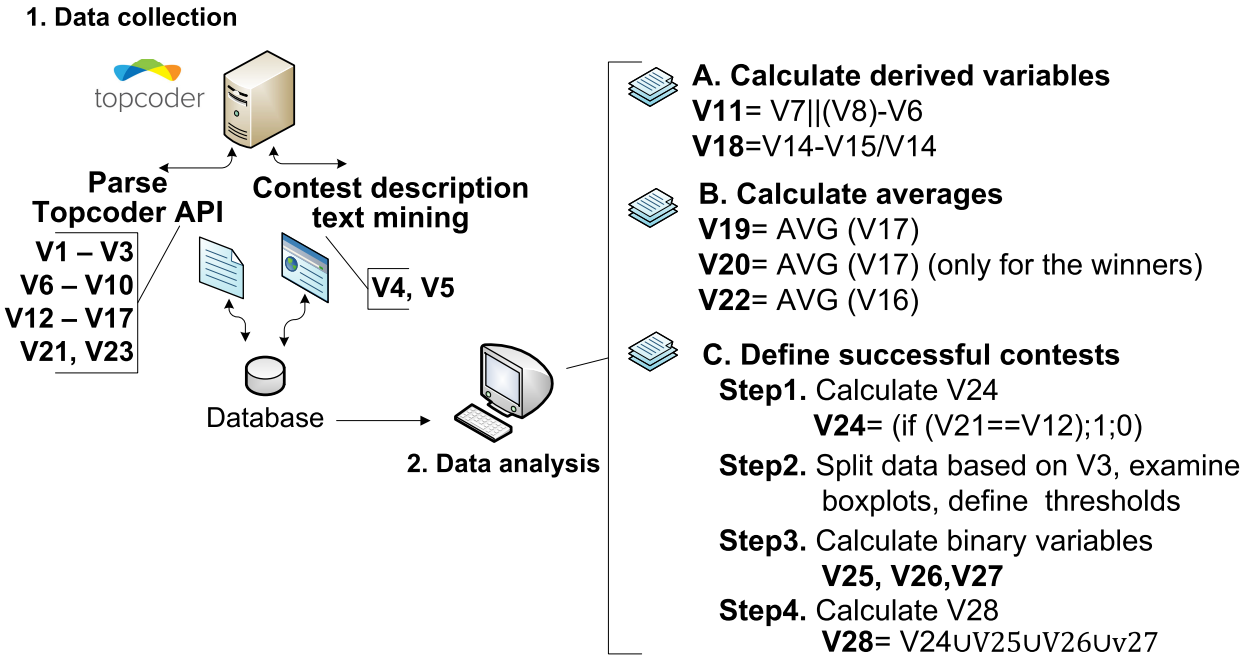

FIGURE 4. The data analysis procedure followed in this study.

TABLE 3. Success thresholds per challenge type.

Challenge Type	Prize	Winners	Duration	Quality
Design	<2600	>0	<14	>90
Development	<2000	>0	<8	>88
Cognitive and AI Solutions	<18000	>0	<15	>90
Quality Assurance	<1200	>0	<5	>93

the corresponding metric fails the threshold of Table 3, or 1 if the metric is within the expected threshold.

Step 4: Finally we calculated the value of Overall $\text{Success}_{\text{binary}}$ (V28) metric which is an aggregated metric that depicts whether a challenge is within the expected thresholds defined for all of 4 success indicators. The value of Overall $\text{Success}_{\text{binary}}$ metric is calculated as a logical function depicted below. Therefore a successful competition is expected to have price and duration lower than the threshold, quality higher than the threshold, and number of winners the same as the one defined by the provider. We note that we preferred the calculation based on binary values (rather than the actual values) in order to avoid mathematical inconsistencies in terms of units of measurement (e.g., dollars, days, etc.).

$$\text{Overall Success}_{\text{binary}} = \text{Solution Acquisition}_{\text{binary}} \cup \text{Prize Success}_{\text{binary}} \cup \text{Duration Success}_{\text{binary}} \cup \text{Quality Success}_{\text{binary}}$$

In order to cover the research questions described in section III-A, we performed several types of analysis as presented in Table 4.

For RQ1 we calculated the descriptive statistics (Min, Max, Mean and St. Error of Mean) of the target variables with respect to different types of tasks crowdsourced. By acknowledging the fact that the extracted data can only act as a proxy for the reasons for crowdsourcing, we have triangulated the findings through a focus group. A focus group is a "*research technique that collects data through group interaction on a topic determined by the researcher*" [15]. The focus group was structured according to the guidelines provided by Kontio *et al.* [15]. During the planning of the focus groups we defined the goal, which was to investigate the reason for crowdsourcing software tasks. Regarding design, the focus group lasted for one hour, whereas as participants we opted for selecting stakeholders with different roles in the software industry. The participants have been retrieved from industrial partners of two research projects consortia, which are spread across EU, and vary in terms of application domains. In total, 10 people from 5 industries participated in the focus group. The software development industries serve the following domains: Healthcare, Telecommunications, Entertainment, Media and Finance. All participants have either crowdsourced a software development task or are in the process of crowdsourcing one. The discussion was focused on the following topics:

- Describe the last crowdsourcing tasks
- Why have you decided to crowdsource?
- What is the main motivation behind the decision to crowdsource software development tasks?

For RQ2 to RQ5 we provide the heatmaps, by cross-tabulating the target and the grouping variables, to visualize the deviation of the expected against the observed values.

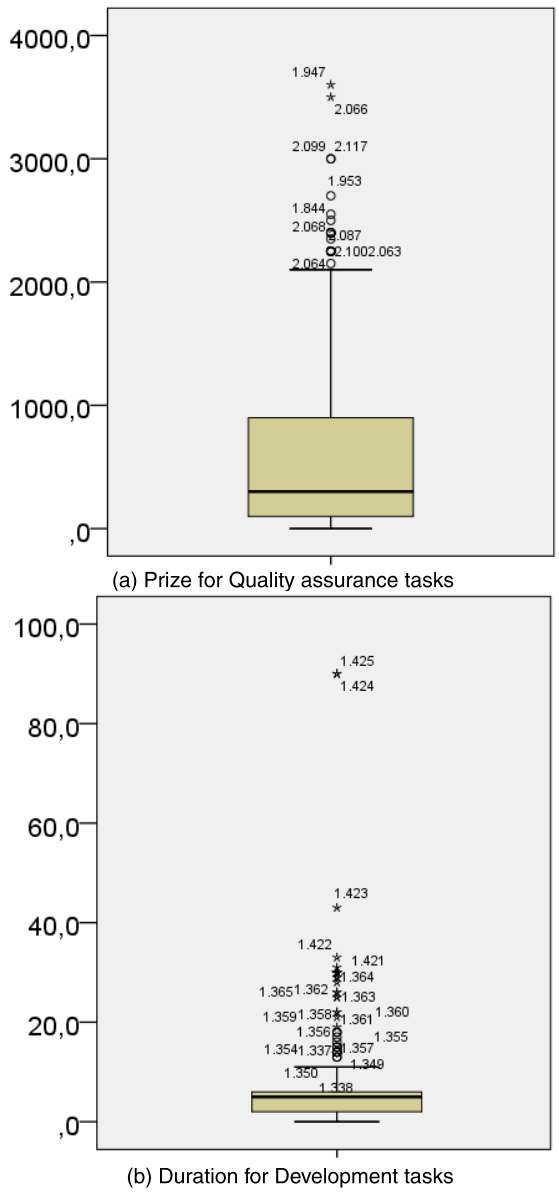

FIGURE 5. Boxplots for Prize and Duration Success Indicators for (a) Quality assurance tasks (b) Development tasks.

In these questions we also employ the Pearson chi-square test to evaluate whether the difference between the different groups of challenges (classified each time by the different grouping variable) are statistically significant. Finally, for RQ6 we adopt the representation formalism of decision trees to investigate how the different parameters for setting up crowdsourcing challenges vary across the different types of software development activities. In particular, we present tree-like models of decisions related to setting-up a new contest, depicting in each branch the average values/percentages observed for the participating variables in the particular group of challenges.

TABLE 4. Data analysis methods.

RQ	Testing Variables	Analysis
RQ_1	**Target variables:** • Prize (V10) • Duration (V11) • Quality (V13) • No. of Winning Solutions (V21) **Grouping variable**: • Type of Task (V3)	- Barcharts
RQ_2	**Target variables:** • Solution Acquisition (V24) • Prize Success (V25) • Duration Success (V26) • Quality Success(V25) • Overall Success (V28) **Grouping variable**: • Type of Task (V3)	- Heatmap - Pearson x^2
RQ_3	**Target variables:** • Solution Acquisition (V24) • Prize Success (V25) • Duration Success (V26) • Quality Success(V27) • Overall Success (V28) **Grouping variable**: • Application Domain (V4)	- Bar chart - Heatmap - Pearson x^2
RQ_4	**Target variables:** • Avg. Winner Rating (V20) • Avg. Rating (V19) • Avg. Number of Subm.(V22) • Quittance Rate (V18) • No. of Winning Sol.(V21) **Grouping variable**: • Type of Task (V3)	- Descriptive Statistics - Pearson x^2
RQ_5	**Target variables:** • Solution Acquisition (V24) • Prize Success (V25) • Duration Success (V26) • Quality Success(V27) • Overall Success (V28) • Prize (V10) • Duration (V11) • No. of. Winners (V12) • Type of Contest (V6) **Grouping variable**: • Start Month (V7)	- Bar chart - Heatmap - Logistic Regression
RQ_6	**Target variables:** • No. of Submissions (V16) • No. of Registrants (V15) • No. of Winners (V21) • Solution Acquisition (V24) • Duration (V11) • Duration Success (V26) • Quality (V13 • Quality Success (V27) • Prize (V10) • Prize Success (V25) • Overall Success (V28) **Grouping Variables**: • Type of Challenge (V2) • Application Domain (V4) • Type of Contest (V6)	- Decision Trees

IV. RESULTS

In this section, we present the findings of this case study, organized by research question. We note that in this section, no interpretation of results is performed, as all results are collectively discussed in section V. In Table 5 we present the descriptive statistics of the different types of tasks crowdsourced with respect to the four success indicators. The rows of the table represent the different application tasks that are crowd sourced grouped into eight broad categories (*Application Design, Application Development, Cognitive & AI, Mobile Design, Mobile Development, Quality Assurance, Web Design and Web Development*) and the columns represent the mean, minimum and maximum values of the variables *Prize, Solution Provision, Duration and Quality*. Additionally, the mean values of the application tasks that present the best performance for a specific success indicator are highlighted in bold.

TABLE 5. Descriptive statistics per application task.

Criterion	Task	Min	Max	Mean	Std. Dev
Duration	**Quality Assurance**	**0**	**30**	**4,51**	**6,67**
	Web Development	0	90	5,64	8
	App Development	0	30	6,45	6,97
	Mobile Development	0	33	6,82	6,87
	Web Design	0	20	8,63	4,7
	Mobile Design	0	38	9,71	5,42
	App Design	0	23	10,1	5,42
	Cognitive & AI	1	28	11,98	6,17
Prize	**Quality Assurance**	**0**	**3600**	**637,64**	**676,94**
	Web Development	0	4100	1018,7	874,38
	App Development	0	5800	1281,9	998,82
	Mobile Development	0	3600	1285,2	976,7
	Web Design	0	5750	1717,4	1069,27
	Mobile Design	1	5750	1989,6	1016
	App Design	1	6250	2112,2	1389,9
	Cognitive & AI	0	86000	9831,4	14296,3
Quality	**Web Design**	**91,4**	**100**	**98,17**	**2,84**
	Quality Assurance	40	100	97,75	4,54
	Mobile Design	93,3	100	97,42	2,44
	Cognitive & AI	81,8	100	96	4,54
	Web Development	75,4	100	95,92	5,29
	App Design	85,6	100	95,16	6,77
	App Development	80	100	95,01	5,86
	Mobile Development	75,3	100	94,92	6,37
Solution	**App Design**	**0**	**8**	**2,93**	**1,59**
	Mobile Design	0	8	2,71	1,23
	Web Design	0	8	2,37	1,23
	App Development	0	8	1,44	1,18
	Web Development	0	5	1,22	0,7
	Mobile Development	0	3	1,17	0,66
	Quality Assurance	0	5	1,15	0,66
	Cognitive & AI	0	5	1	1,41

[RQ1] Why should a contest provider setup a competition?

In this section we discuss the motivation behind the decision to crowdsource a software project. To achieve this goal, we first present a qualitative evaluation, by examining the level to which the four key success indicators are achieved in TopCoder contests. In Fig. 6 we present the percentages of projects that are considered to be successful with respect to the variables *Prize, Solution Provision, Duration, Quality* and *Overall Success*.

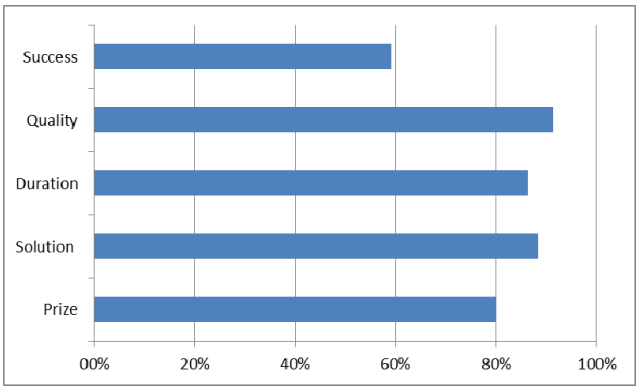

FIGURE 6. Percentages of successful contests for each indicator.

We observe that *Quality*, as a success indicator, is achieved in most of the cases, as 91.3% of the examined contests are successful in terms of assigned quality—i.e, the acquired solutions are within an acceptable quality. Given the relatively high percentage of success in this criterion, it is suggested that managers who are interested in the quality of the end product, have a high chance to achieve it, by crowdsourcing tasks.

In the majority of cases (88.2%) the contests are considered successful in terms of *Solution Provision*. Despite the fact that the analyzed contests (in many cases) require advanced software engineering skills (e.g., algorithmic, design, object-oriented, etc.) there is a large pool of contest participants that are able to provide an acceptable solution. Therefore, a provider can rely on crowdsourcing "*challenging*" or "*specialized*" tasks, since the community is expected to provide a solution.

In terms of *Duration*, 86.3% of the contests are considered to be within the acceptable thresholds. Although this criterion ranks 3rd in this first-level analysis, it is important to consider that for cases in which a solution was provided, 97% of them were delivered timely. Therefore, based on the quantitively analysis, time is a criterion that is almost always satisfied by the community, promoting it as an important reason to crowdsource.

In terms of *Cost*, 80.1% of the contests are considered to be within the acceptable thresholds. Despite the fact that the percentage is quite high, we need to note that in relative terms, this criterion is the one that most frequently fails. Therefore, it is implied that in some cases contest providers might overestimate the effort required to complete a task, leading them to overpay for its completion.

Overall, we observe that only 59.1% of contests are considered successful by taking into account all the aforementioned success indicators. The substantially lower percentage of the overall success, compared to the individual factors, suggest that while crowdsourcing it is not realistic to expect a maximization of all factors, but there are trade-off between them.

As a next step, by acknowledging the fact that the above analysis can only proxy the answer to the ''why'' question, we have performed a qualitative analysis through a focus group. During the discussions in the focus group, four reasons for motivating the decision to crowdsource software development tasks have emerged: (a) the opportunity to quickly exploit skills that are not available in the immediate environment of the company, (b) the access to diverse solutions to a single problem, (c) the shrinking of time to acquire a solution, (d) the need to minimize the risk of failure. The first reason to crowdsource was mentioned by all five participants as being the most important. According to software managers there are certain software development tasks that require specialized skills, which are not easy to find. As mentioned by PM1: ''*we were in a rush to migrate certain functionalities to the cloud and we needed a way to eliminate data queries so as to reduce costs, we crowdsourced the problem and got an optimized solution for queries*''. The second and third most important drivers for crowdsourcing is the access to diverse solutions as quickly as possible: ''*we wanted to have access to a diversity of solutions for the logo and interface for our new patient teleconsultation app, therefore we decided to crowdsource the task*''. Concerning the last motivating factor for crowdsourcing, i.e., the minimization of the risk of failure, it was mentioned that ''*In the past there were cases in which we either devoted internal resources, or outsourced tasks, to produce solutions for particular problems (i.e., platform design). In that cases even if we were dissatisfied with the result, we still needed to pay internal or external resources. In the case of crowdsourcing this risk is minimized, as if we are not satisfied, we will not pay.*''

[RQ2] What types of tasks should be crowdsourced?

In this section, we assess the performance of the most common types of application tasks crowdsourced with respect to the four success indicators. Our goal is to explore the success potentials of each type of task. To achieve this goal, we compare the actual value of a success indicator to the thresholds defined in section III-D. If the value of the particular success indicator is below/beyond the defined threshold then the contest is considered to be a successful with respect the particular success indicator. The heatmap of Table 6 presents the different success indicators for each type of task. In the heatmap, the rows correspond to the eight different types of tasks, and the columns to the different success indicators along with the accumulated Success metric. The colour of each cell suggests if the percentage of projects that succeed specific success thresholds is higher (blue) or lower (red) than the average one for the specific indicator. The percentage that appears inside the cell corresponds to the difference between the percentage of successful projects implementing the specific task, and the total percentage of successful projects, with respect to the success indicator considered.

Among the most successful tasks in terms of ***Cost*** are the *Web Design* tasks, presenting a great difference compared to the rest types of tasks, while the *Application development* tasks present a significant percentage of projects under cost

TABLE 6. Successful CSE indicators per application task.

	Prize	Winner	Duration	Quality	Success
App Design	0.7%	8.0%	-3.0%	-13.0%	3.7%
App Dev.	-7.7%	-1.6%	1.1%	-3.6%	-5.6%
Cognitive &AI	-3.5%	7.5%	-7.6%	6.4%	6.9%
Mobile Design	-0.1%	10.0%	6.4%	12.0%	11.8%
Mobile Dev.	-13.7%	-2.4%	0.0%	-2.4%	-13.5%
Quality Ass.	1.0%	-0.6%	-5.5%	-0.7%	-2.3%
Web Design	9.1%	2.7%	5.6%	12.0%	14.6%
Web Dev.	2.4%	-1.8%	3.8%	1.6%	1.4%

TABLE 7. Relation of success indicators and CSE tasks.

	Task		Sub-Task	
	x^2	Sig.	x^2	Sig.
Prize	44,55	*0,00*	71,09	*0,00*
Winners	23,18	*0,002*	25,21	*0,02*
Duration	38,78	*0,00*	50,58	*0,00*
Quality Assurance	49,00	*0,00*	54,30	*0,00*
Success	44,92	*0,00*	70,36	*0,00*

thresholds. We observe that in terms of ***Solution Provision*** *Application Design* and *Mobile Design* tasks along with *Cognitive& AI* tasks present high percentages of solution acquisition while *Mobile and App Development* tasks are the ones that present the higher probability of not being able to acquire a solution. With respect to ***Duration*** and time-efficiency, *Mobile* and *Web Design* tasks are the most successful in terms of time, with *Cognitive& AI solutions* often exceeding time thresholds. The highest percentages of successful projects in terms of ***Quality*** are observed in *Mobile* and *Web Design* tasks while the lowest percentages are observed in *Application Design* tasks. Overall, with respect to the total ***Success*** indicator, we observe that *Web Design* and *Mobile Design* tasks are the ones that are most probable to achieve good performance with respect to the four success indicators. *Mobile Development* and *Application Development* tasks on the contrary present the highest percentages of contests not achieving the success thresholds for the four indicators. The results of the Chi-square test, between the success indicators and the type of task crowdsourced are presented in Table 7.

The second column of the table presents the p-value that indicates whether these variables are independent of each other. The results show that the success indicators are not independent to the type of task crowdsourced and that there is a statistically significant relationship between these two variables. The Pearson x^2 results are highlighted with italic font when the findings are statistically significant.

[RQ3] Where can the crowdsourced projects be exploited?

In this research question we emphasize on the application domain where the crowdsourced software development tasks are exploited. In order to derive the Application Domain where the result of a contest will be exploited the second author parsed lexically the description of the contest in order to isolate the words that indicate the type of domain. Initially he used the exact term of the domain as mentioned in the contest description. As a second step he grouped domains that belong to the same broad category. In Fig.7 we can see the

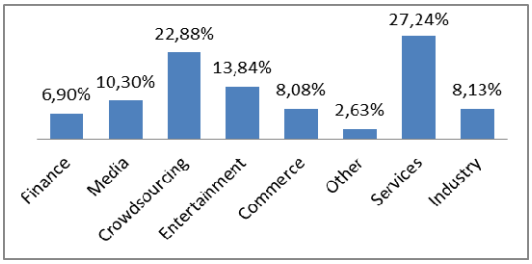

FIGURE 7. Application Domains of CSE.

different types of application domains where CSE is applied. The majority of tasks are deployed in the *Services domain* (Drone Services, Analytics, and Communications) and the *Social Media Business* (Social Networks, Crowdsourcing). Tasks for the *Entertainment Business* (Religion, Sports, and Travelling) and *Media Business* are also very often crowdsourced. In the category ''Other'' we find *Science, and Programming* application domains.

The heatmap of Table 8 presents the different success indicators for each application domain. The interpretation of the heatmap is performed similarly to RQ2:

- We observe that in terms of ***Solution Provision*** the tasks destined for *Social Media and Entertainment* sector present high percentages of solution acquisition while the tasks launched by the *Finance* sector are the ones that present the higher probability of not being able to acquire a solution.
- In terms of ***Cost*** the most cost-effective tasks are under the category *Social Media and Other*. Also we observe that in the *Finance, Commerce and Industry* application domain the percentages of cost success are increased compared to the expected costs.
- With respect to ***Time-efficiency***, tasks crowdsourced for the *Finance* and the *Other* application domains are the most successful in terms of time, while tasks for the *Social Media* application domain often exceed time thresholds.
- The highest percentages of successful projects in terms of ***Quality*** are observed in the application domain of *Commerce* while the lowest percentages are observed in the *Social Media* domain.
- Overall, with respect to the total ***Success indicator***, we observe that tasks crowdsourced for the *Social Media* industry are the ones that are most probable to achieve good performance with respect to the four success indicators. Tasks for the *Commerce Industry* on the contrary present the highest percentages of contests not achieving the success thresholds for the four indicators.

An analytic presentation of the subcategories of the application domains can be found in the Appendix where the corresponding percentages per sub-category are recorded.

The results of the Chi-square test, between the success indicators and the application domains are presented in Table 9. The Pearson x^2 results are highlighted with italic font when the findings are statistically significant. The results show

TABLE 8. Successful contests per application domain.

	Prize	Winners	Duration	Quality	Success
Commerce	-12.10%	-1.10%	1.10%	6.50%	-6.50%
Social Media	7.80%	4.90%	-4.90%	-2.60%	2.60%
Entertain.	-3.70%	2.00%	-2.00%	2.40%	-2.40%
Finance	-13.00%	-5.90%	5.90%	-0.50%	0.50%
Industry	-11.90%	-2.90%	2.90%	-0.90%	0.90%
Media	3.20%	-3.90%	3.90%	-0.50%	0.50%
Other	6.10%	-4.90%	4.90%	3.50%	-3.50%
Services	3.90%	-0.60%	0.60%	-0.70%	0.70%

TABLE 9. Relation of success indicators and application domain.

	Application Domain	
	x^2	Sig.
Prize	*78,81*	*0,00*
Winners	*24,19*	*0,001*
Duration	11,76	0,11
Quality Assurance	12,80	0,08
Success	*38,87*	*0,00*

that the success indicators related to Solution Acquisition and Cost Reduction presents a statistically significant relationship with the type of the application domain where the task will be exploited. Such a relationship is not verified for the Time-efficiency and the Quality indicators. Overall the total Success indicator also presents a statistically significant relationship with the type of the application domain.

[RQ4] Who is going to participate and win the competition?

In this section, we present the results concerning the participants' experience and reliability across different challenges and activities and provide insights on the winners' profile. Crowd experience is measured by the average participants rating that represents the average scores of all participants within a contest, the average winners rating that represents the average of all the winners score. Crowd reliability is measured as the average number of submissions delivered in a contest and the average quittance rate (AVG_QTR) in a contest that is the percentage of participants that registered in a contest but did not submit a solution. Table 10 presents the distribution of contests, per development tasks according to the different number of winners. In this table we do not use any colors. For example, in the first row we observe that 3.9% of the Application Design contests present no winner at all (therefore are considered as unsuccessful), while 6.4% have 1 winner. The second column that represents the unsuccessful contests (zero winners) reveals that Cognitive & AI contests present an increased possibility to not be able to receive a suitable submission. This percentage is increased compared to the rest of the development tasks crowdsourced. Application Design and Mobile Design contests are the most successful in terms of solution acquisition. Web Development, Quality Assurance, Mobile Development and Application Development usually provision 1 or 2 winners. Application Design, Mobile Design and Web Design usually provision 3 or 2 winners. The cases where more than three winners are provisioned are very rare.

TABLE 10. Number of winners per development task.

	0	1	2	3	4	5	6	7	8
App Design	3.90%	6.40%	23.10%	53.90%	2.60%	3.90%	0.00%	1.30%	5.10%
App Development	13.40%	44.10%	37.80%	1.60%	0.00%	0.00%	1.60%	0.00%	1.60%
Cognitive &AI Solutions	59.60%	4.30%	23.40%	6.40%	2.10%	4.30%	0.00%	0.00%	0.00%
Mobile Design	1.80%	10.00%	28.20%	50.00%	0.90%	7.30%	0.00%	0.00%	1.80%
Mobile Development	14.20%	54.90%	30.10%	0.90%	0.00%	0.00%	0.00%	0.00%	0.00%
Quality Assurance	12.50%	62.10%	23.80%	1.10%	0.40%	0.10%	0.00%	0.00%	0.00%
Web Design	9.10%	14.00%	20.40%	50.50%	0.50%	4.80%	0.00%	0.00%	0.50%
Web Development	13.60%	51.40%	34.00%	0.70%	0.10%	0.10%	0.00%	0.00%	0.00%

Table 11 presents the average winner rating, the average registrants rating, the average number of submissions and the average quittance rate for the eight types of development tasks that can be crowdsourced and the 5 success indicators. For example the first row of the table is interpreted as following: In the Application Design contests, the average winner rate in the case of failure, in terms of cost deviations (Prize success indicator), is 351.07 while in the case of success is 151.68, similarly stands for the average registrants rating variable. Also in the Application Design contests the average number of submissions in case of failure is 24.3 and in case of success 19.83, also the average Quit rate in case of failure is 63.2% and in case of success 74.2%. Some of these results can be interpreted intuitively:

- In terms of winners experience we note that the maximum average winner rating is observed in Web Development and Mobile Development challenges (∼2500) while the least experienced winners participate in Application Design and Mobile Design challenges. In most cases the average winner rating is increased in successful projects compared to unsuccessful ones apart from the case of cost indicator. In that case we observe that the average winner rating is decreased for most of the types of development tasks crowd sourced in the case of successful projects. This can make sense considering the fact that in challenges that offer higher prize than the expected ones concentrate the interest of experienced developers.
- In terms of registrants experience we note that the maximum average registrants rating is observed in Web Development and Application Development challenges (∼1300) while the least experienced registrants participate in Application Design and Mobile Design challenges. In the majority of cases the average registrants rating is increased (from 15% to 25% in most cases) in successful projects compared to unsuccessful ones.
- Regarding the number of submissions, we note that the maximum number is observed in Cognitive & AI challenges (∼30) followed by Application Design and Mobile Design challenges (∼8 submissions). Application, Mobile, Web development and Quality assurance challenges present the minimum number of submissions (∼3). The number of submissions remains stable in both successful and unsuccessful projects in most cases.
- Overall the maximum Quit rate is observed in Application Development and Mobile development challenges (∼85%), while Cognitive & AI challenges present the smallest Quit rates (∼45%). The rest types of challenges present a percentage around 75%. In most cases the Quit percentage is decreased in successful projects compared to unsuccessful apart from the case of cost indicator. Similarly, this makes sense considering the fact that in challenges that offer higher prize than the expected ones the participants are highly motivated and therefore are more persistent to submit a solution.

Additionally, in this RQ we have also examined the winners' location and whether there is a collocation with the project's location. In Fig.8 we can see the distribution of contests winners' locations, whereas the majority of the location for contest providers is United States. To statistically test the possibility of collocation between contest providers and winners, we have performed a chi-square test, by crosstabulating the variables. The results suggest that there is a relation between the two ($x^2 = 618$, df $= 490$, $p < 0.01$). However, this result shall be treated with caution, since the large majority of contest providers is concentrated in US.

[RQ5] When is the right time to crowdsource a project?

In this research question we examine the time trends regarding the periods within a calendar year where crowdsourcing activity is observed with respect to the success indicators under study. Our target is to reach conclusions regarding the months during which a new contest should be set, so as to achieve maximized results in terms of the four success indicators.

In Fig.9 we present the number of competitions announced during each month of the last year. Based on the results March, April and May are the months during which the highest numbers of new contests are launched. On the other hand February and September are among the most inactive months. The heatmap of Table 12 presents the different success indicators for the contests announced during each month of the last year. The interpretation of the heatmap is performed similarly to RQ2.

In terms of ***Cost*** we observe that contests announced during August, May and January are considered to be successful. Contrary to that the contests announced in November, December and February present higher costs than the expected. We observe that in terms of Solution Provision

TABLE 11. Crowd factor mean values per success indicator and different development tasks.

		Avg. winner rating		*Avg. registrants rating*		*Avg. nof submissions*		*Avg. Quit rate*	
		Failure	**Success**	**Failure**	**Success**	**Failure**	**Success**	**Failure**	**Success**
Prize	App Design	351.07	151.68	166.43	209.16	8.3	9.5	63.2%	74.2%
	App Development	2368.37	1917.16	334.84	377.52	2.8	5.6	93.3%	84.8%
	Cognitive and AI Solutions		2141.17		111.00	35	28.2	23.02%	57.32%
	Mobile Design	135.09	506.72	167.93	182.22	14.6	8.6	72.33%	75.3%
	Mobile Development	2576.90	2386.01	331.94	386.16	2.2	3.2	93.5%	86.9%
	Quality Assurance	2468.79	2146.03	349.63	383.21	3.1	3.3	91.97%	78.8%
	Web Design		241.39	161.42	208.82	14.9	6.9	66.11%	75.5%
	Web Development	3225.81	2583.13	328.97	377.91	3.1	3.5	92.27%	82.6%
Winners	App Design		197.63		194.90		11.5		71.1%
	App Development		2357.02	279.03	379.16	13.3	3.5	90.95%	86.5%
	Cognitive and AI Solutions		1712.93	163.25	181.55	6.1	30.9	81.3%	47.9%
	Mobile Design		440.40	178.93	179.37	4.2	9.9	91.1%	74.4%
	Mobile Development		2854.36	305.91	378.15	1.34	3.1	95.7%	88.1%
	Quality Assurance		2520.77	357.49	379.54	1.3	3.5	93.7%	79.6%
	Web Design		237.10	153.11	208.81	5.4	8.1	53.6%	44.7%
	Web Development		3120.12	315.47	377.85	3.1	3.5	88.5%	83.7%
Duration	App Design	273.31	173.37	152.24	210.68	10.1	11.3	81.2%	70.3%
	App Development	1576.50	2108.54	278.82	378.29	16.1	3.2	84.3%	87.5%
	Cognitive and AI Solutions	0.00	2083.30	10.60	105.14	28	30.3	29.6%	54.6%
	Mobile Design	322.88	440.98	184.98	178.92	6.1	10.1	85.5%	73.8%
	Mobile Development	1642.84	2578.55	312.12	376.80	3.4	2.8	87.2%	89.4%
	Quality Assurance	2539.16	2128.27	323.32	389.57	3.5	3.1	88.4%	79.6%
	Web Design	198.13	216.95	176.86	206.08	6.5	7.9	84.0%	73.7%
	Web Development	1583.68	2817.42	299.42	377.03	5.5	3.2	85.1%	84.2%
Quality	App Design	1068.00	587.33	406.57	360.92	1	11.2	95.7%	71.8%
	App Development	2449.94	2365.47	359.86	385.66	2.7	5.2	92.1%	86.3%
	Cognitive and AI Solutions	3773.00	4312.29	235.20	286.33	1	30.4	98.2%	48.2%
	Mobile Design		5542.67		302.60		9.8		74.7%
	Mobile Development	2631.14	2892.01	311.56	389.39	2.3	2.9	93.8%	88.4%
	Quality Assurance	2164.57	2572.38	386.57	378.53	2.5	3.3	91.6%	80.1%
	Web Design		1404.19		378.63		7.8		74.5%
	Web Development	2494.64	3187.19	348.58	381.25	2.2	3.6	92.9%	83.4%
Success	App Design	281.86	135.67	187.90	208.66	12.8	10.1	74,1%	71.1%
	App Development	1905.86	2159.21	325.90	400.33	6.2	3.7	90.5%	84.2%
	Cognitive and AI Solutions	235.81	2364.81	41.73	107.37	29.8	29.9	39.9%	54.1%
	Mobile Design	173.59	538.56	172.88	182.02	11.9	8.9	76.8%	73.8%
	Mobile Development	2114.42	2851.19	331.35	411.61	2.5	3.4	92.2%	85.5%
	Quality Assurance	1996.23	2367.43	348.98	397.94	2.9	3.5	89.5%	74.8%
	Web Design	60.65	270.79	164.00	217.93	9.8	7.1	56.6%	77.7%
	Web Development	1949.42	3183.39	327.47	396.76	3.3	3.5	89.3%	81.1%

the tasks announced in May, January and July present high percentages of solution acquisition. Tasks announced during November, December and February are the ones that present the lowest percentages of ***solution acquisition***. With respect to ***Time-efficiency***, tasks crowdsourced during May, August and June are the most time-efficient, while tasks announced during December, February and January often exceed time thresholds. The highest percentages of successful projects in terms of ***Quality*** are observed in February, May and August while the lowest percentages are observed during September, November and June. Overall, with respect to the total ***Success indicator***, we observe that tasks

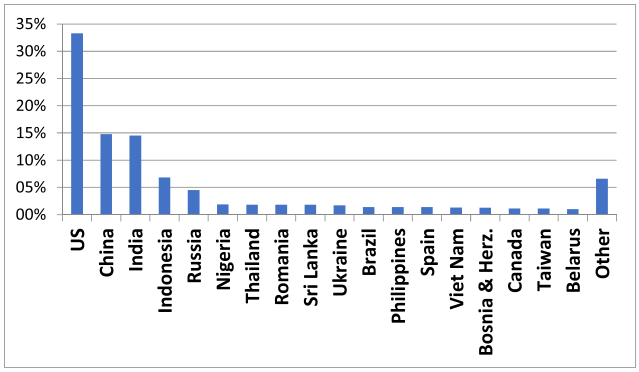

FIGURE 8. Winners' location.

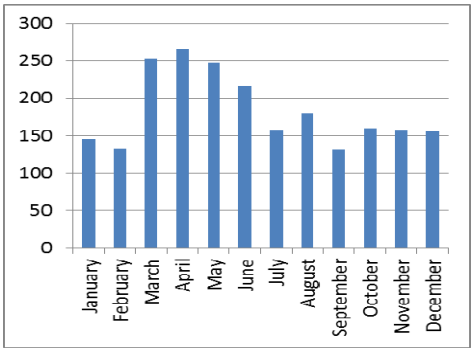

FIGURE 9. Number of contests per month.

crowdsourced during May and August are the ones that are most probable to achieve good performance with respect to the four success indicators. Tasks announced during December, February and November usually fail in achieving the success thresholds for the four indicators. Table 13 presents the results of Logistic Regression for estimating the extent to which the month when the contest is launched affects the success of the contests. The contest start month served as an independent variable while the contest attributes (Prize- V10, Duration –V11, Type of Contest –V3, Noof Winners- V12) as covariates. The results suggest that both covariates and indepe——— ndent variable are statistically significant. By comparing the importance of each month, we have observed that months January, May, and August are those with the highest probability to lead to successful contests (marked with italics in Table 13).

[RQ6] How do the different contest settings affect the success of the different types of software development activities crowdsourced?

In this research question we focus on exploring the level to which contest configuration factors (i.e., contest type, number of winners, cost, duration, and prize) and project factors (i.e., application domain, type of task crowd sourced) affect the success of the contest. To answer this research question we constructed four graphical models (see Fig. 10-13), one for each challenge type (Design, Development, Quality Assurance, Cognitive & AI solutions), that depict accumulatively the values of the contest configuration parameters

TABLE 12. Deviations form success indicators per month.

	Prize	Winners	Duration	Quality	Success
January	5,5%	5,6%	-13,0%	-0,8%	-5,0%
February	-5,1%	-3,4%	-15,1%	4,2%	-13,6%
March	-2,6%	-2,8%	-5,3%	-0,7%	-7,7%
April	1,9%	-0,2%	3,9%	-0,2%	2,9%
May	6,2%	7,3%	7,2%	4,0%	17,8%
June	-2,7%	-1,1%	5,4%	-2,8%	1,3%
July	-4,3%	4,8%	4,8%	-1,9%	4,6%
August	6,6%	1,8%	6,5%	7,0%	12,0%
September	0,8%	2,6%	1,5%	-5,9%	3,5%
October	1,8%	-1,3%	5,0%	-1,8%	0,3%
November	-4,8%	-11,0%	4,2%	-3,3%	-9,7%
December	-7,0%	-3,6%	-15,1%	-0,8%	-17,4%

TABLE 13. Variables and coefficients of logistic regression for success indicators.

	Prize B	Duration B	Quality B	Winner B	Success B
January	*.61*	.36	.04	*.99*	*.58*
February	.08	.04	.55	.02	.29
March	.04	.32	.26	.40	.06
April	.54	.33	.08	.01	.34
May	.45	*.85*	.51	.81	*.59*
June	.06	.55	.14	.20	.37
July	.07	.19	.05	.41	.40
August	*.77*	.26	*.99*	*.52*	*.65*
September	.17	.27	.29	.02	.46
October	.58	.59	.39	.03	.40
November	.12	.05	.08	.62	.11
December	Used as base of comparison				

and the project success factors. For each parameter used to quantify a success factor we provide its mean value (in a parenthesis we present the standard deviation), based on the contests that reside in the particular group, along with the percentage of projects that are considered successful in the particular context (in the parenthesis we present the mean values). Especially for the application domain and the contest type, which are the sole categorical variables, we provide the percentage of the challenges performed in each application domain/ type of contest.

For example, in Fig. 10 we observe that 15.2% of the Design challenges are performed for the Commerce application domain, 96.5% of which are crowdsourced as Round-Match contests. In that case the average number of submissions is 8.6, the average number of registrants is 41.9, the average number of winners is 4.3 (95% of contests acquired a solution), the average duration of the contests is 12.4 days (83% of contests are within time thresholds), the average cost is 2086$ (78% of contests are within cost thresholds) and the average quality reaches 100% (and all contests are considered successful in terms of quality). Overall 60% of contests are considered to be successful considering all success indicators (solution acquisition, time, cost, and quality).

Regarding **Design** challenges presented in Fig.10:

- **Application Domain:** Design challenges are very popular in the Services domain (Analytics, Communications,

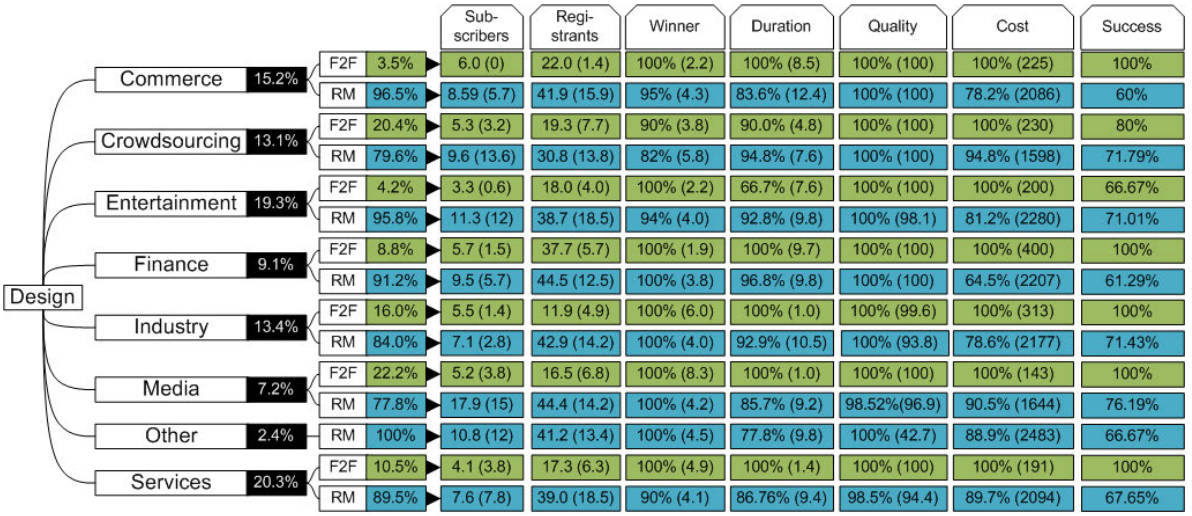

FIGURE 10. Design challenges success rates and contest settings.

IoT ser-vices), Entertainment domain and the Commerce domain.

- **Contest Type:** Design challenges are crowdsourced mainly as Single Round Matches (RMs) that are fixed duration con-tests.
- **Submission rate:** We observe that the number of submissions is increased (over 30% in most cases) in RMs compared to F2F. Also the number of submissions is comparable between different application domains (it ranges from 3.3 to 17.9 submissions).
- Participation: The overall participation remains increased in Single Round Matches (on average ∼10 registrations per contest) compared to F2F contests (∼5 registrations on average).
- **Winner:** Design challenges in the majority of cases acquire a winning solution. The average number of winners in most cases remains the same (it ranges from 2.8 to 8.5 winners).
- **Duration:** First-to-Finish contests in that case present greater success potentials in terms of duration. Overall the duration of Design challenges ranges from ∼7 to ∼12 days in RM contests, and from ∼2 to ∼5.5 days for F2F contests.
- **Quality:** Most of the Design challenges achieve 100% success in terms of quality.
- **Cost:** The cost of Design challenges ranges approx. from 250$ (for F2F challenges) to 2000$ (for RMs). F2F contests present increased possibility of being successful in terms of cost. This finding can be explained by the fact that F2F contests usually crowdsource tasks of limited scope compared to RMs and therefore the prizes provisioned are smaller.

Regarding **Development** challenges presented in Fig. 11:

- **Application Domain:** Development challenges are very popular in the Services domain (Analytics, Communications, IoT services), Crowdsourcing and Entertainment domain.
- **Contest Type:** It is a common practice to crowd-source Development problems as Single Round Matches (RMs) that are fixed duration contests. This fact though in the majority of cases leads to decreased success percentages with respect to all indicators.
- **Submission rate:** We observe that the number of submissions presents a very small difference between First to Finish and Single Round Match contests, with the first type of contest concentrating slightly increased number of submissions. Also the number of submissions is comparable between different application domains (it ranges from 2.7 to 4.8 submissions).
- **Participation:** The overall participation seems to be slightly increased in Single-Round-Matches that is opposed to the trend observed in the number of submissions. The participation does not present differences between different application domains (it ranges from 17 to 41.7 registrations).
- **Winner:** Development challenges present a relatively high possibility of acquiring a winning solution; the average number of winners in most cases remains the same.
- **Duration:** In terms of duration we see that as expected First-to-Finish contests present greater success potentials. Overall the duration of Development challenges ranges from ∼6 to ∼10 days in RM contests, and from ∼2 to ∼5.5 days for F2F contests.
- **Quality:** We observe that F2F challenges present greater quality percentages. This can be interpreted by the fact that quality demands in F2F challenges may be looser compared to RMs, emphasizing in time constraints.
- **Cost:** The cost of Development challenges is decreased compared to the other types of challenges, ranging approximately ∼500 $ for F2F challenges and 1200%

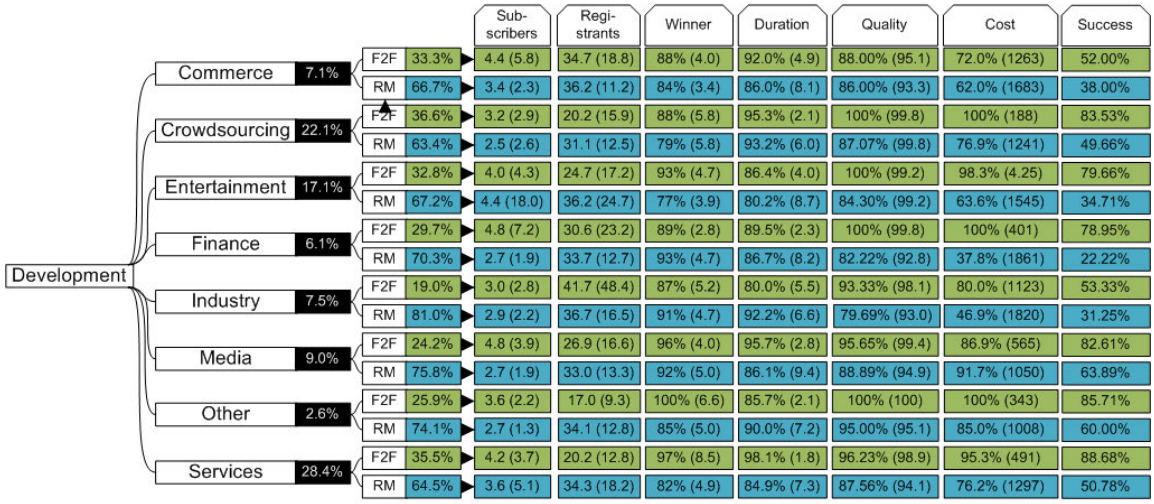

FIGURE 11. Development challenges success rates and contest settings.

for RMs. F2F contests present increased possibility of being successful in terms of cost. This finding can be explained by the fact that F2F contests usually crowd-source tasks of limited scope compared to RMs and therefore the prizes provisioned are smaller.

Regarding **Quality Assurance** challenges presented in Fig. 12:

- **Application Domain:** Quality assurance challenges are very popular in the Services domain (Analytics, Communications, IoT services), Crowdsourcing domain and Media domain.
- **Contest Type:** First-to-Finish contests and Single-Round-Matches are both popular in Quality Assurance challenges. In this case where the performance of the challenges can be directly compared we observe that in almost all success indi-cators, F2F contests present greater success percentages.
- **Submission rate:** We observe that the number of submissions presents a very small difference between First to Finish and Single Round Match contests. Also the number of submissions is comparable between different application domains (it ranges from 2.4 to 4.9 submissions).
- **Participation:** The overall participation seems to be slightly increased in Single Round Matches (∼28 registrations).
- **Winner:** Quality Assurance challenges present increased percentages of not acquiring a winning solution, compared to the other types of challenges. Challenges performed for the Crowdsourcing, Media and Industry domain in the form of F2F contests present higher possibility of not receiving a winning solution. The average number of winners in most cases remains the same.
- **Duration:** In terms of duration overall the duration of Quality Assurance challenges ranges from ∼4.7 to ∼11 days in RM contests, and from ∼1.2 to ∼3.4 days for F2F contests.
- **Quality:** In Quality Assurance challenges the quality ranges from ∼91.67% to 100%.
- **Cost:** The cost of Quality Assurance challenges is on average 700 $ for F2F challenges and 1050$ for RMs. F2F contests present increased possibility of being successful in terms of cost.

Regarding **Cognitive & AI**challenges presented in Fig.13:

- **Application Domain:** Cognitive & AI challenges are very popular in the Services domain, Crowdsourcing and Enter-tainment domain.
- **Contest Type:** It is a common practice to crowd-source Cognitive & AI challenges as Marathon Matches (MMs) that are long duration contests. Marathon Matches have longer duration compared to RMs, and therefore decreased percentages in terms of time success. Also MMs offer greater prizes; thus they concentrate more participations and submissions.
- **Submission rate:** We observe that the number of submissions is highly increased compared to the other types of chal-lenges, that is for RMs ∼10 submissions and for Marathon Matches ∼40 submissions.
- **Participation:** The overall participation on the other hand is increased in Single Round Matches (∼85) compared to Marathon Matches (∼60).
- **Winner:** Cognitive & AI challenges present high participation and a relatively high possibility of acquiring a winning solution. But in the cases of RMs in ∼50% of the challenges there is no winning solution.
- **Duration:** In terms of duration RM contests present greater success potentials. Overall the duration of Cognitive & AI challenges ranges from ∼10 to ∼18 days in MM contests, and from ∼5.5 to ∼12.5days for RM contests.
- **Quality:** The quality of Cognitive & AI challenges is most of the time 100%, therefore the success in terms of quality is usually achieved.

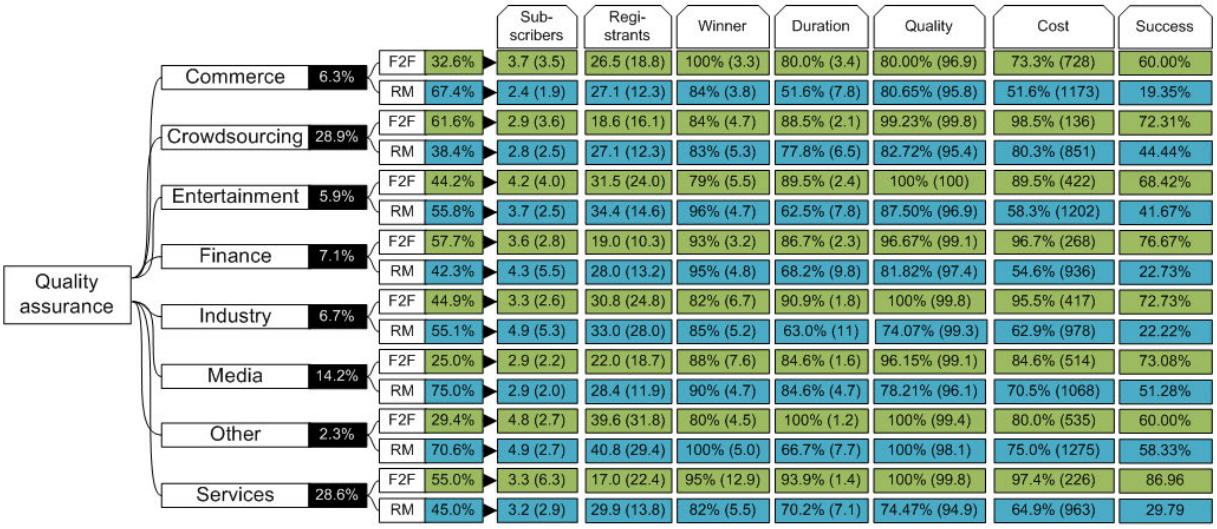

FIGURE 12. Quality assurance challenges success rates and contest settings.

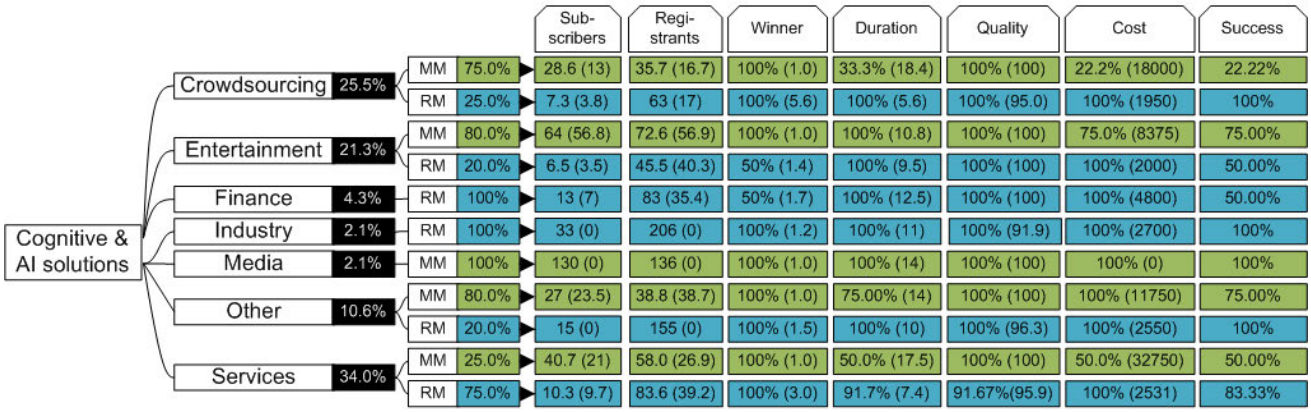

FIGURE 13. Cognitive & AI solutions challenges success rates and contest settings.

- **Cost:** The cost of Cognitive & AI challenges is increased compared to the other types of challenges mainly due to the specificity and the complexity of the problems crowdsourced. Cognitive & AI present the higher awards (14.000$ on average at MMs, 2.700$ on average at RMs).

V. DISCUSSION

In this section we interpret the results and discuss the implications of the findings to researchers and practitioners. Based on the Overall Success of all types of challenges, we observed that F2F types of contests are more efficient in terms of all success indicators. By focusing on the F2F contests and comparing them to the corresponding RM contests we identified four major differences. In particular, F2F contests:

- crowdsource tasks of limited scope, requiring less time compared to RMs and therefore are assigned to smaller prizes. For example, in Development challenges an F2F task might be a small script for updating an instance of the system while an RM task may include the development of an API;
- are well documented presenting clearly the scope of the challenge, without overwhelming information;
- present clear acceptance criteria for a winning solution; and
- involve (in their majority) Quality Assurance challenges. Bug hunting contests when formulated as ''The first one that finds more bugs are the one that wins'' are easily perceived by the contributors and present great success potentials.

Additionally, *Cognitive & AI s*olutions are the most popular in terms of crowd participation, probably due to the high prizes and the challenging nature of the crowdsourced tasks. This fact explains also the high failure percentages of these contests since (often) a winning solution cannot be obtained, especially when the contests are crowdsourced as RMs, instead of MMs. For challenging problems (such as

Cognitive & AI) that require more time and greater prizes are assigned, it seems that MMs can offer greater control mechanisms of the submissions and can more effectively handle the variability in the quality of the winning solutions. Design and Development tasks are the most frequently crowdsourced. It seems that they present great success potentials especially when they are crowdsourced as F2F challenges. In both cases it seems that careful decomposition of the problem into tasks that can be effectively crowdsourced can help towards improving the percentages of successful contests in terms of duration and cost.

The major findings of this study suggest that there are significant differences across different crowdsourced software development activities, in terms of success indicators. This fact can be further investigated by researchers and exploited by practitioners. We encourage ***researchers*** to explore the factors introduced in this paper and propose new metrics and procedures, for quantifying crowdsourcing success factors. In this study, to ensure the objectivity and reliability of the results, we selected to examine a small set of metrics, the majority of which were raw data, directly provided and measured from TopCoder. There are also other low-level metrics such as task size, task difficulty, task specificity, not considered within the scope of this study due to the fact that there are no formal procedures yet for quantifying them within the context of CSE. We believe that a well-specified procedure for quantifying such low-level metrics will help into: (a) more accurately describing the tasks to be crowdsourced; and (b) better formulating crowdsourcing success factors and testing their effects. Furthermore, we encourage researchers to compare the performance and the overall management and success of crowdsourced projects against non-crowdsourced ones.

Additionally, we believe that an analysis of success factors and indicators on contests performed on other popular software development crowdsourcing platforms like Bountify or even in BugCrowd, which is a specialized platform for Bug finding and fixing, will provide insights on the influence of crowdsourcing factors to CSE success in different environment settings. In such a study, it would be interesting to make a comparison within the crowdsourcing host platforms and observe the differences between the types of performed challenges, the costs and duration of related tasks and explore how crowdsourcing success factors and indicators vary with respect to the different platforms. Such deviations are expected, since platforms are expected to launch crowd sourced tasks of different granularities and complexities. In such a case, it is important to experiment on the appropriate task decomposition mechanisms for defining the right granularity of tasks that will lead to more successful challenges in terms of time, costs, quality and participation. Similarly, to any empirical endeavor, we encourage the replication of this study in larger samples of challenges, performed in different platforms, including more metrics describing the tasks crowdsourced.

Regarding ***practitioners***, the decision trees presented in Figs. 10-13 along with the analysis performed in Section IV can be a useful tool in the hand of challenge providers, during the decision-making process, when configuring crowdsourcing software development challenges. Based on the results of this study, software developers can have indications on the level to which different crowdsourcing goals are achieved across different software development activities. In this context a software practitioner can employ the following four step process to define crowdsourcing needs of his company:

- Select the goal of crowdsourcing (is it Quality, Participation, Low costs or Duration?)
- Based on the goal and the company needs select the challenge to be crowd
- Based upon goal and challenge type define the contest settings (select contest type, define prize, define duration (if needed), describe the challenge and set quality thresholds with the help of Figs. 6 to 9.
- After the completion of the challenge compare the actual success indicators of the challenge to the expected ones, and customize the empirical models presented in this study to the company's findings.

VI. THREATS TO VALIDITY

In this section we discuss the threats to validity based on the four main types appointed by Runeson and Höst [24]: construct, internal, external and reliability validity. Construct validity defines how effectively a test or experiment measures the hypotheses examined. Internal validity examines weather the causal relationships identified by the experiment performed are supported by the experimental environment. External validity is related to the generalization of the study results to other relevant settings and Reliability is associated to the reproducibility of the study by other researchers, i.e. considering that they will be able to repeat the same process, collect data and reach the same results.

Construct Validity: The set of metrics selected to assess the success of crowd sourced software development activities could pose threats to construct validity. It goes without saying that there are also other factors that may affect crowd sourced software development success (e.g., platform usability, crowd collaboration, project size) that we have not considered in this study. Since in this study we focused on analysing data from TopCoder platform, a pioneer in crowdsourced software development, platform usability was excluded as it could be considered the same for all types of challenges and activities studies. Additionally, crowd collaboration [40] was out of the scope of the study since we focused on competitive CSE, not collaborative and therefore factors addressing crowd collaboration were not applicable in this case. Another threat to validity is the selection of thresholds for transforming continuous variables to binary ones. In order to minimize the subjectivity in the selection of thresholds a solid mathematical approach has been employed. Nevertheless, the use of different thresholds might alter the results.

TABLE 14. Deviation from success indicators per specific application domains.

		#	%	Prize	Winners	Duration	Quality	Success
Commerce	CRM	100	4,54%	7,40%	11,80%	1,20%	-16,60%	3,40%
	eCommerce	16	0,73%	-14,10%	4,80%	-7,30%	-0,90%	-13,10%
	Marketing	62	2,81%	-14,00%	-7,60%	-7,30%	-12,10%	-17,20%
Social media	Social networks	104	4,72%	-11,80%	-5,50%	-2,60%	-11,50%	-14,90%
	Crowdsourcing	400	18,16%	12,90%	-4,70%	4,00%	6,10%	8,20%
Entertainment	Culture	159	7,22%	-4,10%	-8,20%	13,70%	-21,30%	4,90%
	Education	25	1,13%	-2,30%	11,80%	13,70%	-54,70%	-3,50%
	Games	9	0,41%	-5,70%	7,10%	-7,20%	9,10%	-1,00%
	Nature	43	1,95%	-21,50%	4,90%	-3,50%	-21,30%	-10,80%
	Religion	29	1,32%	19,90%	11,80%	13,70%	12,00%	40,90%
	Social	4	0,18%	-0,20%	-7,70%	-3,90%	3,50%	-6,30%
	Sports	15	0,68%	6,60%	11,80%	7,00%	12,00%	20,90%
	Travelling	21	0,95%	-13,40%	2,30%	-14,90%	12,00%	-11,50%
Finance	Banking	17	0,77%	-15,40%	5,90%	-9,80%	4,30%	-17,90%
	Finance	72	3,27%	-12,00%	4,90%	-3,00%	4,30%	-10,50%
	Insurance	63	2,86%	-13,40%	7,00%	7,40%	-6,20%	-5,10%
Industry	Agriculture	13	0,59%	12,20%	11,80%	-17,10%	12,00%	2,40%
	Civil Engineering	1	0,05%	19,90%	11,80%	13,70%	-88,00%	-59,10%
	Healthcare	85	3,86%	-10,70%	-1,10%	-3,90%	-8,80%	-9,70%
	Management	32	1,45%	-23,80%	5,60%	5,40%	-3,20%	-15,30%
	Occupation	48	2,18%	-8,20%	5,60%	13,70%	-8,80%	-2,80%
Media	Audio/Video	180	8,17%	1,00%	3,50%	1,50%	-3,70%	4,80%
	Graphics	47	2,13%	11,40%	5,40%	-3,30%	5,30%	9,00%
Other	Programming	17	0,77%	10,10%	2,00%	-3,40%	9,10%	6,80%
	Science	41	1,86%	-3,60%	11,80%	-3,90%	12,00%	5,60%
Services	Analytics	239	10,85%	2,70%	4,70%	0,30%	-0,40%	4,90%
	Communications	76	3,45%	8,80%	-10,40%	2,60%	12,00%	7,60%
	Drones	201	9,12%	-22,60%	-10,70%	3,70%	-4,70%	-11,60%
	Energy Management	9	0,41%	-23,80%	-6,90%	7,50%	-21,30%	-9,10%
	IoT	16	0,73%	-1,20%	-30,30%	-49,50%	-21,30%	-38,00%
	Utility	40	1,82%	6,70%	-7,90%	-7,40%	-1,00%	-5,20%
	Virtualisation	19	0,86%	11,90%	5,30%	7,70%	3,80%	18,00%

Furthermore, metrics, for which there was not a clear and reliable method for calculating them, like project size and difficulty, were also excluded in order to avoid introducing researcher bias in the model. In general, we selected to use metrics that were directly available by the TopCoder platform so as to ensure the objectivity of the results. As our target is to provide an easy to use, reliable method for helping challenge providers define the activities and the configuration for crowdsourcing them we believe that we included the most important factors that should be considered, i.e., type of activity, time, cost, quality, participation. Although we agree that more CSE metrics, like platform usability, size and complexity of crowd sourced activity, collaboration metrics, can further be considered in other contexts.

Internal Validity: The target of the proposed study was to explore whether there are differences between success factors and indicators (such as contest type, platform, duration, quality, participation) across the different software development activities crowd sourced. We identified trends and common practice when it comes to crowdsourcing different software development activities; however, we do not claim that these form causal relationships.

Reliability: We believe that the followed research process ensures the reliability and safe replication of our study. The process that has been followed in this study has been thoroughly documented in the case study protocol, provided in Section III. Additionally, the data on which this study is based are publicly available in therefore, the re-production of the case study can be easily performed by any interested researcher. However, concerning researcher bias, although the majority of data participating in the analysis are raw data directly provided by TopCoder platform, we should state that it was introduced during the data collection and data analysis

process in one case, while classifying crowdsourcing challenges to particular application domains. In that case the challenge description was inspected by the second author, who identified frequently appearing keywords that helped him classify the challenges into the existing application domains. In order to increase the reliability of this process the first and the third author validated the results.

External Validity: To ensure the generalizability of our results we have examined a wide range of challenges providing a number of different activities crowd sourced that offer a representative sample of the population. Though as mentioned in RQ1, not all types of challenges are equally represented in this sample, but still this shows a trend appointing that certain types of challenges are rarely selected probably due to the criticality of the task crowdsourced and the high costs. Additionally, we note that (as in almost any case study) in case that the sample challenges change, the results are expected to deviate from those reported in the study. Similarly, the results might not be generalizable to other platforms, in the sense that using challenges that come from other hosting platforms, would require an adjustment in the used model, so as to map the environment settings of the new platform. Therefore, a replication of this study in an even larger multi-platform set of challenges would be valuable in verifying and generalizing current findings. By acknowledging the fact that not all crowdsourcing activities are formulated as ''contest-based'' we note that the results cannot be generalized to all types of crowd-sourced challenges.

VII. CONCLUSION

Crowdsourcing practices in the domain of software engineering have proved to be a promising and viable approach to software related problems in terms of quality, cost and time reduction. However, selecting the appropriate software engineering tasks to be crowdsourced and deciding upon the important managerial decisions for formulating such challenges is far from trivial due to a number of parameters that affect the success of the whole endeavor. In this study, we have performed a case study on 2,209 contests launched during 2018 through TopCoder platform and explored the level to which crowdsourcing success factors and indicators vary across different software engineering activities. The findings of the study suggest that there are significant differences in the level to which crowdsourcing goals are reached across different software development activities. In terms of participation Cognitive & AI contests are the most popular ones, in terms of duration Quality Assurance contests present the minimum times, in terms of cost and quality Development challenges present the lowest prizes and achieve among the highest quality scores. The results are summarized in decision graph models that capture the variance of success factors across different challenges and contest settings. Such information can prove to be useful to practitioners who will easily identify and decide upon important settings of crowdsourced projects.

APPENDIX

See Table 14.

REFERENCES

[1] T. Alelyani and Y. Yang, ''Software crowdsourcing reliability: An empirical study on developers behavior,'' in *Proc. 2nd Int. Workshop Softw. Anal.*, New York, NY, USA, 2016, pp. 36–42.
[2] N. Archak, ''Money, glory and cheap talk: Analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on TopCoder.com,'' in *Proc. 19th Int. Conf. World Wide Web*, New York, NY, USA, 2010, pp. 21–30.
[3] V. R. Basili, ''Software modeling and measurement: The goal/question/metric paradigm,'' Univ. Maryland College Park, College Park, MD, USA, Tech. Rep. CS-TR-2956, 1992.
[4] E. Bonabeau, ''Decisions 2.0: The power of collective intelligence,'' *MIT Sloan Manage. Rev.*, vol. 50, no. 2, pp. 45–52, 2009.
[5] L. Dahlander and H. Piezunka, ''Open to suggestions: How organizations elicit suggestions through proactive and reactive attention,'' *Res. Policy*, vol. 43, no. 5, pp. 812–827, Jun. 2014.
[6] I. Dissanayake, J. Zhang, and B. Gu, ''Task division for team success in crowdsourcing contests: Resource allocation and alignment effects,'' *J. Manage. Inf. Syst.*, vol. 32, no. 2, pp. 8–39, Apr. 2015.
[7] A. Doan, R. Ramakrishnan, and A. Y. Halevy, ''Crowdsourcing systems on the world-wide Web,'' *Commun. ACM*, vol. 54, no. 4, pp. 86–96, Apr. 2011.
[8] A. Dwarakanath, N. C. Shrikanth, K. Abhinav, and A. Kass, ''Trustworthiness in enterprise crowdsourcing: A taxonomy & evidence from data,'' in *Proc. 38th Int. Conf. Softw. Eng. Companion*, New York, NY, USA, 2016, pp. 41–50.
[9] Elo, A. *The Rating of Chessplayers, Past and Present*. Nagoya, Japan: Pub Arco, 1978.
[10] B. Fitzgerald and K. J. Stol, ''The dos and dont's of crowdsourcing software development,'' in *Proc. Int. Conf. Current Trends Theory Pract. Informat.* Berlin, Germany: Springer, 2015, pp. 58–64.
[11] G. Hart, ''The five W's: An old tool for the new task of task analysis,'' *Tech. Commun.*, vol. 43, no. 2, pp. 139–145, 1996.
[12] J. Howe, *Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business*, 1st ed. New York, NY, USA: Crown, 2008.
[13] M. R. Karim, Y. Yang, D. Messinger, G. Ruhe, ''Learn or earn?—Intelligent task recommendation for competitive crowdsourced software development,'' in *Proc. 51st Hawaii Int. Conf. Syst. Sci. (HICSS)*, Hilton Waikoloa Village, HI, USA, Jan. 2018, pp. 1–10.
[14] A. Kittur, ''The Future of Crowd Work,'' in *Proc. CSCW*, New York, NY, USA, 2013, pp. 1301–1318.
[15] J. Kontio, J. Bragge, and L. Lehtola, ''The focus group method as an empirical tool in software engineering,'' in *Guide to Advanced Empirical Software Engineering*. London, U.K.: Springer, pp. 93–116.
[16] T. D. LaToza, W. Ben Towne, A. Van Der Hoek, J. D. Herbsleb, ''Crowd development,'' in *Proc. 6th Int. Workshop Cooperat. Hum. Aspects Softw. Eng. (CHASE)*, May 2013, pp. 85–88.
[17] T. D. LaToza and A. van der Hoek, ''Crowdsourcing in software engineering: Models, motivations, and challenges,'' *IEEE Softw.*, vol. 33, no. 1, pp. 74–80, Jan. 2016.
[18] N. Leicht, D. Durward, I. Blohm, and J. M. Leimeister, ''Crowdsourcing in software development: A state-of-the-art analysis,'' in *Proc. 28th Bled eConference*, Maribor, Slovenia, 2015, pp. 389–403.
[19] K. Li, J. Xiao, Y. Wang, and Q. Wang, ''Analysis of the key factors for software quality in crowdsourcing development: An empirical study on TopCoder.Com,'' in *Proc. IEEE 37th Annu. Comput. Softw. Appl. Conf.*, Jul. 2013, pp. 812–817.
[20] A. Khanfor, Y. Yang, G. Vesonder, G. Ruhe, and D. Messinger, ''Failure prediction in crowdsourced software development,'' in *Proc. 24th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Nanjing, China, Dec. 2017, pp. 495–504.
[21] K. Mao, L. Capra, M. Harman, and Y. Jia, ''A survey of the use of crowdsourcing in software engineering,'' *J. Syst. Softw.*, vol. 126, pp. 57–84, Apr. 2017.
[22] D. Mehta, ''An insight into software crowd sourcing: How crowd can transform the business model for technology service providers,'' *Int. J. Comput. Appl.*, vol. 101, pp. 34–40, Sep. 2014.
[23] P. Runeson, C. Wohlin, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Germany: Springer, 2012.

[24] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Softw. Eng.*, vol. 14, no. 2, Apr. 2009, Art. no. 131.
[25] R. L. Saremi, Y. Yang, G. Ruhe, and D. Messinger, "Leveraging crowdsourcing for team elasticity: An empirical evaluation at TopCoder," in *Proc. IEEE/ACM 39th Int. Conf. Softw. Eng., Softw. Eng. Pract. Track (ICSE-SEIP)*, May 2017, pp. 103–112.
[26] A. Sarı, A. Tosun, and G. I. Alptekin, "A systematic literature review on crowdsourcing in software engineering," *J. Syst. Softw.*, vol. 153, pp. 200–219, Jul. 2019.
[27] M. A. Sohibani, N. A. Osaimi, R. A. Ehaidib, S. A. Muhanna, and A. Dahanayake, "Factors that influence the quality of crowdsourcing," in *New Trends in Database and Information Systems II*. Cham, Switzerland: Springer, 2015, pp. 287–300.
[28] K.-J. Stol, B. Caglayan, and B. Fitzgerald, "Competition-based crowdsourcing software development: A Multi-method study from a customer perspective," *IEEE Trans. Software Eng.* vol. 45, no. 3, pp. 237–260, Mar. 2019.
[29] K.-J. Stol and B. Fitzgerald, "Researching crowdsourcing software development: Perspectives and concerns," in *Proc. 1st Int. Workshop Crowd-Sourcing Softw. Eng. (CSI-SE)*, New York, NY, USA, 2014, pp. 7–10.
[30] K.-J. Stol and B. Fitzgerald, "Two's company, three's a crowd: A case study of crowdsourcing software development," in *Proc. 36th Int. Conf. Softw. Eng. (ICSE)*. New York, NY, USA: ACM, 2014, pp. 187–198.
[31] K.-J. Stol, T. D. LaToza, and C. Bird, "Crowdsourcing for software engineering," *IEEE Softw.*, vol. 34, no. 2, pp. 30–36, Apr. 2017.
[32] A. Suganthy and T. Chithralekha, "Application of crowdsourcing in software development," in *Proc. Int. Conf. Recent Trends Inf. Technol. (ICRTIT)*, Apr. 2016, pp. 1–6.
[33] H. Tajedin and D. Nevo, "Determinants of success in crowdsourcing software development," in *Proc. Annu. Conf. Comput. People Res. (SIGMIS-CPR)*, New York, NY, USA, 2013, pp. 173–178.
[34] N. H. Thuan, P. Antunes, and D. Johnstone, "Factors influencing the decision to crowdsource," in *Proc. Int. Conf. Collaboration Technol.* Berlin, Germany: Springer, Oct. 2013, pp. 110–125.
[35] L. Vaz, I. Steinmacher, and S. Marczak, "An empirical study on task documentation in software crowdsourcing on TopCoder," in *Proc. ACM/IEEE 14th Int. Conf. Global Softw. Eng. (ICGSE)*, May 2019, pp. 48–57.
[36] X. Wang, W. Wu, and Z. Hu, "Evaluation of software quality in the TopCoder crowdsourcing environment," in *Proc. IEEE 7th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2017, pp. 1–6.
[37] E. R. Q. Weidema, C. López, S. Nayebaziz, F. Spanghero, and A. van der Hoek, "Toward microtask crowdsourcing software design work," in *Proc. 3rd Int. Workshop CrowdSourcing Softw. Eng. (CSI-SE)*, New York, NY, USA, 2016, pp. 41–44.
[38] W. Wu, W.-T. Tsai, and W. Li, "An evaluation framework for software crowdsourcing," *Frontiers Comput. Sci.*, vol. 7, no. 5, pp. 694–709, Oct. 2013.
[39] Y. Yang, M. R. Karim, R. Saremi, and G. Ruhe, "Who should take this task?: Dynamic decision support for crowd workers," in *Proc. 10th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas.*, New York, NY, USA, 2016, pp. 8:1–8:10.
[40] D. Yu, Z. Zhou, and Y. Wang, "Crowdsourcing software task assignment method for collaborative development," *IEEE Access*, vol. 7, pp. 35743–35754, 2019.

STAMATIA BIBI received the B.Sc. degree in informatics and the Ph.D. degree in software engineering from the Aristotle University of Thessaloniki, Greece, in 2002 and 2008, respectively. She is currently an Assistant Professor in software engineering with the Department of Electrical and Computer Engineering, University of Western Macedonia, Kozani, Greece. Her research interests include process models, cost estimation, quality assessment, and cloud computing.

IOANNIS ZOZAS received the B.Sc. and M.Sc. degrees in applied informatics from the University of Macedonia. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Western Macedonia, Greece. He currently works as an IT Specialist with the National Bank of Greece. His research interests include software quality models, and software maintenance and evolution.

APOSTOLOS AMPATZOGLOU received the B.Sc. degree in information systems, the M.Sc. degree in computer systems, and the Ph.D. degree in software engineering from the Aristotle University of Thessaloniki, in 2003, 2005, and 2012, respectively. He is currently an Assistant Professor in software engineering with the Department of Applied Informatics, University of Macedonia, Greece. Before joining the University of Macedonia, he was an Assistant Professor with the University of Groningen, The Netherlands. He has published more than 80 articles in international journals and conferences, and is/was involved in over 15 Research and Development ICT projects, with funding from national and international organizations. His current research interests are focused on technical debt, maintainability, reverse engineering, quality management, and design.

PANAGIOTIS G. SARIGIANNIDIS received the B.Sc. and Ph.D. degrees in computer science from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2001 and 2007, respectively. He has been an Assistant Professor with the Department of Electrical and Computer Engineering, University of Western Macedonia, Kozani, Greece, since 2016. He has published over 120 articles in international journals, conferences, and book chapters. He has been involved in several national, EU, and international projects. He is currently a Project Coordinator of the H2020 Project entitled SPEAR: Secure and Private Smart Grid (H2020-DS-2016-2017/H2020-DS-SC7-2017). His research interests include optical and wireless telecommunications, resource allocation, the Internet of Things, and security and privacy in smart networks.

GEORGE KALAMPOKIS received the B.Sc. degree from the Department of Electrical and Computer Engineering, University of Western Macedonia, in 2017. He is currently pursuing the master's degree in machine learning techniques with the Aristotle University of Thessaloniki.

IOANNIS STAMELOS received the Diploma degree in electrical engineering and the Ph.D. degree in computer science from the Aristotle University of Thessaloniki, in 1983 and 1988, respectively. He is currently a Professor with the Department of Informatics, Aristotle University of Thessaloniki, where he carries out research and teaching in the area of software engineering. He has published more than 200 articles in international journals and conferences. His current research interests are focused on open source software engineering, software project management, and software education. He is/was the scientific coordinator or principal investigator for his university in 30 research and development projects in information and communication technologies with funding from national and international organizations.