frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 12 June 2024
DOI 10.3389/fcomp.2024.1400750

OPEN ACCESS

EDITED BY
Ahmad Nauman Ghazi,
Blekinge Institute of Technology, Sweden

REVIEWED BY
Zainab Yousuf,
Bahria University, Pakistan
Rubia Fatima,
Emerson University Multan, Pakistan
Farzana Jabeen,
National University of Sciences and Technology (NUST), Pakistan
Monica Rossi,
Polytechnic University of Milan, Italy

*CORRESPONDENCE
Shamaila Qayyum
✉ shamaillla@gmail.com

RECEIVED 14 March 2024
ACCEPTED 30 May 2024
PUBLISHED 12 June 2024

CITATION
Qayyum S, Imtiaz S, Hayat Khan H, Almadhor A and Karovic V (2024) Working with agile and crowd: human factors identified from the industry. *Front. Comput. Sci.* 6:1400750. doi: 10.3389/fcomp.2024.1400750

Working with agile and crowd: human factors identified from the industry

Shamaila Qayyum[1]*, Salma Imtiaz[1], Huma Hayat Khan[2], Ahmad Almadhor[3] and Vincent Karovic[4]

[1]Department of Software Engineering, International Islamic University, Islamabad, Pakistan, [2]Department of Software Engineering, National University of Modern Languages, Islamabad, Pakistan, [3]Department of Computer Engineering and Networks, College of Computer and Information Sciences, Jouf University, Sakaka, Saudi Arabia, [4]Department of Information Management and Business Systems, Faculty of Management, Comenius University Bratislava, Bratislava, Slovakia

Introduction: Crowdsourcing software development (CSSD) is an emerging technique in software development. It helps utilize the diversified skills of people from across the world. Similar to all emerging techniques, CSSD has its own benefits and challenges. Some unique challenges arise when CSSD is used with Agile methodology. This is because many characteristics of CSSD differ from Agile principles. CSSD is a distributed approach where workers are unknown to each other, whereas Agile advocates teamness and is mostly suitable for colocated teams. Many organizations are now combining crowdsourcing software development (CSSD) and Agile methodologies, yet there is limited understanding on the implications of this integration. It is crucial to emphasize the human factors at play when implementing Agile alongside CSSD. This involves considering how teams interact, communicate, and adapt within these frameworks. By recognizing these dynamics, organizations can better navigate the complexities of integrating CSSD and Agile, ultimately fostering more efficient and collaborative development processes.

Method: This study aimed to explore the human factors involved in the integration of CSSD with Agile, by identifying the challenges that practitioners face when they follow Agile with CSSD and the strategies they follow. The study contributes by providing an in-depth understanding of a new approach, CSSD, integrated with Agile. The study also explores the challenges faced by practitioners that are not already enlisted.

Results and discussion: These identified challenges are grouped into six different categories, which are trust-related challenges, coordination and communication challenges, organizational challenges, task-related challenges, project-related challenges, and some general challenges. Strategies for each of these categories of challenges are also identified. The list of challenges and strategies identified in this study can be helpful in further research on CSSD and Agile integration. The practitioners can also follow these strategies to reduce the impact of challenges they face while they perform CSSD along with Agile.

KEYWORDS

agile methodology, crowdsource software development, human factors, challenges, strategies, industry practitioners

1 Introduction

The software development industry is growing tremendously, which results in a shift from traditional in-house development to distributed software development (DSD) (Stol et al., 2017b). However, software engineering is a human activity more than a technical activity (Capretz, 2014; Siegmund, 2024). Previously, software houses used a traditional in-house approach where the colocated team was physically present at the site of work. This approach offers advantages such as control, immediate collaboration, and direct resource access (Colomo-Palacios et al., 2014). However, scalability issues and restricted access to diverse skill sets are its limitations. These limitations affect organizations' ability to cope with evolving market demands and emerging technologies (Colomo-Palacios et al., 2014). To overcome these challenges, organizations have increasingly turned to distributed software development methodologies, such as global software development (GSD) (Jabangwe et al., 2016) and offshoring (Mukherjee et al., 2023). GSD and offshoring have the benefits such as cost-effectiveness and access to specialized resources. Some challenges such as cultural differences, communication barriers, and coordination complexities are among the hurdles that organizations may face during GSD and offshoring (Mukherjee et al., 2023). GSD is an emerging approach for software development, where software development companies offshore or outsource development tasks to teams located across the globe. Some of the challenges of GSD are communication issues among remotely located teams (Yasin et al., 2023), coordination among teams, and team management (Shameem et al., 2015). DevOps, a methodology designed to enhance collaboration between development and operations teams, that emerged as the response to these shortcomings, too has many shortcomings (Laukkanen et al., 2017). While DevOps proves useful for collaboration and integration challenges in distributed development, a need to optimize the process is still realized by organizations. A 1 + 5 model has been proposed that helps realize business problems and cooperates with IT systems, to cater to the main focus of DevOps, which is continuous integration, delivery, and deployment (Górski, 2021a). Another great example is the node deployment packages for blockchain systems. To achieve reliable software development, a deployment framework for the automation of blockchain has been introduced that also offers UML modeling support (Górski, 2021b). A pipeline approach for solving the problem of continuous integration and continuous delivery is proposed, which helps in the deployments of various kinds of applications (Donca et al., 2022).

Crowdsourcing software development (CSSD) has emerged as a trending approach that leverages global skills through social media platforms (Asiegbu Baldwin et al., 2017). Crowdsourcing was initially defined by Howe (2006). According to him, it involves outsourcing to an unknown group of people through an open call (Howe, 2006). Crowdsourcing involves unknown people rather than companies or contractual employees, which makes it different from outsourcing (Stol et al., 2017a). There are three main components in crowdsourcing, requester worker, and platform. Requester makes requests through an online platform to which workers reply and get the job after completing some task. However Howe (2006) defines crowdsourcing as follows:

> *"Crowdsourcing is taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally a large group of people in the form of an open call."*

CSSD enables organizations to find diverse talent and expertise for the swift development of complex problems (Ruhe and Wohlin, 2014; Khan et al., 2019b), and this is enabled using online platforms (Mao et al., 2017). CSSD is known for the development of open-source software. Some projects such as Linux, Mozilla Firefox, and WordPress are good examples of it (Moslehi et al., 2016).

In addition to these approaches, Agile is also widely used in organizations. Agile has many benefits as quick delivery of service, teamness, and adaptation to change. Agile is mostly considered suitable for collocated teams as it emphasizes team building and collaboration among teams. Agile is a lightweight process and is known for its frequent delivery of services in a flexible manner (Mao et al., 2017). The underlying benefits of ASD include increased communication, coordination, trust, and support among team members, self-organized teams, and face-to-face communication (Erich et al., 2017). Interestingly, Agile is also followed by many software development companies that contribute to global software development (GSD) (Agerfalk et al., 2005; Niazi et al., 2016).

Despite having contradicting characteristics of these approaches, Agile software development is believed to have solved many challenges of GSD (Beecham et al., 2021). Specifically, Scrum is beneficial for reducing GSD barriers (Alsahli et al., 2017). In a geographically located approach, like GSD, daily scrum meetings are very helpful in managing communication and coordination among teams (Kausar et al., 2021), team productivity, trust between team members (Verwijs and Russo, 2023), team involvement, project management (Prasetio et al., 2021), and prioritizing customers' requirements (Khan et al., 2019a). Scrum of Scrums can reduce coordination, temporal, and other geographical challenges (Kausar et al., 2021).

Lately, it has been reported that various software companies following ASD also adopt the CSSD approach to get their projects developed from the crowd (Dwarakanath et al., 2015). Requirements gathering via crowd is also practiced (Khan J. A. et al., 2022). It has been developed in research that the integration of these two approaches, CSSD and ASD, is trivial. However, both approaches have contradicting characteristics, which makes it challenging for practitioners to follow (Stol et al., 2017a). Stol (Agerfalk et al., 2005) emphasized the effective integration of Agile software development and CSSD. Literature shows that software industries follow both ASD and CSSD; however, there is scarce literature available on the integration of ASD and CSSD (Agerfalk et al., 2005).

Research on the effective integration of ASD and CSSD is still novel and scarce. CSSD being a distributed approach has many characteristics that contradict the principles and practices of Agile methodology. Agile advocates face-to-face communication, contradicting the practice in CSSD, where workers do not know each other. Because of the heterogeneous nature of CSSD, team management and coordination also become challenging, which are otherwise strongly recommended in Agile. Thus, it is challenging

for software development companies to follow crowdsourcing while working in an Agile environment (Qayyum et al., 2020). It is deemed important to understand the challenges that are faced by the software industry practitioners that follow ASD and CSSD together (Agerfalk et al., 2005). It is known that human factors play a great role in software engineering (Lenberg et al., 2015; Siegmund, 2024). Some factors such as organizational change, communication, and involvement in solutions are some of the challenges in software engineering, as software development is a team activity, so the challenges faced by the team can impact the software (Lenberg et al., 2015). As the integration of ASD and CSSD creates some challenges for the people involved in software development, understanding their challenges and providing strategies for them is important for effective software development (Qayyum et al., 2020).

Despite having scarce literature on the integration of these two approaches, the integration of ASD in GSD is a great motivation for researchers as ASD is believed to overcome many shortcomings of GSD (Beecham et al., 2014). Certainly, CSSD and GSD are different approaches, especially regarding team formation. In GSD, there are designated teams, whereas CSSD workers are not designated employees of the organization; rather, they are a large group of unknown people who do not know each other (Li et al., 2015). Agile approaches help in successful software development by emphasizing the needs and challenges of people involved in software development. Communication is one of the greatest human factors in software engineering that affects the software engineering process (Lenberg et al., 2015), and agile emphasizes the communication and collaboration of people involved for effective software development (Barros et al., 2024). However, if Agile practices benefit GSD, they can be integrated and used during CSSD for effective software development. The research intends to find the human factors in Crowd–Agile development. These are the challenges faced by industry practitioners who follow Agile and CSSD. Research also aimed to find strategies to solve and reduce these challenges. This research contributes valuable insights to industry practitioners, researchers, and organizations in various ways. This study contributes to the existing body of knowledge by providing knowledge about the challenges of Crowd–Agile development. Industry practitioners can follow this research to learn how they can overcome the challenges to effectively develop software. This leads to the formulation of two research questions:

RQ1: What challenges are faced by industry practitioners working in an Agile–Crowd environment?
RQ2: What strategies are used to overcome the challenges of an Agile–Crowd environment?

The rest of the article is structured as follows. Section 2 covers the related work. Section 3 contains the proposed methodology. Section 4 contains survey design and execution. Section 5 covers the survey analysis. Statistical analysis of the survey is presented in Section 6. Research findings are discussed in Section 7. Section 8 covers the discussion and limitations of this research, and Section 9 concludes the manuscript.

2 Related work

2.1 Crowdsourcing software development

Crowdsourcing software development is a kind of outsourcing where employers assign tasks to a group of large, unknown people through online platforms (Mao et al., 2017). Crowdsourcing helps organizations to scale their productivity (Stol and Fitzgerald, 2014a). Many organizations have started using crowdsourcing approaches to improve their services (Mao et al., 2017). Crowdsourcing software development is usually carried out around three actors. The first actor is the requester (or employer), who posts a task and makes an open call to the crowd for a job. A requester is the person or organization paying for the tasks (Stol and Fitzgerald, 2014a; Mao et al., 2017). The second actor is the crowd worker who participates in the job. A worker is usually the person who gets paid for completing a task (Stol and Fitzgerald, 2014a; Mao et al., 2017). In crowdsourcing software development, crowd workers complete the portions of software development assigned to them (Stol and Fitzgerald, 2014a). The third actor is the online platform (or social media marketplaces), which helps employers and crowd workers to meet (Mao et al., 2017). These online marketplaces serve as a meet-up platform for requesters and workers. There are many platforms used for crowdsourcing software development such as Amazon Mechanical Turk, Topcoder, and Stack Overflow. Topcoder is considered the largest platform for crowdsourcing software development (Stol and Fitzgerald, 2014a). Crowdsourcing and machine learning are also used in RE (Ali Khan et al., 2020).

2.2 Challenges of CSSD

Crowdsourcing software development faces many challenges. Some of the key concerns of CSSD are task decomposition (Zhen et al., 2021), communication and coordination, planning, scheduling, quality assurance, knowledge, motivation (Stol and Fitzgerald, 2014b), developing a volunteer network, trust (Bhatti et al., 2020), and team development (Hosseini et al., 2014). Many tasks that are complex need to be broken down for assignment and resource allocation (Stol and Fitzgerald, 2014a; Mao et al., 2017). Effective task decomposition is a challenging task. A good task decomposition helps in utilizing a large pool of human resources. The difficulty in decomposing tasks with a crowd perspective arises from assumptions, interfaces, and dependencies (Stol and Fitzgerald, 2014a). Managing these dependencies needs proper coordination among crowd workers and managers. Crowdsourcing has a relatively more extensive turnover rate of workers, so managing expertise and intellectual property becomes a challenge (Stol and Fitzgerald, 2014a). A crowd can be any group of a large number of anonymous people composed of experts, fresh graduates, and inexperienced people. Crowdsourcing helps utilize the skills of variant individuals in a single project (Beretta et al., 2021). This is a way of getting the work done by many experienced workers, which was previously done by few workers.

2.3 Agile software development

Companies are widely adopting Agile software development methodology (Li et al., 2015). Interestingly, some of the characteristics faced as a challenge by CSSD are the benefits achieved while following ASD. Agile keeps the team productive and motivated and provides quality products (Alsahli et al., 2017). ASD emphasizes face-to-face communication (Hamilton and Holler, 2023) and works in iteration with the close collaboration of the team, project manager, and business people (Srivastava et al., 2017). Trust development among team members is another strength of ASD (Tyagi et al., 2022). A daily meeting is held to discuss the progress of the project (Alsahli et al., 2017). ASD also helps lower the project cost as there is increased communication, which decreases the chances of rework and therefore cost overrun and project delay (Bowes, 2015). The main problem is executing CSSD in an Agile environment (Li et al., 2015). CSSD with ASD is emerging, and there is scarce literature on integrating the two. CSSD has many characteristics similar to those of GSD as distributed teams, geographical differences, time zone differences, and socio-economic differences. GSD is already implemented with ASD in many organizations. To understand the integration of CSSD and ASD, this research first explains GSD and its integration with ASD.

2.4 Global software development

GSD focuses on developing software with distant teams. This recent software development trend is becoming a favorite approach of researchers and developers (Khan R. A. et al., 2022). It does not only help in reducing overall project costs but also increases overall performance (Ojha and Chaudhary, 2022). Carmel suggests that one of the important reasons for following GSD is the limited availability of expertise required for a project at a site. Cost-effectiveness is yet another reason for developing the software across the globe. Along with these reasons, it sometimes requires hiring of expertise that may be locality-specific. One of the biggest benefits that can be achieved by GSD is "round-the-clock" development (Ilyas et al., 2024). However, developing software globally is not an easy task and has many challenges and limitations. The shortcomings of GSD are the problems of infrastructure as the teams are located at geographically distant sites, network connectivity among different teams, environment for different teams, build-up testing, and change management (Al-Saqqa et al., 2020; Yasin et al., 2023).

With the tremendous growth of dependable software and continuous requirement volatility, software development initially faced many challenges. In such situations, a different approach to traditional software development proved very beneficial, known as Agile development. Agility refers to the flexibility of continuously addressing change (Al-Saqqa et al., 2020). Agile is a lightweight process widely adopted by software industries (Rasnacis and Berzisa, 2017). The Agile method offers promising advantages for rapid changes in software development (Al-Saqqa et al., 2020). The Agile Manifesto states that Agile prefers individuals over processes, working software over documentation, collaboration over contracts, and changes over following a plan (Al-Saqqa et al., 2020).

The Agile Manifesto is based on 13 Agile principles. These principles focus on customers' satisfaction through frequent and early software delivery while continuously accepting changes. Agile also focuses on keeping the clients on-site for feedback and discussion and having a face-to-face communication among people. Agile prefers individual skills, trust among them, and self-organization and emphasizes providing team members the support they need. Agile also focuses on working software rather than extensive design, while arguing to keep things simple (Al-Saqqa et al., 2020).

2.5 Agile global software development

Ågerfalk (2006) argues that when agile is used in global software development (GSD), Agile tends to minimize many challenges of GSD. Extreme programming tends to be very helpful when distributed Agile teams can pair themselves while adjusting their time zones. Pair programming improves sharing among different teams. Time-shifting patterns among globally distributed teams also help in reducing temporal distance (where two teams find it difficult to communicate due to differences in their time zones). Amitoj provides a list of eight factors that assure the applicability of Agile in a distributed environment (Singh et al., 2015). These factors are schedule and process management, techniques, communication and collaboration, risk and resource management, tools, users' adaptability, organizational culture, and financial and temporal aspects.

Crowdsourcing software development also involves workers from distant locations (Howe, 2006). With crowd workers from distant locations, following Agile practices can be challenging for CSSD as it is in GSD. It is known that human dimensions play a critical role in the success of software projects (Meier and Kock, 2023). Researchers have started working on the integration of agile and crowdsourcing (Qayyum et al., 2020). In our previous study (Qayyum et al., 2023), we have conducted a systematic literature review (SLR) to find out the challenges of Agile–Crowd development. The systematic literature review (SLR) was conducted through a series of methodical steps. Initially, the focus was on understanding the challenges associated with crowdsource software development (CSSD). CSSD shares similarities with global software development (GSD) as both of them are distributed in nature, prompting an examination of challenges faced in Agile global software development (AGSD). Given the inherent contradictions between Agile principles and GSD characteristics, identifying the specific challenges of AGSD was imperative. Subsequently, these challenges from both CSSD and AGSD were synthesized through thematic analysis to create a comprehensive list of potential obstacles that may arise when practitioners adopt Agile CSSD. This list serves as a foundational framework, which is further validated and explored in this study.

TABLE 1 List of possible challenges for crowd agile [taken from Kasunic (2005)].

Challenge category	Challenge name
Team issues	Trust issues among team/**crowd** Team/crowd organization Team/crowd performance issues Motivation and remuneration issues.
Coordination and communication issues	Less communication within the team Less communication with the customer Communication process issues Cross-team communication Communication medium issues Communication overhead
Organizational issues	Organizational difference Legal considerations Technological issues Planning and scheduling issues
Project	Configuration and version management Quality assurance Costing issues
Task	Task design Task assignment Task monitoring

A list of possible challenges adapted from this SLR (Qayyum et al., 2023) is presented in Table 1. Table 1 contains the list of challenges obtained as a result of SLR.

3 Proposed research method

This study aimed to identify the human factors involved in Crowd–Agile Software Development. These factors are the challenges faced by industry practitioners working in a Crowd–Agile setup. The study also focuses on identifying the strategies to reduce these challenges. Survey methodology is used to conduct this study. The survey has both open-ended and closed-ended questions. To design the survey instrument, existing knowledge on the possible challenges of Crowd–Agile development is used. Researchers have previously conducted a systematic literature review (SLR) to find a list of possible challenges in the Agile–Crowd integration (Qayyum et al., 2023). As there is scarce literature on the integration of Crowd–Agile, researchers have conducted SLR by studying in detail the challenges of AGSD and CSSD. The distributive nature of GSD is similar to that of CSSD, with some exceptions. This motivated researchers that many challenges faced during Agile with GSD may be faced during Crowd–Agile development. Hence, the researchers conducted SLR to find challenges of AGSD and CSSD. A list of possible challenges is produced after thematic analysis and focused coding of the challenges (Qayyum et al., 2023). It is worth mentioning that the papers selected for the SLR are according to the merit criteria of the IMT checklist (Yasin et al., 2022). However, this list is an outcome of the SLR and is considered as a possible challenge only. The authentication of these challenges is needed to be verified.

In this study, we take this opportunity to validate these challenges and identify any new challenges that are faced by practitioners working in a Crowd–Agile environment. The survey questionnaire is designed based on this list. The possible challenges from SLR are shown in Table 1. The survey is conducted to find out what challenges from this list practitioners face when using crowdsourcing while implementing the Agile software process model. The survey also identifies the strategies practitioners adopt to overcome the challenges of crowdsourcing when used with Agile. Not only the list of challenges identified from the SLR is verified during an industrial survey, but also the survey identifies some other challenges, which are not identified during SLR. The survey explores the strategies that are used by organizations to minimize the identified challenges. The survey is *exploratory* in nature.

The survey is self-administered. The targeted audience for the survey is software development practitioners and managers with experience in crowdsourcing and Agile software development. Any role of software development practitioners can fill this survey as we need to see different perspectives of the development team. The respondents are project managers, software developers, testers, designers, architects, quality assurance teams, business analysts, and system analysts.

For appropriate characterization of the targeted audience, the guidelines of Kasunic (2005) are followed. All the software development organizations across the world working with crowdsourcing and the Agile approach are contacted for the survey via email. For checking the quality and effectiveness of the questionnaire, a pilot study is conducted. This pilot study is conducted by representative organizations situated in Islamabad/Rawalpindi.

The population is hard to identify as no company on their website mentions that they follow crowdsourcing. For this reason, the snowballing technique is used. The snowballing technique is best used when the population is hard to find and the sample size is unknown. Crowdsourcing is a new term, so people are usually unaware of it too. This is another reason to use snowballing as it needs to be explained to each candidate respondent and then ask for their recommendations. However, there are some risks regarding this technique. Authors are not directly connected to the respondents and hence cannot have first-hand information from respondents. The reference given by participants can be based on their biasness; it is also possible that we have missed some key respondents due to the potential biasness of respondents.

The survey starts with acquaintances working as project managers in OSLO, UK, and Pakistan. Their company is performing crowdsourcing software development following Agile. They are requested to recommend some practitioners from their company to proceed with snowballing. Initially, five waves of snowballing were planned, but as the population is rare and there were no more recommendations, so snowballing process stopped at wave 3 after waiting for 2 months.

The survey questionnaire can be accessed with the following link; however, for the integrity of results, no new responses are accepted via this link. https://docs.google.com/forms/d/1wbLkrnLI2mzRwq5ETCo5YJJGEZ-36BhxAsjYCFNvN9s/edit.

The data and responses of the survey are provided in the following link: https://docs.google.com/spreadsheets/d/1vyRngAFDXz0GY64-wg95lZnXtiMXwxR7mmz8y0QTGg0/edit?usp$=$~sharing.

4 Survey design and execution

The purpose of this survey study is 2-fold: 1. to validate the challenges identified by the SLR study, which are already discussed in previous chapters and 2. to explore challenges not mentioned in the literature but faced by practitioners. The survey study also intends to discover the strategies practitioners follow when facing any challenges. A few strategies to overcome these challenges are identified from the SLR and are validated using the survey, and some new strategies are explored. All ethical and legal considerations are kept in mind during survey conduction. Appropriate measures are taken in this regard. Participants are informed in a disclaimer that the data will only be used for research purposes, and no identities will be disclosed. No personal information about respondents is taken during the survey. The survey has two types of questions: open-ended and closed-ended. The first section of the survey is about the demographics of the respondents. This asks about the respondent's role, experience, the methodology they follow, and platform they work with. Later on, this information is used to find different statistical analyses. In the next section, respondents are asked to mention the challenges they face. This section contains both closed-ended and open-ended questions. In Table, all challenges are categorized into five categories. Each question is related to one category. One question is about "Team-related challenges." In this category, there are four possible challenges. A Likert scale of 1–5 is given against each challenge. Respondents are asked to rank the challenge on a scale of 1–5 depending on its frequency of occurrences. The "N/A" option is also available against each challenge which means that the challenges do not exist. At the end of these four challenges, an open-ended question is given which asks the respondents to mention any other challenges that they face related to the team/crowd. Next to this question, some possible strategies for team-related issues are given and respondents are asked to select the strategies they use to overcome their team-related challenges. In the fourth part of this question, another open-ended question is given to ask for any strategies they follow which are not present in the list. Next to team-related challenges, four parts of questions on the same pattern are designed for communication and coordination-related challenges. This category has five possible challenges. This pattern is followed to design questions for other categories too, which are organizational challenges, software project-related challenges, and task-related challenges. At the end of the survey, two open-ended questions are given. One question asks about any other challenges they face, which may not link to any of the suggested categories, and the other question asks about strategies followed to overcome these challenges.

5 Survey analysis

The survey aims to identify the challenges faced by practitioners who perform crowdsource software development while staying in the Agile setup. Therefore, open-ended questions for strategies are also part of this survey. The results of the survey are analyzed using MS Excel, SPSS, and Python. Different statistical tests are run via SPSS. The Panda library of Python is used for analysis. The details are presented in this chapter.

5.1 Meta data of respondents

The survey is designed using Google Forms and is conducted online. It was started with the project managers who were involved in Agile–Crowd development. These project managers were from Pakistan and Oslo. According to the snowball technique, as we approached other participants, it was found that they were from different geographical locations. The majority of the participants are from the UK, a few from Norway, Pakistan, and Sweden. At the start of the survey, the participants are asked to confirm that they follow both Agile and CSSD. The participants who confirmed their involvement with both techniques were given access to the questionnaire. A total of 82 responses are received, out of which two responses were incomplete, and hence, they are not included. Most of these responses are received from the project manager followed by the project developers; 34.1% of respondents are project managers, 24.4% are software developers, 13.4% are testers, and 9.8% are architects. The rest are designers, scrum masters, quality engineers, system engineers, business analysts, and system analysts. Among 82 respondents, the majority of respondents have more than 8 years of experience; 42.7% of respondents have more than 8 years of experience; 25.6% of respondents have 2–4 years of experience in the industry; 22% had 5–7 years of experience; and only 9.8% of respondents had $\leq$1 year of industry experience. Table 2 shows the crosstab representing the demographics of the survey respondents.

5.2 Frequency of occurrence of challenges

The survey respondents are asked to rank each challenge according to its frequency of occurrence. Respondents are given a Likert scale of 1–5 and are asked to select a scale based on the frequency of occurrence of each challenge. In this Likert scale, 1 means that the challenges are least occurring, whereas 5 means frequently occurring challenges. The mean frequency is then calculated to find the average occurrence of each challenge. While calculating frequency, only those challenges considered that have a mean value $\geq$2.5 and, at the same time, their standard deviation does not exceed 0.5. The frequency of occurrence of these challenges for Agile–Crowdsource development can be seen in the stacked bar chart provided in Figure 1. Legend is given the right corner that shows the different colors of frequencies: extremely frequent to least frequent. To make the stacked bar chart more comprehensible, the not applicable option is excluded. The bar chart shows that quality assurance is the most frequent challenge, practitioners face when they follow CSSD with Agile. Followed by quality assurance, costing issues are the second most frequent challenge. The third most frequent challenge is the motivation and remuneration of a crowd. According to this chart, crowd performance is not reported to be extremely frequent.

The results show that quality assurance is the most frequently occurring challenge. The second most frequent issue is cross-team communication. Ignoring the least frequent and moderately frequent challenges, costing issues and motivation and remuneration of the crowd remain very frequent. Other highly

10.3389/fcomp.2024.1400750

TABLE 2 Demographics of survey respondents.

Count of 2. What is your experience	Column labels				
Row Labels	< =1 years	2–4 years	5–7 years	8+ years	Grand TOTAL
Architect	**2**		**1**	**5**	**8**
Scrum	1		1	2	4
Scrum, DSDM				1	1
Scrum, Kanban	1				1
Scrum, XP				1	1
Scrum, XP, Lean, Kanban, A mix of different agile methodologies (if yes, please mention name)				1	1
Business analyst				**1**	**1**
Scrum				1	1
Designer		**1**			**1**
A mix of different agile methodologies (if yes, please mention name)		1			1
Manager		**3**	**8**	**17**	**28**
Kanban				1	1
Scrum		1	4	8	13
Scrum, A mix of different agile methodologies (if yes, please mention name)		1		1	2
Scrum, Kanban			2	3	5
Scrum, Kanban, A mix of different agile methodologies (if yes, please mention name), Scrumban				1	1
Scrum, Lean, Kanban		1	1	1	3
Scrum, Lean, Kanban, A mix of different agile methodologies (if yes, please mention name)			1		1
Scrum, XP, Lean, Kanban				1	1
XP, Lean, Kanban, A mix of different agile methodologies (if yes, please mention name), Scrumban				1	1
No			**1**		**1**
None			1		1
Product designer		**2**			**2**
Scrum, Lean		1			1
Scrum, Lean, Kanban, Google Design Sprints		1			1
Quality engineering lead/business analyst		**1**			**1**
Scrum		1			1
Researcher				**1**	**1**
Scrum, XP, Crystal				1	1
Scrum master	**1**				**1**
XP	1				1
Scrum master, test manager				**1**	**1**
Scrum				1	1
Software developer	**4**	**8**	**5**	**3**	**20**
A mix of different agile methodologies (if yes, please mention name)			1		1
A mix of different agile methodologies (if yes, please mention name), Scrum And Kanban	1				1

(Continued)

10.3389/fcomp.2024.1400750

TABLE 2 (Continued)

Count of 2. What is your experience	Column labels				
Row Labels	< =1 years	2–4 years	5–7 years	8+ years	Grand TOTAL
Kanban			1		1
Lean				1	1
Scrum	1	5	2	1	9
Scrum, A mix of different agile methodologies (if yes, please mention name)	1	2			3
Scrum, Kanban		1		1	2
Scrum, Kanban, A mix of different agile methodologies (if yes, please mention name)			1		1
XP	1				1
Sr. executive—corporate performance				**2**	**2**
A mix of different agile methodologies (if yes, please mention name)				2	2
Student		**1**			**1**
FDD, Crystal		1			1
System analyst		**1**		**2**	**3**
Scrum		1		1	2
Scrum, Lean				1	1
system engineer		**1**			**1**
Scrum, customized process		1			1
Test Manager			**1**		**1**
Scrum, Kanban			1		1
Tester	**1**	**4**	**2**	**4**	**11**
Scrum		4		1	5
Scrum, FDD			1	1	2
Scrum, Kanban	1		1	1	3
Scrum, Lean, Kanban				1	1
Grand total	**8**	**22**	**18**	**36**	**84**

frequent challenges are communication process and cross-team communication. Quality assurance has a high frequency too.

The overall results show that the most occurring challenges for the practitioners who work in an Agile setup and perform CSSD as well are as follows: costing issues, communication process, cross-team communication, motivation and remuneration, quality assurance, scheduling and planning, communication with clients, crowd performance, and trust issues among the crowd.

Frequencies of challenges are represented in a stacked bar chart to make the findings more clearly verified. The extremely frequent challenge chart is shown in Figure 2. According to respondents, the majority of 9% of respondents have ranked "cross-team communication" as a highly frequent challenge. For 8% population, "crowd performance" is a highly frequent challenge; 7% of the population have ranked "motivation and remuneration," "communication process," and "costing issues" as very frequent, as they ranked these challenges as 3 on the ordinal scale.

The overall results show that the most occurring challenges for the practitioners who work in an Agile setup and perform CSSD as well are costing issues, communication process, cross-team communication, motivation and remuneration, quality assurance, scheduling and planning, communication with clients, crowd performance, and trust issues among the crowd.

The challenges discussed above are confirmed by the respondents and are identified from the literature. There is a possibility that practitioners face some other challenges, which are not previously reported in the literature. To cover this gap, the survey is designed 2-fold: confirmatory, to confirm the challenges presented already; and exploratory, to explore any other challenges not listed already. The survey led to the exploration of some new challenges as well which are faced by practitioners when they are working on Crowd–Agile development.

5.3 New challenges explored from the survey

For every category of challenges, there is an open-ended question asking whether they faced any other challenges. This is designed to explore any new challenges that practitioners face, which are not mentioned in the literature already. The challenges identified by the respondents are as follows:

- Some problems related to organization are identified by the respondents: resource provision, technological issues, employee handbook not being taken seriously, language barriers, and crowd productivity. The majority of respondents reported that resource provision is one of the frequently occurring problems when Agile is used with crowdsourcing software development
- For challenges related to software projects, 33.3% of respondents reported a "lack of knowledge and understanding" as a problem when dealing with the crowd. For 22.2% population, "unrealistic deadlines" are a problem when working in Crowd–Agile development. Other reported problems are unrealistic expectations, undefined goals, responsibility chain, and deployment issues.
- 60% of respondents reported "lack of task ownership" as a problem related to the task, and 40% reported "requirements gathering" as challenges related to the task.
- Some new challenges are identified by the respondents, related to team/crowd, which are not present in the literature. These are time management issues, crowd collaboration, crowd attitude, upskilling crowd, team building, lack of common vision, and monitoring and control. A total 41.7 % of respondents reported "time management as an issue," and the second most reported issue is "crowd collaboration" reported by 37.5% of respondents.
- The survey respondents identified some new challenges related to communication and coordination; 33.3% that the "manager and employee gap" and "communication with the crowd" are the problems they faced; 22.2 respondents faced the problems of "question asking" and "time difference"; and 11.1% of respondents faced these problems: "talent not showcased properly" and "taking of big picture by the crowd."
- Respondents are asked to report any other challenges they faced, which may (not) lie under any of the mentioned categories. The problems reported are politics within the crowd (reported by 60% of respondents), testing issues, intellectual property issues, process issues, and requirements change (reported by 20% of respondents).

Strategies followed to overcome these challenges: The survey identifies the strategies that are followed by industry practitioners to overcome the challenges faced. Some strategies are identified. A list of strategies is explored from the survey. Table 3 shows the strategies confirmed and explored via the survey.

During the survey, respondents were asked to identify the strategies they followed to overcome the challenges of Crowd–Agile development. As the challenges are grouped into five, namely, team-related problems, communication and coordination problems, organizational problems, software-related problems, and task-related problems, the respondents were asked to mention the strategies followed for each group of problems broadly. The responses from the participants are shown in Table 2.

6 Statistical analysis

Statistical analysis was performed to understand how challenges correlated to each other. The significance of metadata on challenges is also identified. Correlation tests and significance tests were conducted on the data. Correlations are found to check how strongly each challenge is related to others, and significant differences are found to check whether the nominal data have any significance over the challenges identified. The survey responses are saved as Excel files and imported to Jamovi for statistical analysis. Jamovi is the alternative to the SPSS tool and is used for statistical analysis.

6.1 Correlation test

Correlations are found among different challenges. As the data are ordinal, correlations can be found. Before checking the correlation, the possibilities of conducting tests are checked. The correlation is found for ordinal, interval, or ratio scale data only. As the frequency of challenges is ordinal data, correlation can be applied only to the challenges. Correlation cannot be applied to strategies as the data are not ordinal. Correlation cannot be applied to the roles, and demographics of the respondents, as it is the nominal data. For our data, Spearman's correlation is used as the relationship between the challenges is not linear. For monotonic relationships of variables, Spearman's correlation is most suitable, and hence, non-parametric correlation is performed. The strength of the correlation is presented according to the ranges given below; if the value of p is closer to +- 1, it indicates a very strong correlation (Glen, 2015):

1. 0.000–0.19 (very weak correlation)
2. 0.20–0.39 (weak correlation)
3. 0.40–0.59 (moderate correlation)
4. 0.60–0.79 (strong correlation)
5. 0.80–1.0 (very strong correlation)

Table 4 shows only strong correlations among challenges.

Correlation is calculated among the main categories of challenges as well as the challenges. The correlations of different categories of challenges are shown in Figure 3.

From Figure 3, it is shown that organization problems are positively corelated to software project-related challenges.

- Organizational-related issues have a strong correlation with software project-related issues, as the *p*-value is 0.75.
- Software project-related issues have a strong correlation with task-related issues, as the *p*-value is 0.73.
- Configuration and version management have a strong correlation with costing issues. *P*-value is 0.76.

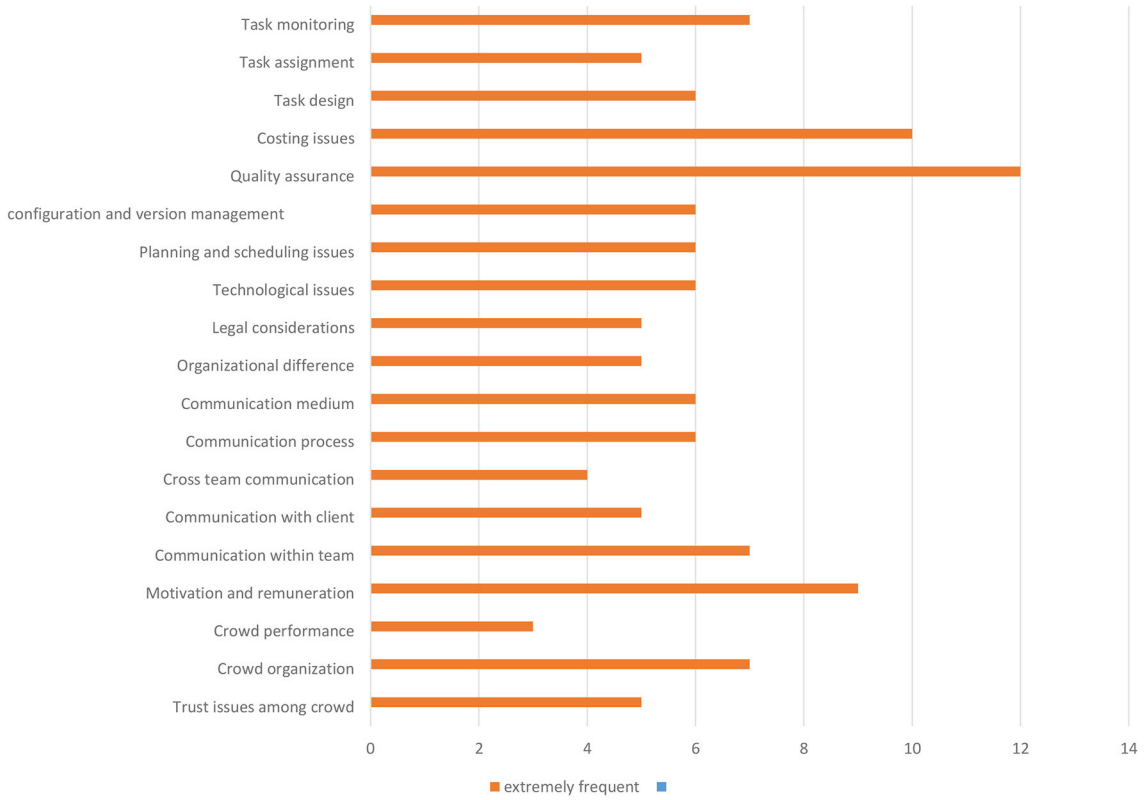

FIGURE 1
Extremely frequent challenge.

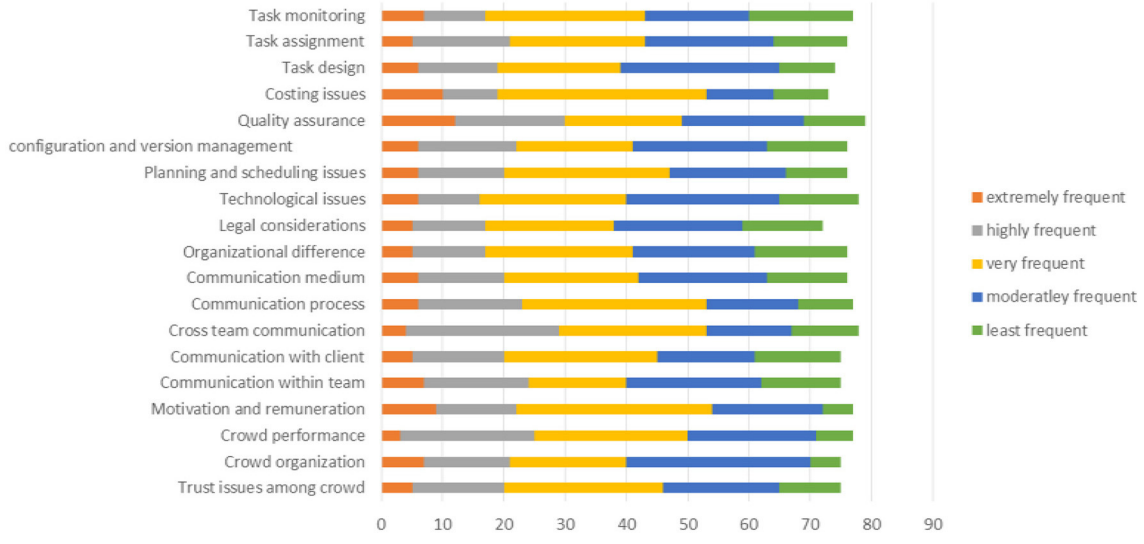

FIGURE 2
Frequency of occurrence of challenge.

TABLE 3 Strategies followed by industry practitioners.

Category	Strategies from SLR	Percentage response	New from survey	%age
Team /Crowd	Use monitoring systems that encourage collective and individual responsibilities	53.5%	Collaborating for realistic requirements for sprint	2.5%
	Rotate the staff between different roles of Agile projects regularly	31.3%	Knowledge sharing	4.9%
	Promoting group chat	57.5%	Simplify requirements	1.2%
	Promote informal interactions	46.3%	Video calls	1.2%
	Promote visits among distributed sites	22.5%	In-person and group meeting	1.2%
Coordination and communication	Training on collaboration and coordination tools	51.3%	Knowledge database to be shared across teams	1.3%
	Provide multiple communication modes and tools	47.4%	Creating opportunities within diverse teams	1.3%
	Support to face-to-face synchronous communication	64.1%	Sign language interpreter for hearing impaired	1.3%
	Create communication protocols	41%		
	Deploy knowledge transfer mechanisms	53.8%		
	Reduce the cross-communication	16.7%		
Organizational	Increasing common interests such as project and team goals and providing an organizational chart to all teams and members	82.7%	Describe value chain and business models visually	1.3%
	Letting the crowd plan among themselves without supervision	32%	A balance between mandating work and independent team to get there	1.3%
		Let crowd plan under supervision	1.3%	
		Introduce opportunities to increase interest	1.3%	
Software project r	Deploy and use a configuration management system	55.4%	Describe and share high-level goals	1.4%
	Promoting group chat	44.6%	Introduce virtual hierarchy	1.4%
	Using documentation and standards for the common design and goals	64.9%		
	Before participating, contestants must register for a certain competition	36.5%		
	Peer review of the submissions by the community	44.6%		
Task	Providing a sufficiently detailed specification for the task being crowdsourced	60.5		
	Decompose into small modules with clear requirements	75%		
	Limited interdependencies between modules	42.1%		
	Organizes tasks as competitions	35.5%		
	Peer review of the submissions by the community	32.9%		
Any Other	Proper requirements	20%		
	Common communicator	20%		
	The common time zone of work	20%		
	Clear project goals	20%		
	Project manager with a software background	20%		
	Less frequent requirements change	20%		

TABLE 4 Strong correlations among challenges.

Correlation among challenges			
Very strong correlation	None		
	Challenge 1	Challenge 2	Value
Strong correlations	Communication with client	Crowd performance	0.636843131
	Communication process	Crowd performance	0.668495597
	Communication process	Communication within team	0.625368
	Communication medium	Communication within team	0.625506
	Communication process	Communication with client	0.684233
	Communication medium	Communication with client	0.6649
	Communication process	Cross-team communication	0.622343
	Communication medium	Communication process	0.660405
	Communication process	legal considerations	0.614267
	Quality assurance	Communication process	0.65983
	Task design	Communication process	0.622524
	Organizational difference	Configuration and version management	0.700072
	Organizational difference	Costing issues	0.611892
	Task design	Legal considerations	0.611892
	Configuration and version management	Costing issues	0.768975
	Configuration and version management	Task design	0.64964
	Configuration and version management	Task assignment	0.673818
	Task design	Quality assurance	0.600327
	Task Assignment	Task monitoring	0.616171
	Task design	Task assignment	0.669559
Correlation among categories			
Challenges categories	Category 1	Category 2	Values
	Communication and coordination-related issues	Team-related Issues	0.601599
	Software Project-related issues	Organizational-related issues	0.753612
	Task-related issues	Organizational-related issues	0.694171
	Task-related issues	Software Project-related issues	0.734654

- Configuration and version management have a strong correlation with organizational differences. *P*-value is 0.70.

6.2 Significant difference

A significant difference is obtained to find whether any nominal data differ from the ordinal data. The significant difference is measured for groups of two nominal values. All possible nominal values are grouped to find their significant difference on any specific challenge. For significant difference, the Mann–Whitney test is used. The data are checked for the test applicability first. The Mann–Whitney test is suitable when the data are in ordinal form, and the difference between two samples is studied. The sample or independent variable should be in nominal form. In our case, the independent variables are the "roles" and the dependent variables are the challenges they face. This makes the Mann–Whitney test suitable for our data. Jamovi is used to carry out this test. The significance test is conducted between different roles and different sets of experiences. Significant difference between different methodologies and platforms is not conducted because organizations often follow more than one methodology or platform at a time. The test is also not conducted where the mean value of both independent values has a huge difference, so only a few sets of tests are conducted. The groups that are excluded from the test are as follows: *Manager and System Analyst*, *System Analyst and architect*, *System Analyst and software developer,* System analyst and tester, and experience *<=1 and 8+*. Significant difference and effect size are calculated as *P< 0.05, which means there is a significant difference*. The main findings of the test are as follows:

- To evaluate the difference between the frequencies of the "Quality assurance," the challenge faced by managers and

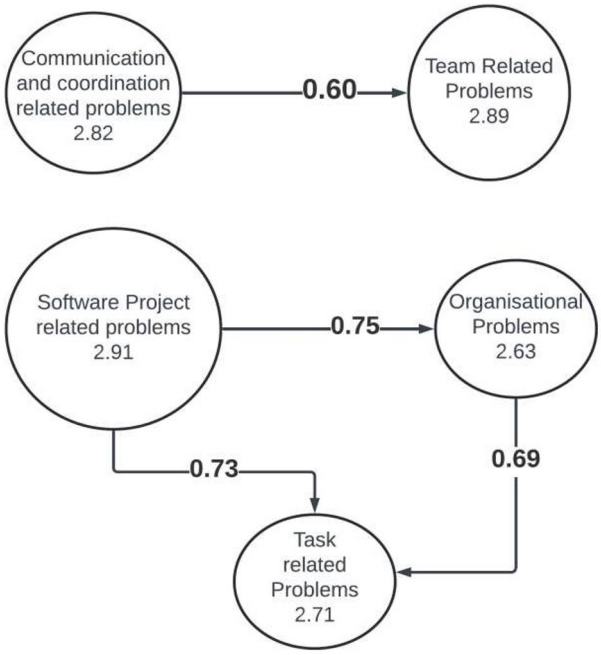

FIGURE 3
Correlation among categories.

architects is tested using the Mann–Whitney U-test. The test revealed significant difference in the challenges faced by managers (median = 3_, $N = 28$_) and architects (median = 1.5, $N = 8$), U = 61.0, $p = 0.049$, $r = 0.45536$.

- To evaluate the difference between the frequencies of "Task Monitoring," the challenge faced by managers and architects is tested using the Mann–Whitney U-test. The test revealed significant difference in the challenges faced by managers (median = 2.79_, $N = 28$_) and architects (median = 1.71, $N = 7$), U= 45.0, $p = 0.026$, $r = 0.54082$.
- To evaluate the difference between the frequencies of the "Technological issue," the challenge faced by architects and software developers is tested using the Mann–Whitney U-test. The test revealed a significant difference in the challenges faced by architects (median = 1.8_, $N = 8$_) and developers (median = 3, $N = 19$) U = 36.0, $p = 0.031$, $r = 0.5263$.
- To evaluate the difference between the frequencies of "configuration and version management," the challenge faced by Architects and software developers is tested using the Mann–Whitney U-test. The test revealed a significant difference in the challenges faced by architects (median = 1.86, $N = 7$) and developers (median = 2.95, $N = 19$) U = 29.5, $p = 0.028$, $r = 0.5564$.
- To evaluate the difference between the frequencies of the "Task monitoring," the challenge faced by architects and software developers is tested using the Mann–Whitney U-test. The test revealed significant difference in the challenges faced by architects (median = 1.71, $N = 7$) and developers (median = 2.84, $N = 19$) U = 30, $p = 0.029$, $r = 0.5489$.
- To evaluate the difference between the frequencies of the "Communication with client," the challenge faced by unexperienced (<1 year) and experienced (5–7 years) practitioners is tested using the Mann–Whitney *U*-test. The test revealed a significant difference in the challenges faced by unexperienced (<1 year) (median = 3.5, $N = 6$) and experienced (5–7 years) (median = 2.28, $N = 18$) U = 22, $p = 0.028$, $r = 0.5926$.
- To evaluate the difference between the frequencies of "trust issues among crowd," the challenges faced by unexperienced (<1 year) and experienced (5–7 years) practitioners are tested using the Mann–Whitney *U*-test. The test revealed a significant difference in the challenges faced by unexperienced (<1 year) (median = 3.38, $N = 8$) and experienced (5–7 years) (median = 2.5, $N = 18$) U= 37.5, $p = 0.050$, $r = 0.4792$.

7 Findings

This section provides a final list of challenges and strategies for Crowd–Agile development. Some challenges and strategies are initially identified from the SLR and validated through the survey. The details of the survey responses are discussed in the previous sections. The correlations and significance of these challenges are also discussed in previous sections. Only those challenges and strategies are included in the final list that are frequently occurring, i.e., which have a mean value of 2.5 or more and a standard deviation of <0.5. The challenges having a mean value of <2.5 are not included in the list. It is worth noting that these data are gathered from the participants who are globally dispersed, so these findings represent the opinion of a diverse group of people working across the globe. The challenges are categorized into different groups. A list of challenges and the strategies for each category is as follows:

7.1 Team/crowd

This category represents the challenges and strategies related to the team/crowd.

7.1.1 Team/crowd-related challenges

The final list of challenges related to crowds includes the following:

- Trust issues among the crowd working on the same project, related to their task.
- Crowd building/organization of crowd to effectively distribute the task of the same project.
- Crowd's attitude toward the task provided in terms of their performance.
- Remuneration is provided to the crowd to keep them motivated.
- Time management by the crowd for the given task (s).
- The upskilling crowd, as they are not part of the team so managers have no authority over their upskilling.

7.1.2 Strategies for team/crowd-related problems

The final list of strategies for team/crowd-related problems is as follows:

- Use monitoring systems that encourage collective and individual responsibilities.
- Rotate the staff between different roles of Agile projects regularly.
- Promoting group chat.
- Promote informal interactions.
- Promote visits among distributed sites.
- Knowledge sharing among the crowd.

7.2 Coordination and communication

This category represents the challenges and strategies that are related to coordination and communication. The final list of challenges for this category includes the following:

- Communication with team/crowd regarding a task that is from the same project.
- Communication with the client regarding requirements and acceptance of a task.
- Cross-team communication: when different teams of the crowd are working on different modules of the same project, they may need to communicate.
- The communication process among the crowd workers is usually not defined, and they all follow the different communication process.
- Communication medium among different crowd workers is different.

The finalized list of strategies for communication and coordination challenges is as follows:

- Train the crowd workers on collaboration and coordination tools.
- Provide multiple communication modes and tools for the crowd workers.
- Support face-to-face communication among crowd workers and their management.
- Create communication protocols for crowd workers who work on the same project.
- Deploy a knowledge transfer mechanism that provides knowledge about the project to crowd workers.
- Reduce cross-team communication by designing low-coupled tasks for a crowd.

7.3 Organizational

This category represents the challenges and strategies that are related to the organization. The final list of organization-related challenges includes the following:

- Organizational structure is different when the crowd is from different organizations.
- Legal considerations, e.g., copyrights, and employee rights of the requesting organizations for the crowd can differ.
- Technological issues among the requesting organization and crowd workers.
- Planning/scheduling of tasks that are to be assigned to crowd workers.
- Resource provision to crowd according to the project's needs.

A final list of strategies to reduce organizational challenges includes the following:

- Increasing common interests among the crowd such as project and team goals.
- Providing an organizational chart to all crowd members for their knowledge.
- Letting the crowd plan their tasks among themselves without the supervision of a manager.

7.4 Software project

This category represents the challenges and strategies that are related to software projects. The final list of challenges related to software projects includes the following:

- Configuration and version management of tasks that are performed by the crowd.
- Quality assurance of the task performed by the crowd.
- Costing of tasks that are to be assigned to crowd workers.
- Lack of knowledge/understanding of the project by a crowd
- Unrealistic deadlines are given to the crowd resulting in delays.

The final list of strategies for software project-related problems includes the following:

- Deploy and use a configuration management system.
- Promoting group chat among the crowd working on the same project and the manager.
- Using documentation and standards for the common design and goals for crowd workers.
- Before participating, contestants (crowd workers) must register for a certain competition.
- Peer review of the submissions by the crowd worker.

7.5 Task

This category represents the challenges and strategies that are related to the task. The final list of challenges related to tasks is as follows:

- Task design to be assigned to crowd.
- Task assignment to crowd workers as per their skills.

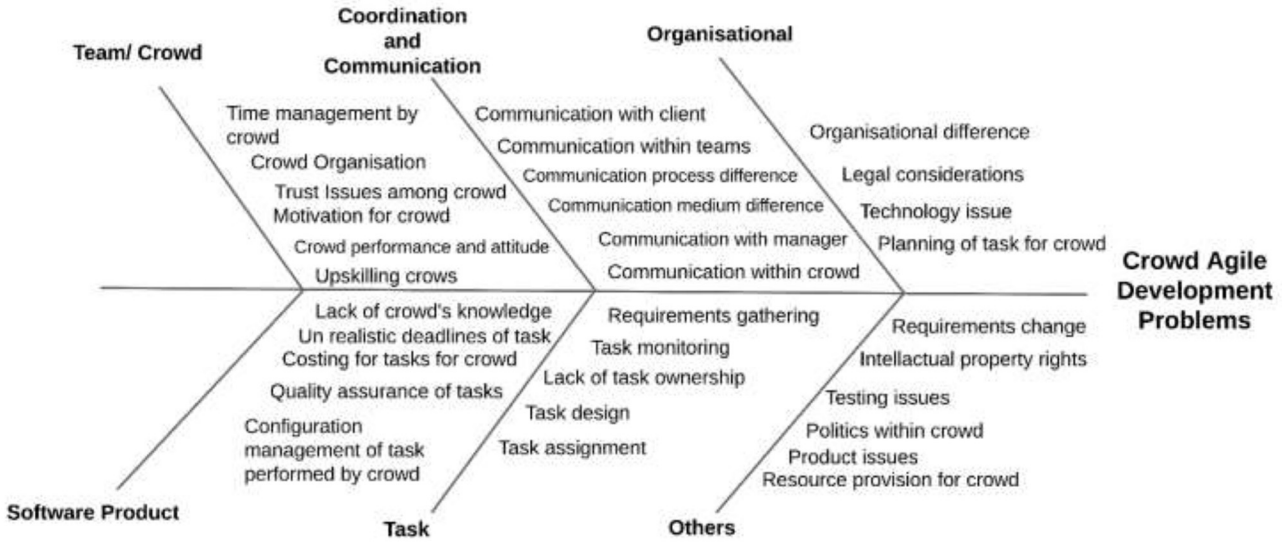

FIGURE 4
List of challenges for crowd–agile development.

- Task monitoring by a manager for task completion and quality.
- Lack of task ownership by crowd performing the task.
- Requirements gathering from the client.

The final list of strategies related to the task includes the following:

- Providing a sufficiently detailed specification for the task being crowdsourced.
- Decompose tasks into smaller sub-tasks with clear requirements and goals.
- Limited interdependencies between modules by designing low-coupled tasks.
- Organizes tasks as competitions, i.e., the best-performed task by the crowd is selected.
- Peer review of the submissions by the crowd community.

7.6 Others

During the survey, industry practitioners are asked to state any other challenges they face while performing any software development activity using crowdsourcing and Agile; 60% of the respondents state that politics within the crowd is a major challenge; 20% of respondents mentioned testing issues, intellectual property rights, product issues, and change of requirements as frequently occurring challenges at their end. Some new challenges that are identified from the survey are as follows:

- Politics within the crowd workers.
- Testing issues for the task performed by the crowd.
- Intellectual property rights of the tasks performed by the crowd.
- Product-related issues, like the sensitivity of the information, and privacy.
- Change of requirements is difficult to cope with by crowd workers.

Figure 4 shows the finalized challenges faced by practitioners during Crowd–Agile development.

Industry practitioners are also asked to mention the strategies they follow to reduce the challenges faced while working in Agile-Crowdsource software development. The final list of new challenges coming from the industrial survey is as follows:

- Proper requirements should be given to the crowd.
- Common communicator for the crowd workers working on the same project.
- The common time zone of work among crowd workers.
- Clear project goals should be delivered to crowd workers.
- The Project Manager should from a software background to understand the technical issues of crowd.
- Less frequent requirements change.

8 Discussion and limitations

The changing nature of software has led to different paradigms of software development. The new paradigms come with their pros and cons. Crowdsource software development is one such paradigm. However, it becomes more challenging when it is used within an Agile environment. Many software development industries perform crowdsourcing software development while following agile methodology. Contradicting characteristics of both approaches make it challenging for practitioners to follow both approaches for effective software development. This study finds the challenges from practitioners who are involved in crowdsourcing software development and agile. An online survey is conducted

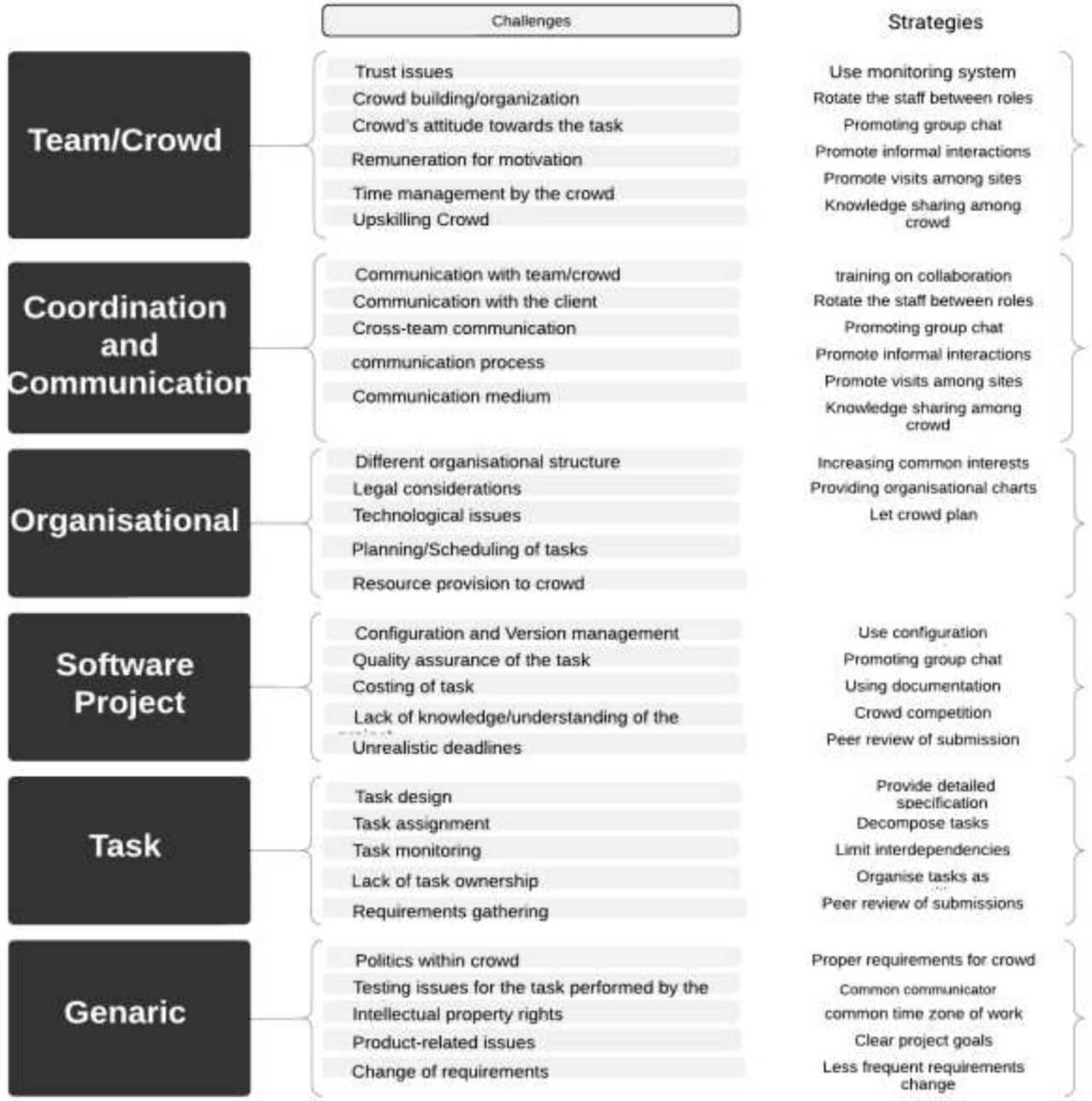

FIGURE 5
Findings of the study.

for this purpose. Survey questions are designed carefully based on SLR outcome, and data are also carefully analyzed. In particular, the challenges faced by practitioners are related to the trust issues among crowd members. Another main factor involved is the coordination and collaboration among crowd workers. As the collaboration among the crowd is challenging, it affects the software development. Task design and assignments for a crowd are also very challenging. Another problem faced by crowd workers is that the knowledge about the project is not shared openly with the crowd, which results in an inappropriate understanding of the project. Differences in the legal considerations of organizations also become challenging. Practitioners have shared some strategies to reduce the impact of these challenges. They have suggested that a communication protocol is to be used and crowd workers should be trained for these tools. Informal group chats and visits are encouraged. It is suggested to share the project goals and specifications with all the crowd workers for clarity. It is deemed important to design a task in such a way that they are loosely coupled. Figure 5 presents the findings of this study.

This is the first hand information from industry practitioners, but the study still has many limitations. The challenges and strategies identified by the practitioners need to be empirically verified. Researchers can use this list to empirically verify the list of challenges and strategies. Another limitation of this study is the limited audience; as the study was conducted till third wave of snowballing, it is suggested to identify the challenges for a larger scale and bigger projects.

The survey questionnaire is carefully designed; however, the survey may pose some construct validity threat as some information might have been missed. To overcome this, a survey was pilot-tested, and the suggestions from the participants were taken to improve survey questionnaires to remove any ambiguity. The data are gathered from the practitioners of industry, so their responses may be based on personal biases or preferences. The authors have a personal suggestion of using Distributed Ledger Technology (DLT) (Gorski and Bednarski, 2020), which is used for sharing data storage for collaborating parties. This is in line with the architecture of crowdsourcing software development. This can be helpful when crowd workers are also following Agile as suggested by the survey participants. DLT can help them store data in a decentralized manner, and the crowd working on the shared data can avoid the challenges they face due to lack of shared knowledge and data. Requirement engineering by the crowd can also be solved by gamification of requirements (Yasin et al., 2021). Gamification is a promising strategy in requirement gathering, leveraging game-like interfaces to engage users more effectively. By incorporating elements of gameplay into the process, gamification encourages greater user participation and precision in articulating their requirements. This approach has shown potential in mitigating communication barriers that can impede accurate requirements gathering. Through interactive and immersive experiences, gamification not only enhances user engagement but also facilitates clearer and more comprehensive communication, ultimately contributing to the accuracy and efficacy of the requirement-gathering process. Authors have another personal suggestion to further conduct research on how these game-based techniques can work for the crowd.

9 Conclusion

In this study, we have identified the human factors involved in Crowd–Agile software development. These factors involve the challenges that practitioners face during crowdsourcing software development while working in an agile setup. The research produces a verified list of challenges from literature and industry. It is shown that trust issue among the crowd is very challenging, which hinders knowledge sharing and affects the ultimate goal of the project. As the crowd is a group of heterogeneous people, collaboration and coordination among them is very difficult. Many large and complex projects require communication and collaboration among members, the absence of which may create issues in the project. Considering this situation task design also becomes challenging. Some challenges vary for different projects and organizations. Another challenge for crowd members is not having access to project goals and knowledge. However, practitioners also follow some strategies to reduce these challenges, such as sharing a repository of knowledge, providing communication protocols, encouraging informal chat, and making chat groups. This list of validated challenges and strategies is helpful for researchers for further research as the Agile–Crowd is relatively a new term in the literature. This is also helpful to other practitioners in Crowd–Agile development. Practitioners can use this list as a guideline to reduce the challenges they face. The challenges are grouped into categories, and strategies are also suggested against each category. Industry practitioners can identify the relevant category of the challenge they face. Within the challenge category, they can find enlisted strategies, which can help them reduce stated challenges. The research highlights the importance of embracing a pragmatic managerial approach in Crowd–Agile software development. It advises software project managers to implement the suggested strategies to address common challenges such as communication, trust, and collaboration, thereby enhancing the likelihood of project success. By adopting these strategies, managers can also harness innovative software development practices while fostering a sustainable development environment. Researchers can further use this list to empirically verify these challenges. This research can be conducted on a larger scale in future for a larger population. In future, we intend to propose a model for Crowd–Agile development. We are also working on a detailed analysis of these issues identified by the industry.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Ethics statement

Ethical review and approval was not required for the study on human participants in accordance with the local legislation and institutional requirements. Written informed consent from the patients/participants or patients/participants' legal guardian/next of kin was not required to participate in this study in accordance with the national legislation and the institutional requirements.

Author contributions

SQ: Conceptualization, Data curation, Methodology, Validation, Visualization, Writing – original draft. SI: Conceptualization, Methodology, Supervision, Writing – review & editing. HH: Methodology, Supervision, Writing – review & editing. AA: Writing – review & editing. VK: Writing – review & editing.

Funding

The author (s) declare that no financial support was received for the research, authorship, and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Ågerfalk, P. J. (2006). "Towards better understanding of agile values in global software development," in *EMMSAD,* 13–20.

Agerfalk, P. J., Fitzgerald, B., Holmstrom Olsson, H., Lings, B., Lundell, B., and Conchúir, E. (2005). "A framework for considering opportunities and threats in distributed software development," in *Proceedings of the of DiSD'05* (Austrian Computer Society), 47–61.

Ali Khan, J., Liu, L., Wen, L., and Ali, R. (2020). Conceptualising, extracting and analysing requirements arguments in users' forums: the CrowdRE-Arg framework. *J. Softw. Evol. Proc.* 32:e2309. doi: 10.1002/smr.2309

Alsahli, A., Khan, H., and Alyahya, S. (2017). Agile development overcomes GSD challenges: a systematic literature review. *Int. J. Comput. Sci. Softw. Eng.* 6:7.

Al-Saqqa, S., Sawalha, S., and AbdelNabi, H. (2020). Agile software development: Methodologies and trends. *Int. J. Int. Mob. Technol.* 14:13269. doi: 10.3991/ijim.v14i11.13269

Asiegbu Baldwin, C., Oluigbo Ikenna, V., Ajakwe Simeon, O., and Onyike Gerald, O. (2017). Crowdsourcing software development: concept, benefits, and adoption. *Int. J. Sci. Res. Comput. Sci. Eng.* 5, 7–16.

Barros, L., Tam, C., and Varajao, J. (2024). Agile software development projects–Unveiling the human-related critical success factors. *Inf. Softw. Technol.* 170:107432. doi: 10.1016/j.infsof.2024.107432

Beecham, S., Clear, T., Lal, R., and Noll, J. (2021). Do scaling agile frameworks address global software development risks? An empirical study. *J. Syst. Softw.* 171:110823. doi: 10.1016/j.jss.2020.110823

Beecham, S., Noll, J., and Richardson, I. (2014). "Using agile practices to solve global software development problems–a case study," in *2014 IEEE International Conference on Global Software Engineeering Workshops*. IEEE, 5–10.

Beretta, M., Frederiksen, L., Wallin, M., and Kulikovskaja, V. (2021). Why and how firms implement internal crowdsourcing platforms. *IEEE Trans. Eng. Manage.* 70, 3036–3049. doi: 10.1109/TEM.2020.3045118

Bhatti, S. S., Gao, X., and Chen, G. (2020). General framework, opportunities and challenges for crowdsourcing techniques: a comprehensive survey. *J. Syst. Softw.* 167:110611. doi: 10.1016/j.jss.2020.110611

Bowes, J. (2015). *Kanban vs Scrum vs XP-an Agile Comparison*. Kanban vs Scrum. Available online at: https://manifesto.co.uk/kanban-vs-scrum-vs-xp-an-agile-comparison/ (accessed December, 2021).

Capretz, L. F. (2014). Bringing the human factor to software engineering. *IEEE Soft.* 31, 104–104. doi: 10.1109/MS.2014.30

Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., García-Peñalvo, F. J., and Tovar, E. (2014). Project managers in global software development teams: a study of the effects on productivity and performance. *Softw. Q. J.* 22, 3–19. doi: 10.1007/s11219-012-9191-x

Donca, I. C., Stan, O. P., Misaros, M., Gota, D., and Miclea, L. (2022). Method for continuous integration and deployment using a pipeline generator for agile software projects. *Sensors* 22:4637. doi: 10.3390/s22124637

Dwarakanath, A., Chintala, U., Shrikanth, N. C., Virdi, G., Kass, A., Chandran, A., and Paul, S. (2015). "Crowd build: A methodology for enterprise software development using crowdsourcing," in *2015 IEEE/ACM 2nd International Workshop on CrowdSourcing in Software Engineering*. IEEE, 8–14.

Erich, F. M., Amrit, C., and Daneva, M. (2017). A qualitative study of DevOps usage in practice. *J. Softw. Evol. Proc.* 29:e1885. doi: 10.1002/smr.1885

Glen, S. (2015). *Spearman Rank Correlation (Spearman's Rho): Definition and How to Calculate It. Statistics How to. in StatisticsHowTo.com: Elementary Statistics for the Rest of Us!, 2022*. Available online at: https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/spearman-rank-correlation-definition-calculate/

Górski, T. (2021a). The 1+ 5 architectural views model in designing blockchain and IT system integration solutions. *Symmetry* 13:2000. doi: 10.3390/sym13112000

Górski, T. (2021b). Towards continuous deployment for blockchain. *Appl. Sci.* 11:11745. doi: 10.3390/app112411745

Gorski, T., and Bednarski, J. (2020). Applying model-driven engineering to distributed ledger deployment. *IEEE Access* 8, 118245–118261. doi: 10.1109/ACCESS.2020.3005519

Hamilton, A. F. D. C., and Holler, J. (2023). Face2face: advancing the science of social interaction. *Philos. Trans. Royal Soc. B* 378:20210470. doi: 10.1098/rstb.2021.0470

Hosseini, M., Phalp, K. T., Taylor, J., and Ali, R. (2014). "Towards crowdsourcing for requirements engineering," in *Proceeding of REFSQ Co-Located Events*.

Howe, J. (2006). The rise of crowdsourcing. *Wired Magazine* 14, 176–183.

Ilyas, M., Khan, S. U., Khan, H. U., and Rashid, N. (2024). Software integration model: an assessment tool for global software development vendors. *J. Softw. Evol. Proc.* 36:2540. doi: 10.1002/smr.2540

Jabangwe, R., Šmite, D., and Hessbo, E. (2016). Distributed software development in an offshore outsourcing project: a case study of source code evolution and quality. *Inf. Softw. Technol.* 72, 125–136. doi: 10.1016/j.infsof.2015.12.005

Kasunic, M. (2005). *Designing an Effective Survey*. Pittsburgh, PA: Software Engineering Institute.

Kausar, M., Ishtiaq, M., and Hussain, S. (2021). Distributed agile patterns-using agile practices to solve offshore development issues. *IEEE Access* 10, 8840–8854. doi: 10.1109/ACCESS.2021.3136923

Khan, J. A., Liu, L., Wen, L., and Ali, R. (2019a). "Crowd intelligence in requirements engineering: Current status and future directions," in *Requirements Engineering: Foundation for Software Quality: 25th International Working Conference, REFSQ 2019, Essen, Germany, March 18–21, 2019, Proceedings 25*. Springer International Publishing, 245–261.

Khan, J. A., Xie, Y., Liu, L., and Wen, L. (2019b). "Analysis of requirements-related arguments in user forums," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 63–74.

Khan, J. A., Yasin, A., Fatima, R., Vasan, D., Khan, A. A., and Khan, A. W. (2022). Valuating requirements arguments in the online user's forum for requirements decision-making: the CrowdRE-VArg framework. *Software Prac. Exp.* 52, 2537–2573. doi: 10.1002/spe.3137

Khan, R. A., Khan, S. U., Alzahrani, M., and Ilyas, M. (2022). Security assurance model of software development for global software development vendors. *IEEE Access* 10, 58458–58487. doi: 10.1109/ACCESS.2022.3178301

Laukkanen, E., Itkonen, J., and Lassenius, C. (2017). Problems, causes and solutions when adopting continuous delivery—A systematic literature review. *Inf. Softw. Technol.* 82, 55–79. doi: 10.1016/j.infsof.2016.10.001

Lenberg, P., Feldt, R., and Wallgren, L. G. (2015). Human factors related challenges in software engineering–an industrial perspective. In *2015 ieee/acm 8th international workshop on cooperative and human aspects of software engineering* (pp. 43-49). IEEE. doi: 10.1109/CHASE.2015.13

Li, W., Tsai, W. T., and Wu, W. (2015). Crowdsourcing for large-scale software development. *Crowdsourcing: Cloud-Based Softw. Dev.* 11, 3–23. doi: 10.1007/978-3-662-47011-4_1

Mao, K., Capra, L., Harman, M., and Jia, Y. (2017). A survey of the use of crowdsourcing in software engineering. *J. Syst. Softw.* 126, 57–84. doi: 10.1016/j.jss.2016.09.015

Meier, A., and Kock, A. (2023). The human factor in agility: exploring employee dedication in agile project organizations. *Int. J. Proj. Manage.* 41:102527. doi: 10.1016/j.ijproman.2023.102527

Moslehi, P., Adams, B., and Rilling, J. (2016). "On mining crowd-based speech documentation," in *Proceedings of the 13th International Conference on Mining Software Repositories,* 259–268.

Mukherjee, D., Kumar, S., Pandey, N., and Lahiri, S. (2023). Is offshoring dead? A multidisciplinary review and future directions. *J. Int. Manage.* 29:101017. doi: 10.1016/j.intman.2023.101017

Niazi, M., Mahmood, S., Alshayeb, M., Riaz, M. R., Faisal, K., Cerpa, N., and Richardson, I. (2016). Challenges of project management in global software development: A client-vendor analysis. *Inf. Softw. Technol.* 80, 1–19. doi: 10.1016/j.infsof.2016.08.002

Ojha, T. R., and Chaudhary, P. (2022). Enabling extreme programming (XP) in global software development (GSD) practice. *J. Advanc. Softw. Eng. Testing* 5, 15–25.

Prasetio, R. T., Ramdhani, Y., and Alamsyah, D. P. (2021). "Scrum method in help-desk ticketing and project management system," in *2021 3rd International Conference on Cybernetics and Intelligent System (ICORIS)*. IEEE, 1–6.

Qayyum, S., Imtiaz, S., and Khan, H. H. (2020). "Crowd agile model for effective software development," in *Agile Processes in Software Engineering and Extreme Programming–Workshops: XP 2020 Workshops, Copenhagen, Denmark, June 8–12, 2020, Revised Selected Papers 21*. Cham: Springer International Publishing, 272–279.

Qayyum, S., Imtiaz, S., and Khan, H. H. (2023). Challenges of agile–crowd software development: a systematic literature review. *J. Circ. Syst. Comput.* 32:2330001. doi: 10.1142/S0218126623300015

Rasnacis, A., and Berzisa, S. (2017). Method for adaptation and implementation of agile project management methodology. *Proc. Comp. Sci.* 104, 43–50. doi: 10.1016/j.procs.2017.01.055

Ruhe, G., and Wohlin, C. (2014). *Software Project Management in a Changing World, Vol. 21*. Berlin: Springer.

Shameem, M., Kumar, C., and Chandra, B. (2015). "Communication related issues in GSD: An exploratory study," in *2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*. IEEE, 1–5.

Siegmund, J. (2024). *New Perspectives on the Human Factor in Software Engineering. Software Engineering 2024 (SE 2024)*. Bonn: Gesellschaft für Informatik eV, 23.

Singh, A., Singh, K., and Sharma, N. (2015). Agile in global software engineering: an exploratory experience. *Int. J. Agile Systems Manage.* 8, 23–38. doi: 10.1504/IJASM.2015.068607

Srivastava, A., Bhardwaj, S., and Saraswat, S. (2017). "SCRUM model for agile methodology," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 864–869.

Stol, K. J., Caglayan, B., and Fitzgerald, B. (2017a). Competition-based crowdsourcing software development: a multi-method study from a customer perspective. *IEEE Trans. Softw. Eng.* 45, 237–260. doi: 10.1109/TSE.2017.2774297

Stol, K. J., and Fitzgerald, B. (2014a). "Researching crowdsourcing software development: perspectives and concerns," in *Proceedings of the 1st International Workshop on CrowdSourcing in Software Engineering,* 7–10.

Stol, K. J., and Fitzgerald, B. (2014b). "Two's company, three's a crowd: a case study of crowdsourcing software development," in *Proceedings of the 36th International Conference on Software Engineering,* 187–198.

Stol, K. J., LaToza, T. D., and Bird, C. (2017b). Crowdsourcing for software engineering. *IEEE Softw.* 34, 30–36. doi: 10.1109/MS.2017.52

Tyagi, S., Sibal, R., and Suri, B. (2022). Empirically developed framework for building trust in distributed agile teams. *Inf. Softw. Technol.* 145:106828. doi: 10.1016/j.infsof.2022.106828

Verwijs, C., and Russo, D. (2023). A theory of scrum team effectiveness. *ACM Trans. Softw. Eng. Methodology* 32, 1–51. doi: 10.1145/3571849

Yasin, A., Fatima, R., Ali Khan, J., Liu, L., Ali, R., and Wang, J. (2023). Counteracting sociocultural barriers in global software engineering using group activities. *J. Softw. Evol. Proc.* 36:e2587. doi: 10.1002/smr.2587

Yasin, A., Fatima, R., JiangBin, Z., Ali Khan, J., and Ali Khan, A. (2021). Gamifying requirements: an empirical analysis of game-based technique for novices. *J. Softw. Evol. Proc.* 22:e2617.

Yasin, A., Fatima, R., Liu, L., Ali Khan, J., Ali, R., and Wang, J. (2022). On the utilization of non-quality assessed literature in software engineering research. *J. Softw. Evol. Proc.* 34:e2464. doi: 10.1002/smr.2464

Zhen, Y., Khan, A., Nazir, S., Huiqi, Z., Alharbi, A., and Khan, S. (2021). Crowdsourcing usage, task assignment methods, and crowdsourcing platforms: a systematic literature review. *J. Softw. Evol. Proc.* 33:e2368. doi: 10.1002/smr.2368