Yu et al., Int J Comput Softw Eng 2020, 5: 155
https://doi.org/10.15344/2456-4451/2020/155

International Journal of
Computer & Software Engineering

Research Article Open Access

Understanding Risk in Software Crowdsourcing: A Preliminary Taxonomy

Daiwei Yu[1], Ye Yang[2] and Yong Wang[1*]

[1]*Department of Computer Science and Technology, Ocean University of China, Qingdao, China*
[2]*School of Systems and Enterprises, Stevens Institute of Technology, Hoboken NJ, USA*

Abstract

Software crowdsourcing (SC) is booming as a popular paradigm for rapid solution development. However, unique characteristics of this emerging paradigm, such as external resources and black-box nature of crowdsourced processes, introduce inherent risk to crowdsourcing. This paper aims at developing better understanding towards risk in SC. To that end, we conduct a literature review of 36 relevant articles, and propose a preliminary taxonomy of SC risk including 13 risk types. These 13 risk types are organized with a two-dimensional, processes vs. entities, structure. For each risk, its description and cause / consequence are introduced. Analysis of possible reasons and practical mitigation suggestions are also offered for crowdsourcing practitioners to better cope with risks. The preliminary results will help both requesters and platforms to be alert to risks by understanding them, so as to ensure the achievement of expected benefits of SC.

Publication History:

Received: March 01, 2020
Accepted: April 23, 2020
Published: April 25, 2020

Keywords:

Software crowdsourcing, Risk taxonomy, Mitigation, Literature review

Introduction

The term "crowdsourcing" was coined by Howe in 2006 [1] as the act of open, collective problem solving by undefined network of people. Specifically, the paradigm of software crowdsourcing (SC) typically involves three kinds of roles, namely platforms, companies (i.e., clients) and workers (i.e., online developers) [2] for open software production. Reported benefits for crowdsourcing clients include shortened development duration [2-4], unlimited resource supply [5], and cost reduction [2,3,6].

Nonetheless, some studies also revealed warnings with respect to negative consequences associated with SC, such as low task quality or task starvation. In cases of task starvation, i.e., receiving no submissions, the clients had to re-post their tasks, leading to potential schedule delay [7]. Specifically, an average of 15.6% failure rate in SC is reported [8]. In addition, other concerns were also discussed in existing studies such as hesitation or resistance among internal employees [9], disputes on ownership of solutions [10, 11], and violation of company security [12]. Apparently, crowdsourcing is no silver-bullet answer. Without proper planning and management, such negative consequences may offset the benefits of crowdsourcing approaches, and potentially lead to failure.

Existing studies attempted to investigate risk factors and causes associated with SC paradigm, using roughly six different terms to indicate problems and troubles in the process of SC in respective context, i.e., namely *barrier, challenge, issue, risk, concern, and uncertainty*. Specifically, some use the term "barrier" [13-16] to refer to the communication and collaboration difficulties in SC; some use "challenge" [17-28] to adopt the SC paradigm from a broader ecosystem perspective; some use "risk" [9, 11, 29-32] to refer to dynamic, task-level influencing factors at a finer granularity; some use "concern" [7, 33-35] to refer to typical concerns from the practitioner perspective concluded in a case study; and others use vaguely defined terms such as "issue" [12, 36, 37]. In a recent study, Law et al. used the term "uncertainty" to express possible problems related to entities influencing SC [38]. Risk refers to the combination of the possibility of a certain dangerous event and its consequences. It manifests itself as uncertainty of loss. During the process of SC activities, various uncertain events are encountered, which may affect the success of SC tasks. In this study, we adopt the concept of "risk" to study risks, challenges, barriers, concerns, issues and uncertainty in SC context. We believe that the term of "risk" provides a more comprehensive view based on existing variations.

On the one hand, SC processes share some common risk factors with traditional software methodologies. For example, requirement risk may occur in both traditional software methodologies and SC, and the common risk factor in requirements is unclarity and ambiguity. On the other hand, SC also exhibits some special risk characteristics. It is constantly subject to uncertain changes due to the loss of control and visibility to individual worker's behaviors. For example, SC tasks are registered by anonymous workers from all over the world, who may submit plagiarized code [28] or poor-quality solutions [22]. Such risks may threaten task success. In addition, there are some common misconceptions among software practitioners. On the optimistic side, some people believe that crowdsourcing is a better way to go without worrying about the risk. On the realistic side, some companies experience pushing back due to low task quality [22], task starvation [10], and employees' reluctance (i.e., turbulence) [9]. Therefore, there is an essential need for a synthesized body of knowledge in order to support more informed risk management in SC practice.

To that end, this paper aims at exploring and extracting special risk items and reasons in the context of SC, in order to support managers to discern and manage their SC project risks in a more effective manner. Based on the existing literature review, this paper proposes a preliminary taxonomy of SC risk including 13 risk types. These are organized with a two-dimensional, processes vs. entities, structure. For each risk, its description and cause / consequence are presented. Analysis of possible reasons and practical mitigation suggestions are also offered for crowdsourcing practitioners to better cope with risks.

The rest of this paper is organized as follows. Section 2 introduces the background and related work to set the context for following passage. Section 3 introduces the research methodology of this paper. Section 4 analyzes the results of the literature review. Section 5 presents the proposed risk taxonomy in SC with 13 risk items. Section 6 provides suggestions to requesting clients and crowdsourcing platforms. Section 7 concludes this paper with future work.

***Corresponding Author:** Dr. Yong Wang, Department of Computer Science and Technology, Ocean University of China, 238 Songling Rd, Laoshan Qu, Qingdao Shi, China; E-mail: wangyong@ouc.edu.cn

Citation: Yu D, Yang Y, Wang Y (2020) Understanding Risk in Software Crowdsourcing: A Preliminary Taxonomy. Int J Comput Softw Eng 5: 155. doi: https://doi.org/10.15344/2456-4451/2020/155

Background and Related Work

Background

Almost every type of software engineering activity can be candidate for crowdsourcing [39]. But in general, development and test are the two most common types of SC tasks. A software project is typically decomposed into several mini tasks before ready for crowdsourcing, each with specific requirements, duration, award, required skills, etc. On the one hand, the client needs to spend time decomposing projects into appropriate granularity tasks [19]. If the decomposition granularity is too fine, it will take more time to complete; if the granularity is too coarse, it will be difficult for workers to understand. On the other hand, the setting of duration time and award is different from traditional software development [41]. The duration of SC is shorter, usually several days. And the award is lower. Too long duration and too high award is unnecessary. Improper decomposition and estimation of settings will cause tasks facing risks.

Crowd workers usually come from all over the world with diverse background in terms of technical skills, availability, and so on [9]. In general, they are highly skilled workers or talents who could finishing tasks perfectly and meet clients' expectations [5, 6]. However, clients have no control over the crowd workers signed up on their tasks, unlike in traditional software development. Crowd workers usually register for tasks based on their interest and availability, and then they complete tasks independently [10]. In addition, there is a high risk of 82.9% crowd workers may become unengaged and quit the tasks in the middle of the tasks [8].

In summary, many factors in SC process, inputs and resources may cause risks in successful crowdsourcing. SC is a new topic and the risk of it has not been properly structured yet; existing researches raised some risks, but they are relatively few; now a comprehensive and clearly classified risk taxonomy is lacking. This is the motivation of the study.

Related Work

Existing literatures proposed and analyzed several types of risk.

Communication/Collaboration barriers: Zanatta et al. identified risks related to competence, collaboration, and time management faced by crowd workers [15]. LaToza et al. articulated a series of risks, such as decomposition risk and communication risk [20]. Farid et al. identified some risks regarding to security, communication, and motivation [22].

Process/Ecosystem risk: Kannangara et al. applied the perspective of business ecosystems to the process of crowdsourcing to conceptualize crowdsourcing-based business ecosystems and identified eight risk categories, such as intellectual property risk and quality risk [11]. Hasteer et al. presented SC risks with regard to schedule, cost, and quality [7]. Stol et al. proposed and analyzed six risks, including decomposition, communication, and intellectual property [16, 33, 34]. Suganthy et al. identified the basic characteristics of SC artifacts and their related risks [36].

Risk management: Some studies established a methodology to mitigate risks. Dwarakanath et al. presented a software development methodology that addresses key risk factors for the application of enterprise crowdsourcing [19]. Some studies gave out recommendations on how to minimize the risks they proposed [11, 15].

People factors: Newcomers are new to crowdsourcing platforms and even the concept of crowdsourcing. They are not familiar with crowdsourcing process and operation methods. They may only sign up tasks without completion. Several studies have focused on this aspect. Zanatta et al. identified risks to SC newcomers and mitigation ways [13]. Machado et al. highlighted that the onboarding process for newcomers is seen as challenging [21].

Risk classification: Several studies classified the SC risks they found. They divided risks according to different dimensions, such as crowdsourcing process and influence entities. Kamoun et al. classified the risks according to a proposed crowdsourcing lifecycle including five phases - initiation, preparation, engagement, evaluation, and commitment [29]. Law et al. illuminated how research norms and practitioners' dispositions were related to risks around five different entities - processes, data, knowledge, delegation, and quality [38].

In summary, existing studies provide multi-folds perspective towards understanding SC risks, but there is a lack of comprehensive and clearly structured SC risk taxonomy to support more proactive understanding and management of risk in SC practice. This motivates the study in this paper.

Methodology

This study follows the methodology depicted in Figure I. First, a literature review is conducted to examine SC risks in existing studies. Next, a taxonomy of SC risk is developed based on synthesizing and extending the review results. Besides, some suggestions are formulated for practitioners to identify and manage these risks appropriately. Each step will be introduced in more details in the context of the study.

Literature Review

In this study, the following query is used to search for relevant research, and the scope of search includes three main databases, namely the ACM Digital Library, the IEEE Xplore Digital Library, and Google Scholar Search:

"crowdsourcing" AND
("risk" or "obstacle" or "barrier" or "challenge" or "concern" or "issue" or "uncertainty" or "threat") AND ("software engineering" or "software development")

The search results are screened in four rounds. In the first round, 505 conference and journal papers were searched out, of which 91, 47, 367 were searched from the three major literature databases. In the second round, we manually filtered out irrelevant papers and duplicates by checking the titles and keywords, and this led to 65 papers remaining. In the third round, we screened the abstracts of these papers and eliminated some papers on crowd funding and civic wisdom, leaving 48 papers remaining. In the fourth round, we read the full text of the rest of the papers and excluded those related to general crowdsourcing. At last, the final 36 papers were remained for detailed research.

For the 36 selected papers, we first extracted their basic information, such as publication year, author, organization, source, source type, and keywords. Then, we read them manually and extracted the main attributes regarding SC risk. The main attributes of the template are summarized in Table 1. Besides, we recorded the crowdsourcing

Attribute	Definition
Paper ID	The serial number of the literature.
Risk ID	The unique identifier of a particular type of SC risk discussed in the literature. Each literature may contain multiple risk items.
Risk Name	The name of a risk item.
Risk Description	The brief textual description of a risk item.
Root Cause	The set of root causes or influencing factors for a unique risk item.
Consequence	The negative impact on project cost, schedule, and quality due to the occurrence of a risk item.
Mitigation	The recommended strategies/actions to prevent or mitigate a particular type of risk.
Crowdsourcing Methodology	Competitive-based vs. collaborative-based crowdsourcing activities.
Crowdsourcing Platform	Specific platform for hosting the crowdsourcing activities.
Risk-related Terminology	Specific terminologies for referring to risk factors associated with SC paradigm.

Table 1: Main attributes of the template we defined.

activity type, crowdsourcing platform, terminology describing risk, and other relevant information of each paper. Based on this information, a preliminary taxonomy of SC risk was developed by the first author and peer reviewed by other two authors. The risk taxonomy will be elaborated in Section 5.

On Classifying Software Crowdsourcing Risk

As discussed above, in this study, we build a two-dimensional matrix structure to organize SC risks. More specifically, one is phase dimension for characterizing various phases in SC lifecycle adapted from a crowdsourcing lifecycle [29], and the other is entity dimension for representing different sources of risk factors adapted from five influence entities [38]. This two-dimensional structure clearly shows the phase in which risk occurs and the influence entity it has.

Phase Dimension: SC activities run through the entire lifecycle. Kamoun et al. proposed a crowdsourcing lifecycle that spans the entire project's process into five phases [29], i.e., initiation, preparation, engagement, evaluation, and commitment phase. This lifecycle is similar to SC lifecycle, but the boundary between some phases are often blurred with each other in practice. First, the initiation phase and the preparation phase are often parallel and iterative, thus we merge these two phases into one phase – the initiation phase. In this phase, clients design the requirements, estimations, awards, and other attributes of the task to publish it on the crowdsourcing platform. Second, workers register and submit tasks during engagement phase. Clients communicate with workers to obtain better quality and higher submitting rate, then the platform scores the submissions through peer review in the evaluation phase. We merge these two phases into one phase – the engagement phase, since these two phases involve the same risk, e.g., schedule risk. At last, the commitment phase mainly involves companies giving awards, integrating submissions, and summarizing lessons learned. In a word, we summarize and merge the SC lifecycle into three phases, i.e., the initiation, engagement, and commitment phase to simplify and better organize our findings.

Entity Dimension: In traditional software engineering, entity is something that exists in itself. For example, customers, products, and requirements are known as "entities". While in SC, entity is an element related to the process. Law et al. frame the crowdsourcing risks by five entity types - process, data, knowledge, delegation (i.e., worker), and quality [38]. These five entity types have some overlapping parts. For example, data entity is caused by untrustworthy behaviors of workers, which can be incorporated into the delegation entity. Knowledge entity is mainly related to the risk of intellectual property.

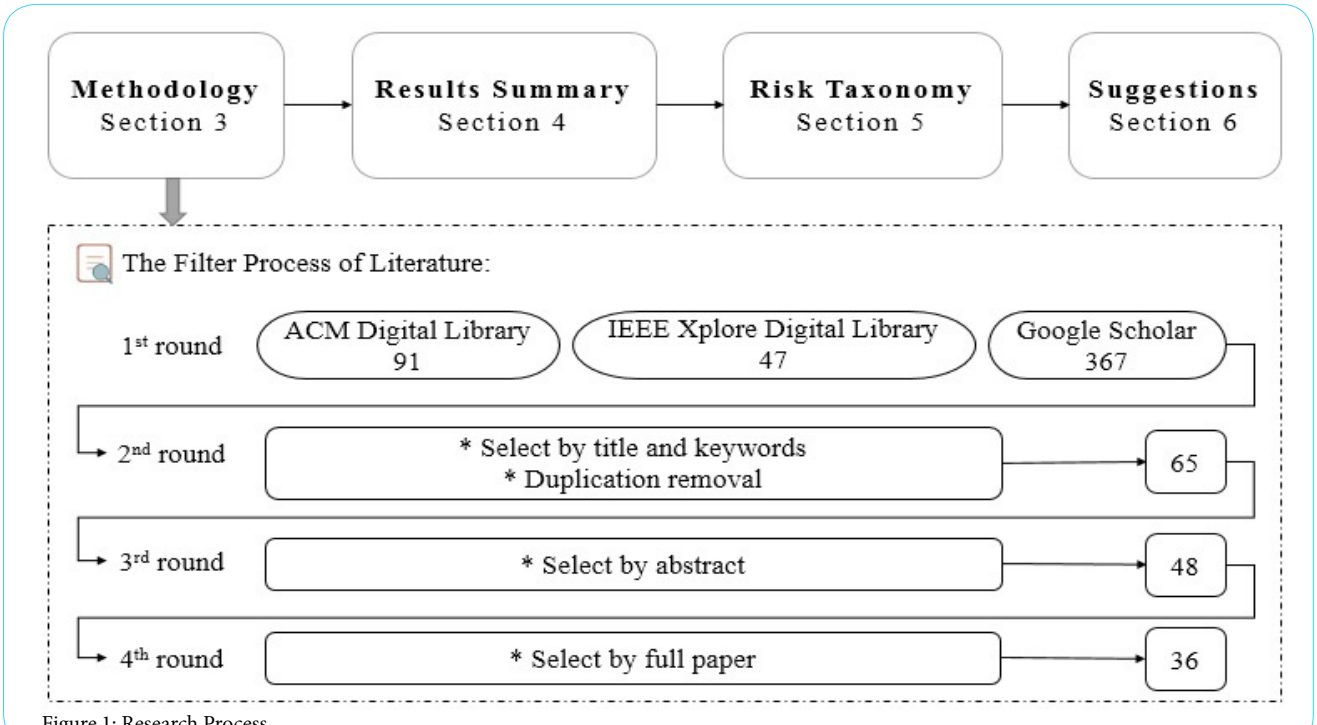

Figure 1: Research Process.

This is mainly due to the confusion of the submission ownership, malicious submissions, code plagiarism, etc., and appears at the end of the crowdsourcing process. So, we merge it to the process entity. In short, three entities remained - process, quality, and delegation.

Summary of Review Results

Risk mapping

In design/process coordinates, turbulence and decomposition risks may occur. If the requester does not coordinate the relationship between crowdsourcing activities and internal development before designing the crowdsourcing task, the company may experience organizational/cultural turbulence. If requesters do not decompose the large project into suitable crowdsourcing task modules when designing the task, they may face the decomposition risk. Among them, decomposition risk is mentioned in 8 papers, indicating that it has received much attention and needs to be paid attention by requesters. In design/quality coordinates, requirement and estimation risks may occur. Requirement risk refers to the unreasonable, unclear and incomplete requirement description. Estimation risk mainly discusses the unreasonable estimation of award and duration. If the requirement, duration and award are not properly designed, they will affect the final quality of the task. Some researchers consider both requirement risk and estimation risk as one. Since there are several studies on estimation, these two risks will be researched separately in this study. It is worth noting that there is no risk occurred in the design/delegation coordinates, because no crowd workers are involved in the design phase.

In engagement/process coordinates, schedule and communication risks may occur. It is worth noting that 17 papers mentioned communication risk (more than half of the literature we reviewed), which indicates that it has received extensive attention and research. In engagement/quality coordinates, starvation and quality risks may occur. Quality risk has received widespread attention, but starvation risk needs to be given more attention. Because once the starvation risk occurs, the SC task will fail due to starvation. In engagement/delegation coordinates, worker continuity and engagement risks may occur. Although there are few literatures discussing these two risks, managing them properly can improve the success rate of SC tasks.

In commitment/process coordinates, intellectual property (IP) risk may occur. Unlike traditional software development, IP risk only occurs in the context of the SC, which reminds requesters to understand and pay attention to it. In commitment/quality coordinates, integration risk may occur. It refers to incompatibilities that arise when the requester integrates submission into existing procedures within the company. If paying attention to this risk early, clients can effectively avoid these unnecessary troubles. In commitment/delegation coordinates, worker trustworthiness risk may occur, and then threaten company security and data security.

References of risks

Table 2 summarizes the results of the literature review -the risk and its citations organized in a two-dimensional structure, as introduced in the previous section 3.2. Note that the percentage numbers in the header row, the first column and other cells show the portion of papers in the whole 36 study set. The main findings are, among all 36 studies:

1) In the Phase dimension, the most discussed SC risk is related to the engagement phase, including issues associated with communication, quality, schedule, task starvation, worker continuity, and worker engagement, i.e., covered in 75% (27 number) of studies.

2) In the Entity dimension, the most discussed SC risk is related to the process entity, encompassing issues associated with organizational/culture turbulence, task decomposition, communication, schedule, and IP, i.e., covered in 72.2% (26 number) of studies.

3) Within the Engagement phases, 63% covers risk related to the process entity, 32.4% covers the quality entity, and 21.6% covers the delegation entity.

4) More specifically, across all studies:

a. The communication risk is discussed the most, i.e., in 47.2% (17 number) of studies;

b. The quality risk and intellectual property risk are both covered in 33.3% (12 number) of studies, respectively;

c. Only 27.8% number of papers propose risk mitigation suggestions, i.e., covered in 18 number of papers.

Phase \ Entity	Process (72.2%)	Quality (61.1%)	Delegation (36.1%)
Design (50%)	Turbulence (5.6%) [9, 29]	Requirement (22.2%) [13-16, 21, 22, 31, 36]	†
	Decomposition (25%) [7, 12, 14, 15, 19, 20, 21, 31, 34]	Estimation (19.4%) [7, 12, 22, 36, 41-43]	
Engagement (75%)	Communication (47.2%) [7, 9, 13-16, 18-20, 22-24, 31, 33-36]	Quality (33.3%) [9, 10, 22, 25, 29, 31-33, 35-37, 40]	Worker Continuity (5.6%) [33, 35]
	Schedule (2.8%) [7]	Starvation (11.1%) [10, 29, 32, 36]	Worker Engagement (16.7%) [14, 18, 26, 27, 44, 45]
Commitment (36.1%)	Intellectual Property (33.3%) [9-11, 17, 19, 28-30, 33, 35, 36, 40]	Integration (8.3%) [33, 35, 36]	Worker Trustworthiness (13.9%) [9-11, 22, 40]

Table 2: Mapping existing software crowdsourcing risks studies.
† Note: No risk occurs, since no workers involved here.

Taxonomy of Software Crowdsourcing Risk

The literature review leads us to develop a taxonomy of SC risks, following the same entity-phase categorization structure, as shown in Table 2. Each entity-phase pair corresponds to various types of risk items. There is a total of 13 unique risk types, which is further characterized by its causes and / or influence entities. This section will elaborate this taxonomy in more details. For simplicity, we will introduce the 13 risk types following the horizontal entity dimension only, in the order of process, quality, delegation entities through Section 4.1-4.3.

Risks related to process entity

As shown in Table 3, process entity corresponds to five risk types across all three phases of SC lifecycle, i.e., the risk of turbulence, decomposition, communication, schedule, and intellectual property.

1) **Turbulence**. It refers to the cases where in-house managers and / or employees may refuse or hesitate to accept SC as an alternative solution for software product development. This is ranked as the highest risk level of potential risk in crowdsourcing by Gebert [9], however, it is not reported in the majority of the studies. **Causes / Consequences**: One possible cause is that in-house employees may perceive crowdsourcing as a threat to their job security. This risk is often easily overlooked. However, if employees are not fully devoted, it is not conducive to implement SC.

2) **Decomposition**. It refers to the difficulty associate with decomposing a project into a set of mini tasks suitable for SC. It is essential to decompose SC tasks into smaller pieces. However, they are often concerned with specific contexts, for which decomposition may be non-trivial [12]. **Causes / Consequences**: Decomposition is an essential challenge in SC [7, 19, 34]. If the divided mini tasks are highly coupled with each other, i.e., task dependency is too high, it is difficult for workers to understand the entire complexity of the whole [20, 31]. If the divided tasks are big and complex, workers will experience difficulty to understand them [15].

3) **Communication**. It refers to the indirect, untimely and asynchronous communication between requesting companies and crowd workers [14]. Most communication is done through online discussion forums in SC. During crowdsourcing processes, registered workers could post task-related doubts and questions in the forum and get responses. Forum-based communication is less effective than real time or in-person communication taken place in traditional in-house teams. **Causes / Consequences**: One possible cause is the lack of clear communication between clients and workers [9, 31]. The second possible cause is the bad coordination and lack of co-pilots to coordinate workers [20, 33, 35, 36]. Coordination is important to ensure the timely execution of tasks and achieve the ultimate goal [34]. It is a challenge to coordinate a large number of crowd workers to work together effectively while minimizing the information losses during crowdsourcing process [23, 24]. For large projects, answering a quantity of questions in the forum is time-consuming [19]. Without timely reply, questioners may lose their patience and leave the task, which would not only lead to communication risk, but also cause other risks later on [7]. Third, some platforms provide poor usability, which cause a lack of communication tools or ineffective communication [22]. Last but not least, language barriers [14, 16] and culture differences can also cause this risk. Generally speaking, the language used on crowdsourcing platforms is English, which result in non-native English-speaking workers encountering obstacles in understanding tasks and communicating in task forums [13]. In addition, workers from all over the world may face cultural differences in understanding the context of the task [18].

4) **Schedule**. It refers to schedule delays that may occur due to indirect control over external crowdsourcing processes. **Causes / Consequences**: One possible cause is that in-house software development may be delayed by waiting for crowdsourcing submissions. Another possible cause is that the task may receive low quality or unsatisfactory submissions, or even worth, no submissions, which will lead to quality risk and starvation risk, respectively. Such tasks need to be reposted, which then causes schedule delay [7].

5) **Intellectual Property (IP).** It refers to IP losses caused by submission ownership disputes, code plagiarism, and malicious submissions. **Causes / Consequences**: One possible cause are disputes on IP. In SC, the ownership of submissions belongs to the requester, which is unclear to some developers. Without specifying the ownership explicitly may lead to IP risk [9, 10]. IP "leakage" and the consequent loss of competitive advantage are challenges in

Process Entity Level		
Design Phase	Turbulence	• Hesitation or resistance among employees and managers
	Decomposition	• How to decompose a high-level project into several atomic tasks • High inter-dependency among tasks • Complex mini tasks • Low product architecture maturity
Engagement Phase	Communication	• Lack of clear communication • Lack of copilot to coordinate workers • Coordinate many workers to work together is hard for big project • Untimely reply in the forum • Lack of communication tools and ineffective communication • Language barrier and culture difference
	Schedule	• In-house development delay for waiting crowdsourcing submissions • Low quality submission task and starvation task need to be reposted
Commitment Phase	Intellectual Property	• Ownership disputes • Code plagiarism • Malicious or harmful code submission

Table 3: Risks related to process entity.

crowdsourcing [33,35]. Another possible cause is the code plagiarize [28-30]. Untrustworthy workers plagiarize codes or designs online and apply them to their submission. The third possible cause is malicious code submissions [17, 40]. If the client adopts harmful submissions, it will cause IP confusion in the future.

Risks Related to Quality Entity

As shown in Table 4, there are five risks related to quality entity, i.e., the risk of requirement, estimation, quality, starvation, and integration.

6) Requirement. It refers to the situation in which workers have difficulty in understanding the requirement or even give up the task due to incomplete, unclear or even incorrect requirements. Poor designing of tasks also impacts the quality of crowdsourcing tasks [22]. **Causes / Consequences**: One possible cause is the unclear, complex or obscure requirement, resulting in difficulty for workers to understand and complete tasks [15]. To make matters worse, ambiguous or incorrect requirement would lead to task failure [31]. Second, without necessary documents [21], diagrams, and links, workers may have difficulty in understanding the requirement context, resulting in low submission quality or high exit rate [14].

7) **Estimation**. It refers to an unreasonable estimate of time duration and award. Selecting the right interval from the date of advertising the task till the date of submission is a real big task [7]. It is a challenging issue for clients to justify prices and employ appropriate incentivizing strategies across different tasks [41]. **Causes / Consequences:** On the one hand, short time duration and low awards make workers less motivated to complete tasks [22]. Developers without proper motivation may not be able to make consistent contributions [12,42]. On the other hand, long time duration and high awards can certainly attract workers, but it may cause schedule delays and money loss of clients, and even face failure [12,36,42].

8) **Quality.** It refers to the poor submission quality caused by workers' bad habits, incompetence, and so on. **Causes / Consequences:** One possible cause is non-adherence to best practices [10] and lack of proper coding guidelines [9]. The second possible cause is inadequate, inexperience, and inability workers, who are not suitable for performing tasks [25, 29, 31, 32, 36]. Third, if the diversity and decomposition of workers mismatch with requirements, the output will be unsatisfactory [22]. Sometimes clients tend to accept faulty code because they do not want to prevent workers from future tasks [33, 35]. This is also a possible cause of quality risk.

9) **Starvation**. It refers to the task starvation caused by no submissions. Sometimes a task has many registrants, but the number of submissions is zero, leading to the risk of task starvation [32, 36]. As a result, the task fails and needs to be reposted. **Causes / Consequences**: Crowd workers generally have no obligation to complete and submit for tasks they signed up. Some workers may be overwhelmed by other works and therefore will not submit registered tasks, which corresponds to a high task exit rate [8]. Some workers may not have a sense of responsibility and are only satisfied with the thrill of registering, so they also submit nothing. On the one hand, newcomers may attempt to register several tasks. They are not familiar with crowdsourcing platforms, and even new to the concept of SC. Their registration may be out of curiosity, but they did not submit the task due to lack of ability, unfamiliar operation, or no intention to complete. Relevant research shows that the "onboarding process" (that is, the process of trying and completing crowdsourcing tasks for the first time) is challenging for newcomers [13,21]. On the other hand, spammers may maliciously register a large number of tasks, and even write a program to automatically register all newly released tasks, but they will not complete any of them. Newcomers and malicious registrants have never submitted any solutions before, and this time they will largely not.

10) **Integration.** It refers to incompatibility problems when merging worker's code with the code developed in-house [36]. **Causes / Consequences**: One possible cause is the different development process between crowdsourcing platforms and clients [35]. Generally, platforms use a waterfall development process while clients use an agile development process. The waterfall development process

Quality Entity Level		
Design Phase	Requirement	• Lack of necessary documents or diagrams • Unclear, complex or obscure requirement • Ambiguous or incorrect requirement • Scarce requirement context
	Estimation	• Lack of estimation support • Unreasonable duration • Unreasonable award
Engagement Phase	Quality	• Non-adherence to code best practice • Lack of proper coding guidelines and directions • Inadequate, inexperienced and incompetent workers • Diversity and decomposition of workers mismatch with requirement • Company sometimes accept faulty submission for that they didn't want to deter workers from future tasks.
	Starvation	• Not submit on time • No obligation to submit • Lack of contributors
Commitment Phase	Integration	• Code version diversity • Development process difference • Integrate submissions provided by multiple workers may lead to incompatibility problem

Table 4: Risks related to quality entity.

may cause problems during the engagement process accumulated until the final solution, while the agile development process could solve problems at any time. Therefore, the development process for workers on the platform is usually different from that of the client, which is more likely to cause integration risk. Another possible cause is the diversity of code version between workers and clients. When publishing tasks on the platform, the requester gives the necessary code, and workers will continue programming based on this previous code. At the same time, internal employees may continue to develop it, which may lead to different code versions between the client and workers. When integrating the content submitted by winners, it may cause incompatibility issues owing to diversity in code version [33].

Risks Related to Delegation Entity

As shown in Table 5, three risks are related to delegation entity, i.e., the risk of worker continuity, worker engagement, and worker trustworthiness.

11) **Worker Continuity**. It refers to the low continuity associated with a lack of control over crowdsourcing workers [33]. Typically, there are multiple interrelated tasks under one crowdsourcing project. The term "continuity" refers to the situation where the same worker completes multiple tasks on the same project. As defined in the COCOMO II model, developer continuity corresponds to the PCON cost driver, and the higher the driver rating is, the more beneficial the development productivity and quality is [46]. **Causes / Consequences:** For different tasks in the same project, compared with new registrants, continuous workers tend to have stronger motivation and higher probability to submit tasks. Besides, workers who are continuously participating in the same project will spend less effort due to fewer context switches, so they can better accomplish tasks in the project. Thus, lower worker continuity generally corresponds to higher levels of risk [35].

12) **Worker Engagement.** It refers to risk caused by the lack of transparency over workers' progress. As reported in an earlier research report, there is an overall extremely high worker-quitting rate of 82.9% on Topcoder platform [8], which led to about 15.7% task failure rate due to no submission by end of tasks. **Causes / Consequences**: First, the incredibility, unreliability, and anonymity of crowds make tasks facing risks [14,26,27]. Some workers may sacrifice and minimize their efforts when they know a better way to finish the task perfectly [18]. If workers use a simple but awful method, it will result in low submission quality and difficult maintenance. Second, workers may collude with others. Some workers form a group to register several tasks together in order to increase competition level and prevent others from registering. However, each colluding group may only submit one or no submissions, which reduces the possibility of successfully task completion. This phenomenon is called "collusion" [45]. Finally, high rating workers employ strategies to increase their chances of winning in a task competition. They tend to register strategically and submit quickly, which puts a lot of psychological pressure on other workers who want to participate, so they give up registration or give up tasks. This phenomenon is called "cheap talk" [44].

13) **Worker Trustworthiness**. It refers to security of companies and data security issues caused by untrusted workers. **Causes / Consequences**: One possible cause is malicious attacks and sensitive information leakage. Registered workers can access the necessary internal code of companies. Such access may allow them to pass through the firewall legally and then access sensitive information. To make matters worse, sensitive information may be leaked out [10, 40] and attackers may gain access to the internal resources [11]. All of these situations can pose a threat to internal security [22].

Suggestions to Requesting Clients and Crowdsourcing Platforms

Based on the literature review, we derive a taxonomy of SC risk and then summarize some suggestions for requesting clients and crowdsourcing platforms in Table 6 to better organize the SC process. Each risk name corresponds to it in section risks related to process entity.

Conclusion and Future Work

This paper carried out a literature review to examine risk characteristics existing in 36 relevant literature. A taxonomy of 13 SC risk was then introduced with respect to three phases and three influence entities of SC. Besides, we provided some risk management suggestions for crowdsourcing companies and platforms, respectively. We hope that this paper can help companies and crowdsourcing platforms to better understand, anticipate, and mitigate risk in SC ahead of time, so as to achieve the expected benefits of SC.

Nonetheless, there are several limitations. On one hand, the literature database we searched is limited. Future research can include Springer, Elsevier and other literature libraries into the review scope. On the other hand, the proposed risk taxonomy is introduced based on SC phases and influence entities for easy organization and understanding across different players of SC. There might exist other way for organizing risk. And there may be other SC processes different from the competitive and collaborative models considered in this study.

Future works of this study consists of further evaluation of the proposed risk taxonomy to address these limitations through broader scope of quantitative and qualitative studies such as questionnaire, interview, and case studies.

Delegation Entity Level		
Engagement Phase	Worker Continuity	• Less continuous participants
	Worker Engagement	• Finishing the task by an easy but awful way • Collusion • Cheap talk
	Worker Trustworthiness	• Attackers will gain access to internal knowledge of firms • Sensitive information leakage • Untrustworthy workers pose a threat to network security and data security • Incredibility, unreliability and anonymity of crowds

Table 5: Risks related to delegation entity

	Suggestions	
Turbulence	• State the rationale of crowdsourcing and ensure that it is aligned with its business strategies and objectives [29].	• Provide instructions and guidance for newly companies.
Decomposition	• Decompose the complex task into several simple mini tasks [15]. • Divide and describe tasks due to high granularity [21].	• Provide guidance on task decomposition.
Communication	• Foster an open and supportive communication channel [9]. • Foster efficient crowdsourcing mechanisms [24]. • Appoint employees familiar with the project as copilots to coordinate registrants and answer questions [33]. • Focus on continuous communication, timely reply, and coordination [33].	• Implement computer-assisted translation or auto-translation mechanisms [13, 15]. • Improve the construction of task forums.
Schedule	• Pay attention to the progress of crowdsourcing tasks, timely modify and repost the failed tasks.	• Employ empirically machine learning techniques to identify risk tasks automatically.
Intellectual Property	• Explicitly assign all intellectual property to clients and reject to accept submissions containing proprietary information belonging to a third party [9, 30]. • Reinforce the examination of submissions and try to find out plagiarism code [30]. • Using crowdsourcing among the existing employees or through a hybrid crowd comprising of employees and the general public [19].	• Explicitly assign all intellectual property to the requesting company [9]. • Calling crowd workers to support originality, not copying others' results [11]. • Ban those plagiarized workers from registering tasks several days or punish them with points.
Requirement	• Provide clear, complete and consistent requirements [13]. • Provide enough links, documents and diagrams to keep context completely.	• Remind requesters to upload links & diagrams.
Estimation	• Determine the duration and award based on similar tasks.	• Provide auto-estimation support.
Quality	• Use mechanisms such as automated quality control or entry evaluation of participants [9]. • Attracting more and higher skillful developers to participate [25].	• Formulate coding guidelines and directions [10]. • Show code best practice. • Provide task-worker recommendation mechanism.
Starvation	• Provide enough incentive to attract workers [10]. • Send emails to registrants to remind and encourage them to finish tasks.	• Encourage newcomers and guide them to complete the first task. • Impose penalties on spammers.
Worker Continuity	• Send emails to the previous submitters to attract them to register new tasks in the same project.	• Provide more incentives to continuous workers, e.g., faster upgrades.
Worker Engagement	• Use a SC decision support /recommendation system to better identify appropriate resources and increase task submission rates.	• Provide more effective services to monitor ongoing data. • Provide more actionable analytic for managing different types of workers.
Worker Trustworthiness	• Restrict access by login type and strengthen the construction of firewalls and monitoring mechanisms [9,11]. • Anonymizing the data so that it becomes more neutral when released to the crowd [9].	• Restrict access by login type and strengthen the construction of firewalls and monitoring mechanisms [9,11]. • Anonymizing the data so that it becomes more neutral when released to the crowd [9].

Table 6: Suggestions to requesting clients and crowdsourcing platforms

Competing Interests

The author declare that there is no competing interests regarding the publication of this article.

References

1. Howe J (2006) The rise of crowdsourcing. Wired magazine 14: 1-4.
2. Lakhani KR, Garvin DA, Lonstein E (2010) Topcoder (a): Developing software through crowdsourcing. Harvard Business School General Management Unit Case 610-032.
3. Kittur A (2010) Crowdsourcing, collaboration and creativity. XRDS: crossroads, the ACM magazine for students 17: 22-26.
4. Savage N (2012) Gaining wisdom from crowds. Communications of the ACM 55: 13-15.
5. Bonabeau E (2009) Decisions 2.0: The power of collective intelligence. MIT Sloan management review 50: 45.
6. Schenk E, Guittard C (2009) Crowdsourcing: What can be Outsourced to the Crowd, and Why. In Workshop on open source innovation, Strasbourg, France 72: 3.
7. Hasteer N, Nazir N, Bansal A, Murthy BK (2016) Crowdsourcing software development: Many benefits many concerns. Procedia Computer Science 78: 48-54.
8. Yang Y, Karim MR, Saremi R, Ruhe G (2016) Who should take this task? Dynamic decision support for crowd workers. In Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 1-10.
9. Gebert M (2014) Crowdsourcing and risk-management: a survey-based approach (Doctoral dissertation, University of South Wales).

10. Dwarakanath A, Shrikanth NC, Abhinav K, Kass A (2016) Trustworthiness in enterprise crowdsourcing: a taxonomy & evidence from data. In Proceedings of the 38th International Conference on Software Engineering Companion, (ICSE-C), Austin, TX, pp. 41-50.
11. Kannangara SN, Uguccioni P (2013) Risk management in crowdsourcing-based business ecosystems. Technology Innovation Management Review 3: 32-38.
12. Mao K, Capra L, Harman M, Jia, Y (2017) A survey of the use of crowdsourcing in software engineering. Journal of Systems and Software 126: 57-84.
13. Zanatta AL, Steinmacher I, Machado LS, de Souza CRB, Prikladnicki R (2017) Barriers faced by newcomers to software-crowdsourcing projects. IEEE Software 34: 37-43.
14. Machado L, Kroll J, Marczak S, Prikladnicki R (2016) Breaking collaboration barriers through communication practices in software crowdsourcing. In 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE), Irvine, CA, pp. 44-48.
15. Zanatta AL, Machado L, Steinmacher I (2018) Competence, collaboration, and time management: Barriers and recommendations for crowdworkers. In 2018 IEEE/ACM 5th International Workshop on Crowd Sourcing in Software Engineering (CSI-SE), Gothenburg, Sweden, pp. 9-16.
16. Stol KJ, LaToza TD, Bird C (2017) Crowdsourcing for software engineering. IEEE software 34: 30-36.
17. Yu H, Shen Z, Miao C, An B (2012) Challenges and opportunities for trust management in crowdsourcing. In 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Macau, pp. 486-493.
18. Kittur A, Nickerson JV, Bernstein M, Gerber EM, Shaw A, et al. (2013) The future of crowd work. In Proceedings of the 2013 conference on Computer supported cooperative work, pp. 1301-1318.
19. Dwarakanath A, Chintala U, Shrikanth NC, et al. (2015) Crowd build: A methodology for enterprise software development using crowdsourcing. In 2015 IEEE/ACM 2nd International Workshop on CrowdSourcing in Software Engineering, Florence, pp. 8-14.
20. LaToza TD, Van Der Hoek A (2015) IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Montreal, QC, Canada, pp. 301-301.
21. Machado L, Zanatta A, Marczack S, Prikladnicki R (2017) The good, the bad and the ugly: an onboard journey in software crowdsourcing competitive model. In 2017 IEEE/ACM 4th International Workshop on CrowdSourcing in Software Engineering (CSI-SE), Buenos Aires, pp. 2-8.
22. Farid S, Safdar Z, Ahmed MU (2018) Prioritizing Critical Challenges for the Quality of Crowdsource Software Products. Journal of Information Communication Technologies and Robotic Applications, pp. 11-17.
23. Niu X, Qin S (2017) A review of crowdsourcing technology for product design and development. In 2017 23rd International Conference on Automation and Computing (ICAC), Huddersfield, pp. 1-6.
24. Sherief N, Jiang N, Hosseini M, Phalp K, Ali R (2014) Crowdsourcing software evaluation. In proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, pp. 1-4.
25. Li K, Xiao J, Wang Y, Wang Q (2013) Analysis of the key factors for software quality in crowdsourcing development: An empirical study on topcoder. com. In 2013 IEEE 37th Annual Computer Software and Applications Conference, Kyoto, pp. 812-817.
26. Sanagavarapu LM, Reddy YR (2018) Crowdsourcing Security-Opportunities and Challenges. In 2018 IEEE/ACM 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), Gothenburg, pp. 37-40.
27. Abhinav K, Shrikanth NC, Dwarakanath A (2015) Simulation of Consensus Based Approaches to Mitigate the Challenges in Crowdsourcing. In QuASoQ/WAWSE/CMCE@ APSEC, pp. 49-51.
28. Kocsis D, de Vreede GJ (2016) Towards a taxonomy of ethical considerations in crowdsourcing.
29. Kamoun F, Alhadidi D, Maamar Z (2015) Weaving Risk Identification into Crowdsourcing Lifecycle. Procedia Computer Science 56: 41-48.
30. Liberstein MA, Tucker A, Yankovsky AK (2012) Crowdsourcing: Understanding the risks. NYSBA Inside 30: 34-38.
31. Liu S, Xia F, Zhang J, Pan W, Zhang Y (2016) Exploring the trends, characteristic antecedents, and performance consequences of crowdsourcing project risks. International Journal of Project Management 34: 1625-1637.
32. Saremi R (2018) A hybrid simulation model for crowdsourced software development. IEEE/ACM 5th International Workshop on Crowd Sourcing in Software Engineering (CSI-SE), Gothenburg, Sweden, pp. 28-29.
33. Stol KJ, Fitzgerald B (2014) Two's company, three's a crowd: a case study of crowdsourcing software development. In Proceedings of the 36th International Conference on Software Engineering, pp. 187-198.
34. Stol KJ, Fitzgerald B (2014) (2014) Research Protocol for a Case Study of Crowdsourcing Software Development, Lero Technical Report Lero-TR-2014-03, Working Paper, (7 pp), Limerick: University of Limerick.
35. Stol KJ, Fitzgerald B (2014) Researching crowdsourcing software development: perspectives and concerns. In Proceedings of the 1st International Workshop on CrowdSourcing in Software Engineering (pp. 7-10).
36. Suganthy A, Chithralekha T (2016) Application of crowdsourcing in software development," International Conference on Recent Trends in Information Technology (ICRTIT), Chennai, pp. 1-6.
37. Saxton GD, Oh O, Kishore R (2013) Rules of crowdsourcing: Models, issues, and systems of control (2011). Information Systems Management 30: 2-20.
38. Law E, Gajos KZ, Wiggins A, Gray ML, Williams A (2017) Crowdsourcing as a tool for research: Implications of uncertainty. In Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, pp. 1544-1561.
39. Leimeister JM, Zogaj S, Durward D (2015) New Forms of Employment and IT: Crowdsourcing. In: Blanpain R, Hendrickx F & Waas B (Eds.), Bulletin of Comparative Labour Relations Series(Vol.New Forms of Employment in Europe, pp.23 -41).Kluwer Law International BV, The Netherlands.
40. Abhinav K, Dwarakanath A, Singh P (2015) Trustworthiness in crowdsourcing (Doctoral dissertation).
41. Yang Y, Saremi R (2015) "Award vs. Worker Behaviors in Competitive Crowdsourcing Tasks," 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Beijing, pp. 1-10.
42. Mao K, Yang Y, Li M, Harman M (2013) "Pricing crowdsourcing-based software development tasks," 35th International Conference on Software Engineering (ICSE), San Francisco, CA, 2013, pp. 1205-1208.
43. Gefen D, Gefen G, Carmel E (2016) How project description length and expected duration affect bidding and project success in crowdsourcing software development. Journal of Systems and Software 116: 75-84.
44. Archak N (2010) Money, glory and cheap talk: analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on TopCoder. com. In Proceedings of the 19th international conference on World wide web, pp. 21-30.
45. KhudaBukhsh AR, Carbonell JG, Jansen PJ (2014) Detecting non-adversarial collusion in crowdsourcing. In Second AAAI Conference on Human Computation and Crowdsourcing.
46. Barry W. Boehm, Clark, Horowitz, Brown, Reifer, Chulani, Ray Madachy, and Bert Steece (2000) Software Cost Estimation with Cocomo II with Cdrom (1st. ed.). Prentice Hall PTR, USA.