Information and Software Technology 58 (2015) 373–387

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Categorization of risk factors for distributed agile projects

Suprika V. Shrivastava *, Urvashi Rathod [1]

Symbiosis Centre for Information Technology (SCIT), Symbiosis International University (SIU), Rajiv Gandhi Infotech Park, Hinjewadi, Pune 411 057, Maharashtra, India

ARTICLE INFO

Article history:
Received 3 March 2014
Received in revised form 5 July 2014
Accepted 7 July 2014
Available online 21 July 2014

Keywords:
Distributed Agile Development (DAD)
Agile software development (ASD)
Distributed Software Development (DSD)
Risk factor classification
Leavitt's' model
Risk management in distributed agile development

ABSTRACT

Context: Organizations combine agile approach and Distributed Software Development (DSD) in order to develop better quality software solutions in lesser time and cost. It helps to reap the benefits of both agile and distributed development but pose significant challenges and risks. Relatively scanty evidence of research on the risks prevailing in distributed agile development (DAD) has motivated this study.
Objective: This paper aims at creating a comprehensive set of risk factors that affect the performance of distributed agile development projects and identifies the risk management methods which are frequently used in practice for controlling those risks.
Method: The study is an exploration of practitioners' experience using constant comparison method for analyzing in-depth interviews of thirteen practitioners and work documents of twenty-eight projects from thirteen different information technology (IT) organizations. The field experience was supported by extensive research literature on risk management in traditional, agile and distributed development.
Results: Analysis of qualitative data from interviews and project work documents resulted into categorization of forty-five DAD risk factors grouped under five core risk categories. The risk categories were mapped to Leavitt's model of organizational change for facilitating the implementation of results in real world. The risk factors could be attributed to the conflicting properties of DSD and agile development. Besides that, some new risk factors have been experienced by practitioners and need further exploration as their understanding will help the practitioners to act on time.
Conclusion: Organizations are adopting DAD for developing solutions that caters to the changing business needs, while utilizing the global talent. Conflicting properties of DSD and agile approach pose several risks for DAD. This study gives a comprehensive categorization of the risks faced by the practitioners in managing DAD projects and presents frequently used methods to reduce their impact. The work fills the yawning research void in this field.

1. Introduction

In last two decades, software development has evolved from being concentrated at a single site to being geographically distributed across the globe and hence, characterized as Distributed Software Development (DSD) [79]. DSD helps the organizations to gain time-zone effectiveness, leverage a large skill pool, develop software closer to the customer's requirements and exploit low labor cost in certain parts of the world [67]. Although, there are other related terms like Global Software Development (GSD), Multisite Development, Dispersed Development, Off shoring, Outsourcing [50], being commonly used in the literature, but Distributed Software development (DSD) is a general term to represent software development with dispersed teams. We have used DSD in this paper as it does not constrain us to a particular scale [53].

Further, modern organizations are working under tight time and cost constraints and the development of software occurs in highly volatile environment due to the changes in the product requirement, business and market needs. Eventually, many organizations are adopting agile methodology for software development as it is able to deliver products that satisfy customer needs and is faster than the traditional approach [46]. There is an increasing interest in applying agile practices in DSD projects to leverage the combined advantage of both the approaches [80].

Agile when combined with DSD also brings some new challenges and associated risks and makes the software development process more complicated [43]. DSD and agile work on different principles, which makes the distributed agile projects difficult to manage. DSD requires formal communication amongst the geographically distributed team members while agile is based on informal communication with co-located teams working in close collaboration. Several agile best practices including

* Corresponding author. Cell: +91 9373313943.
E-mail addresses: suprikashrvt@gmail.com (S.V. Shrivastava), urvashi.srathod@gmail.com (U. Rathod).
[1] Cell: +91 9421003803.

http://dx.doi.org/10.1016/j.infsof.2014.07.007
0950-5849/

collaboration, face to face communication, self organizing teams, retrospectives, showcases become more challenging in the distributed model [56].

Risks are inherent to software development projects and field experiences with traditional development approach in the past six decades have been instrumental in creating a reasonably comprehensive knowledge base in the discipline [58,88]. However, the current literature does not consider all the aspects of risks in DAD software development projects and the corresponding solutions [42].

Some case studies discuss the problems and challenges while executing distributed agile projects. These studies are scattered in different articles and do not provide a comprehensive view of risks in DAD [48,63,68,99]. As a result, software community lacks an overall picture of what types of risks the companies may encounter while attempting to combine the two methods [47].

Lack of understanding about the risks specific to DAD projects may have played a significant role in the failures of several such projects [66,42,47] and thus, raising doubts on the claims made on their ability to deliver good quality software that can satisfy customer needs in given timeframe. Practitioners in the field have always been keen to learn more from the consolidated experiences of the fraternity. Academic and research community has worked together with practicing developers, not only for consolidation of experiences but also for deriving relevant patterns, frameworks, guiding principles and theories for strengthening the discipline of software engineering [74,79,66]. Present study is a similar effort, wherein, in the absence of much work on risks in DAD projects, a comprehensive exploration of related industry experience has been done. The objective has been to understand the risk factors as experienced by practitioners, risk management techniques used by them and mapping the risks in organizational context so that industry practitioners can take decisions effectively.

This study involved in-depth interviews with the practitioners who have handled DAD projects in various multinational Information Technology (IT) consulting companies. The inputs obtained were strengthened by the analysis of work documents of DAD projects with simultaneous discussions with the involved project managers or team members. Further, support from earlier research literature for the findings in related development environments has been consolidated in this work.

The findings of the study contribute in following ways:

1. This study provides a comprehensive classification of risk factors in Distributed Agile Projects. Forty-five risk factors have been identified, which fall under five major risk categories. This work also documents the risk management approaches used in real world for controlling the respective risk factors.
2. Further, the risk categories identified were mapped to four interacting components of the Leavitt's Model of Organizational Change i.e. Task, Structure, Actor and Technology and their interaction. This mapping would help organizations to develop the policies, procedures and guidelines for managing risks in DAD projects.
3. The study relates the cause of occurrence risk factors in DAD projects to the contradiction between the properties of agile development and distributed development.
4. Although, many of these risk factors that occur in DAD projects also exists in DSD or agile projects, but their severity of impact increases in DAD projects. Moreover, the study uncovers certain risk factors which have very less or in some case negligible mention in the research literature. This is an important contribution of the work as the cognizance of these risk factors will help practitioners while executing projects and also, open new avenues for further research work.
5. Corresponding to each risk factor in DAD project, the properties of DSD and the principles of agile approaches which in contrast with each other and hence become a reason for the risk factor to occur were identified.

As stated above, the initial exploratory work is successful in developing new insights for practitioners and unveiling new avenues for research. The work involves multiple sources of information on risks in DAD projects and so, ensures the depth of the findings and establishes the merit of the work.

The study has some limitations rooted in the reluctance of the practitioners to hand over the project work documents to the researchers due to confidentiality constraints, which may have affected the rigor of the risk identification process. This problem was dealt, by performing analysis of project work documents at the company site and involving the team members of the project in simultaneous discussions.

In this paper, we discuss the research background in Section 2. This is followed by the research method adopted for the study in Section 3, which includes the research objectives, research design and the data analysis approach used. Section 4 presents the results and discussions. Limitations of the work and future scope have been stated in Sections 5 and 6 respectively followed by conclusion in Section 7.

2. Research background

2.1. Significance of risk management in distributed agile development projects

Agile methodologies work very well in highly dynamic business and IT environment as they help the team to respond to change and continuously deliver business value. Many organizations that develop software using agile approach have started looking for skills and talent available at much lower wage-rates and are anxious to source the development work to these centers [22]. Hence, the organizations are using distributed agile development for developing flexible and evolving solutions to fulfill their business needs.

Most of the agile methodologies (e.g. scrum, Xp) assume that the team is located in a single room. Unfortunately, this principle does not fit in the real scenario where agile teams are also distributed across the geographical locations. A survey conducted by VersionOne, states that organizations are constantly scaling agile beyond single team and single project [101]. These facts clearly show that there is a need to extend the agile practices to distributed software development.

Software application development itself is subject to many fundamental risks posed by the typical characteristics of the product and the process. The incompatibility of DSD and agile leads to problems like to weak social interactions, delays due to time-zone differences, lack of coordination, lack of tool support and infrastructure, dependencies between distributed teams and difficulty in knowledge sharing due to team dispersion in different geographic locations [42,24]. Risks inherent to agile are limited documentation, customer non-agile alignment, lack of team agile skills [42], difficulty in having ongoing negotiations between customer and developers to reach acceptable levels of quality and informal people-oriented approach [80].

A recent survey by Scott Ambler on scaling agile shows that, greater is the level of geographic distribution, greater is the risk due to communication and coordination challenges, resulting in lower success rate [1]. Another survey result shows that 60% of co-located agile projects are successful, while roughly 25% can be considered as failed projects. On the other hand, although, more than 50% distributed agile projects have been successful, but 50% of them have failed too [2].

Higher failure rate of projects using DSD in an agile environment is indicative of the extent of difficulty and associated risks in exe-

cuting them. Thus, risk management becomes critical for proactively, preventing risks in a continuous and concurrent manner.

2.2. Review of related literature

As the study is about risk involved in software development; it has to be viewed in the context of projects. Project risks can be defined as uncertain events or conditions that, if occur, have a positive or negative effect on at least one of the project objectives such as time, cost, scope or quality [76]. A risk factor can be defined as a condition that can present a serious threat to the successful completion of a software development project [88]. Risk management is the process of identifying risk, assessing risk and taking steps to reduce the risks to an acceptable level [96].

Research literature reveals the scarcity of work in the area of risk management for distributed agile projects [43]. Although, previous case studies have shown that agile methods, such as scrum, can be easily and successfully customized to distributed projects [27,97], there is a paucity of work on the risk management in distributed agile development [68,42,47]. A comparative analysis of the traditional risk management and agile approach has been done by Nyfjord and Kajko-Mattsson [66]. They state that "None of the agile approaches suggest any risk management taxonomy, nor do they suggest any systematic way of describing and classifying." Authors also assert that "Neither of the majority traditional risk management frameworks except IEEE 1540 and SEI -Software Risk Evaluation nor agile models provide any template for collecting, structuring and communicating risk related information." They suggested that "Agile should learn to integrate traditional risk management practices in order to ensure effective risk management risks."

Some work has been done on risk management in distributed development environment, but does not take agile development in account. A pioneering work of Prikladnicki et al. [79] proposes a reference model for Global Software Development (GSD), wherein, the processes and challenges in various phases in GSD have been addressed. Persson et al. [74] has developed a framework for risk management in distributed software development. These studies have the limitation of being either literature-based or consider only certain cases for identifying the risks in DSD. Mudumba and Lee [63] states that risks in DSD occur due to multiplicity in the location, culture standards and technologies and proposes an agile project management framework to control the risks. This work addresses the risks at abstract level and does not validate the suggested framework. A need to explore the issues arising in distributed software development from the vendor's perspective has also been felt [78]. Challenges in shared decision making in agile development, which includes misalignment of strategic product plans and iteration plans, problems related to resource allocation and performing development tasks have been discussed in an empirical study by [61].

Specifically, in the area of distributed agile development, a conceptual framework on risk in distributed scrum is created by Hossain et al. [42]. This framework is based on systematic literature review and it identifies the key risks in GSD while using Scrum and also suggests mitigation strategies for each risk. As per the author, "The work is not comprehensive, does not include inputs from industry and is based purely on the existing literature review."

Works of Jiménez et al. [44] and Da Silva et al. [23] consolidated the findings from the studies on risks in managing distributed development projects published in quality journals and periodicals. These studies focus on risks related to communication and collaboration amongst stakeholders, project management and infrastructure resulted in missing out many other areas of concerns like risks in software development life cycle, risks related to customers, other vendors and third party involvement in solution development. Another systematic literature review by Dybå and Dingsøyr [28] discusses about benefits and limitations of agile methods, but it does not take into account the challenges specific to distributed agile development. These studies contributed through an extensive review of literature, but overlooked the experiences of industry practitioners.

Eventually, there is a clear need to explore and identify the risks in distributed agile development (DAD) and design solutions to manage those risks for successful execution of such projects.

3. Research method

3.1. Research objectives

Observations from previous section recognize that the organizations are embracing distributed agile development for creating evolving systems at high speed that fulfills the needs of the dynamic business environments. DAD involves significant risks and the scarcity of the related research [47,43] is an indicative of the need for exploration from the field. Following are the research questions addressed in this study.

1. What are the conditions (risk factors) that pose threat to distributed agile projects?
2. What are the risk management techniques that are used for managing the risk factors?
3. How are the risk factors categorized for facilitating their adoption in practice?

3.2. Research design

This study is based on the interviews of 13 DAD practitioners from 4 companies followed by work document analyses of 28 projects from 13 companies. It spanned for a period of 12 months. The objective of the study was to obtain the practitioners' views on the risk factors in distributed agile projects and frequently used risk management techniques.

In order to carry in-depth interviews, a semi-structured open-ended questionnaire was designed to ensure that respondent feels free to answer whatever is most important in his or her thinking [105]. Before using the questionnaire for collection of data, it was checked for the correctness of the questions, their relevance and ability to extract valid information. For detecting the possible weakness in the instrument, it was sent to 10 senior professionals and academicians working in the area of the proposed research. Four of those gave their explicit views on the questionnaire, their understanding of each question and to what extent they think that an appropriate response for that question can be obtained. Most of the views obtained on the quality of the questionnaire were converging and we incorporated the given suggestions. The questionnaire (Appendix A) was hence improved and used for conducting the interview based exploration.

We used purposive sampling [19] for data collection, wherein, the sample conforms to the criterion that the subjects are industry practitioners, who have managed or worked or are working on DAD projects. This sample included program directors, project managers, delivery managers, test managers, developers, agile coaches, scrum masters and team members having experience of 10–20 years in IT development and minimum 3 years of experience in handling DAD projects.

The size of employees in the Information Technology (IT) companies we visited for interviewing the practitioners and project work document analyses varied from 1000 to above 100,000. The companies with number of employees less than 1000 were considered as small sized, between 1000 and 10,000 as mid-sized and above 10,000 as large sized companies.

As a part of the survey, predominantly face-to-face interviews (12) and few telephonic interviews (1) were conducted with each interview lasting for 45 min to 1 h. The interview was initiated with a brief explanation on the research to the interviewee. The response was recorded by the researcher in the form of extensive field notes [91]. The researcher added details to the notes based on her understanding immediately after the interview in order to capture maximum information from the interview. When the notes were found to be incomplete, we contacted the respondent again through e-mail or telephone for clarification of any missed or unclear information.

Questionnaire aimed to collect the data relevant to the stated research questions. In addition to enquiring about the observed risk factors and associated management techniques, explicit effort was spent on distinguishing the software development methodology that holds the origin of risk factors. The source cause of the risk may be present in Distributed Agile Development, or Distributed Development, or Agile Development or in traditional software development itself. Respondents categorically spoke that certain risks, which exist in both traditional development as well as in distributed agile development, were more pronounced in DAD. The cause and form of such risks are different and they have high degree of severity in distributed agile projects.

We started data analysis when 80% of the data collection was done and achieved further flexibility in data collection process [91]. This analysis, explained in detail in the next section, led to the creation of initial categories holding the identified risk factors.

The review of research literature for every risk factor and associated categories started in parallel with the data analysis. That gave researchers a strong basis for explaining the risk factors in DAD projects and also helped to identify those risk factors which have weak visibility in the current literature.

This was followed by the work document analysis [11] of 28 DAD projects. We achieved triangulation by combining document analysis with interviews and ensured the convergence of the observed facts [104]. The sessions were conducted in multiple branches of 13 different IT companies located in cities within and outside India at company's site (except in two cases) as the practitioners were reluctant to handover the documents for the reason of confidentiality. Sessions were supported by the simultaneous discussions with the project manager or the scrum master who was involved in managing the project. This minimized the limitations caused by the low retrievability of the documents [33]. The work documents used for this purpose were risk registers, retrospectives, and sprint backlogs, documents in tools, e-mails, status reports and statement of work of specific projects. For two cases, documents were received via email and related queries were addressed through email and telephonic communication. Corresponding to each DAD project document, field notes were made and transcribed based on the document content and discussion.

The findings from in-depth interviews and work document analyses were further consolidated. Certain risk factors were commonly identified by the two processes while other new factors also emerged. These new factors were analyzed for discerning risk categories and then the aforesaid process of finding literature support was repeated.

3.3. Data analysis

The identified risk factors were classified so that the managers can have a collective viewpoint on a group of logically related factors. Risk classification is an economical way of analyzing risks and their causes by grouping similar risks together into classes [71]. In this study, the risk factors, which were logically related according to the knowledge base of the field, were grouped together to form 'Risk Categories'. For example, all the risk factors pertaining to the challenges in testing of code have been grouped under the Risk Category 'Software Development Life Cycle (SDLC)' since testing is an integral part of SDLC. The detailed explanation of the method adopted to create the risk categorization is as follows:

3.3.1. Analysis of in-depth interview data and initial risk factor categorization

The qualitative data collected through in-depth interviews, primarily, was about the risk factors posing threat to DAD projects, their description, causes, source methodology and the most suitable risk management techniques to control the risks. We used constant comparison method for the analysis of qualitative data that has been identified as a method for classic theory generation in qualitative research literature [91]. The choice of this method can be attributed to the provisions it has for continuous review of the qualitative data and the study of evolving pattern through iterative coding, designing and integrating theoretical notions [91,32].

Constant comparison method is concerned with generating and plausibly suggesting (not provisionally testing) many properties and hypotheses about a general phenomenon [32], which in our case, is the Risk Management in Distributed Agile Development (DAD) projects. As per the explanation given by Glaser, "This method involves four stages: (1) comparing incidents applicable to each category (2) integrating categories and their properties (3) delimiting theory and (4) writing theory." As explained by Seaman [91], "It is iterative such that although we proceed from one stage to another, the previous stages remain in operation and continuously contribute to the evolution of data analysis results." We describe that how every stage in the constant comparison method can be mapped to the data analysis process followed by this study.

3.3.1.1. Stage 1: creation of risk categorization matrix. This stage involved constant comparison at two levels. At initial level, the transcription for each interview was read carefully and the relevant pieces of information were extracted to form a risk categorization matrix [73] with dimensions stated in Table 1. From the first set of qualitative response, the data pertaining to relevant dimensions were recorded. The next set of response was then compared with the previous for ascertaining the similarity or differences. For logically resembling risk factors in the existing matrix, new pieces of information on corresponding dimensions were added to strengthen the risk factors further. Newly recognized risk factors were added to the matrix along with their dimensions. The matrix was enriched iteratively with the data collected from every interview [91].

At level two, when the matrix was getting populated with the risk factors and their dimensions, we started coding them for their logical relevance to specific aspects [77] of software development environment like engineering and management processes. Thus, as the matrix was evolving, the identified categories started emerging. Initially, mention of an individual risk factor was coded as a category then every new such factor was compared to the previous one to strengthen the existing category or to create new categories. Such fracturing of qualitative data facilitated the comparison between the risk factors in the same category or between different categories [91] and generated theoretical properties of a category and its dimensions [32].

During the process of coding, the authors sat together and discussed about the risk categorization in order to identify the most logical conclusions. The discussion helped us to re-think on the work as it was progressing, bring the points missed, and add the relevant points hence, improve upon the evolving results [32].

3.3.1.2. Stage 2: integration of risk categories. Iterative coding based on constant comparison of the cases in previous stage led to the segregation of risk factors in categories based on diverse properties. As a result, moving of factors amongst the risk categories; adding, deleting or merging of risk categories happened in some cases

Table 1
Risk categorization matrix (Dimensions for segregating the incidents/cases mentioned in responses).

Sr. No.	Dimension for risk categorization	Description
1	Risk factor title	Provisional name given to the risk factor. Example: Unavailability of Requirements Documents for Testing
2	Risk explanation	A brief description of the risk factors and their effect on the project success as stated by respondents Example: Agile teams do very less documentation as compared to traditional teams. This creates significant risks to testing since testers use requirements specifications and other documents which form the baseline for evaluating subsequent products and processes (design, testing, verification and validation activities) and for change control
3	Cause of the risk	Indicates the reasons for that risk to occur
4	Source of the risk	Refers to the respondent's view on the origin of the cause of the risk factors as traced back to the software development methodology used. The risk may emanate from the following sources: • The combination of distributed development and agile development (distributed agile development (DAD)) • The usage of agile development (ASD) • The stakeholder are geographically dispersed (Distributed Software Development (DSD)) • The risk which will be there in any software development approach (Traditional software development (SWD)) Example: In reference to the above given risk factor example, the testers get insufficient information for testing. The problem becomes severe in case of distributed agile teams due to geographic separation of team members. When developers and testers are non collocated, the testers rely on their own assumption for carrying the testing of the system being developed
5	Risk management techniques	Refers to the methods that the practitioners use to control the risks as stated by them This includes four basic risk management techniques described below: • Risk mitigation (controlling or reducing the impact of the risk) • Risk avoidance (selecting an alternate course of action where the risk is no longer a possibility) • Risk transference (getting another party to share the consequences of the risk) • Risk acceptance (no proactive action will be taken; a contingency plan will be created)
6	Literature support	Refers to the set of research papers, which relate to the risk factor identified from interviews. This helped later in filtering out the risk factors faced by the practitioners in managing DAD projects, but are not been noticed by the existing research studies

during the analysis. For example, we found that risk factors related to the risk category 'Multiple Vendor Involvement' and that of 'Customer Collaboration' could be integrated since both vendors and customers shared a common property of being entities external to the development organization with whom extra effort has to be taken to reduce the communication gaps. Hence, we created a single category named as 'External Stakeholder Collaboration'. Since data collection overlapped with data analysis, the matrix underwent multiple revisions with the iterative identification of new risk factors and categories before building a basic risk factor categorization.

3.3.1.3. Stage 3: formation of concrete risk categories. The third stage of constant comparison method integrates related properties into categories with an objective of consolidation and subsequent reduction. Formation of concrete risk categories by grouping the risk factors can be referred to as the 'Delimitation of Theory', which involves reduction of categories and generalization enforced by constant comparison. For example, a group of risk factors corresponding to various theoretically pre-defined and known activities of software development like requirement engineering, designing, coding, testing and integration of code are logically related based on the theoretical literature and field of practice formed the risk category, namely, 'Software Development Lifecycle' [77]. Other risk categories that emerged from analysis have been, 'Project Management', 'Group Awareness' and 'External Stakeholder Collaboration' and 'Technology Setup'.

This iterative process involving reading and re-reading of the transcripts, modifying the matrix and the appropriate coding of elements in hierarchy helped us to provide significant knowledge from the exploration [59]. The end result was a comprehensive categorization of risk factors along with other dimensions.

3.3.2. Project work document data analysis

In addition to exploratory interviews, we used 'Project Work Document Analysis' as a supplementary technique for exploring risks in DAD projects.

"Document analysis involves skimming (superficial examination), reading (thorough examination), and interpretation [33]." It combines elements of content and thematic analyses iteratively. Content analysis organizes information into categories related to the central question of the research while thematic analysis involves pattern recognition within the data so that the emerging themes can be used as the categories for further analysis [30]. Together, they can be mapped to the first stage of constant comparison method, which involved risk factor identification and coding of incidents for a category. For example, one 'Status Report' stated, "*Irregular stand-ups: developers do not meet regularly in stand-ups. They (offshore) are not keen to share the updates. They use e-mails for communication only. This happens due to some of the developers located at the client location and others in India.*" This was indicative of the problem in conducting the essential stand-ups daily due to the geographical distance between teams. When compared with the pre-defined codes and categories in the 'initial risk categorization matrix', created from the in-depth interview data, one observation from interview echoed the view – "*Various teams were not following appropriate scrum practices. They had their own methods which were hybrid of scrum and waterfall. While, some teams were not having stand ups daily and others were having but the stand-ups were not effective*". In this way, the issue of irregular stand-up meetings mentioned by the respondent got seconded by the concern raised in status-report.

Analysis of the qualitative data obtained from the project documents was carried in parallel to the data collection of work document as suggested by Coffey and Atkinson [16]. Strengthening of risk categorization matrix and simultaneous updation of codes resulted in repetition of certain patterns and the situation when little more is gained by further data collection leading to the saturation of data [86]. This is similar to the stage 3, i.e., 'Delimiting the Theory' of constant comparison method, wherein, the categories are theoretically saturated and the researcher delimits the original list of categories for coding. After this point, we stopped collecting more cases and performed only re-reading of the data collected from in-depth interview and project work documents in order to further improve upon the risk categorization.

Since, we collected risk factors for DAD projects from multiple sources (in-depth interviews and work documents) we have explicitly shown the source from where we obtained the risk information in Tables 4–8.

3.3.3. Theoretical framework

3.3.3.1. Stage 4: mapping risk categories to Leavitt's model. Coding of data, categorization and identification of major themes was followed by "Writing Theory", which is the fourth stage of the constant comparison method and have roots or fill gaps in existing theories.

Inclusion of DAD risks in organizational context involves the changes in organizational components and their inter-relationships and so, we need to look at the models for organizational change that demonstrate "what to change and how to do it [12]." We assessed the suitability of four models of organizational change, namely, Leavitt's model, Weisbord's six-box model, Nadler-Tushman Congruence model and Tichy's framework and found that the components of Leavitt's model could easily be mapped to the risk categories extracted in this study. Although, Leavitt's model represents only the transformation part and not inputs or outputs of organism, it has gained significant acceptance in organization theory [90] as well as in information systems [49]. DAD risks are relevant to the transformation part of the organization and so are closely mapped to the four components of Leavitt's model, namely, task, structure, actor and technology [54], shown in inner boxes with dashed lines in Fig. 1. Component 'Task' defines the relevance and benefits of the type of tasks an organization is supposed to do for achieving the goals of delivering the product and/or services. 'Structure' deals with the communication, authority and workflow systems of organization while 'Actors' refer to the people who perform the tasks and 'Technology' to technical means and tools employed for doing tasks. The arrows connecting these components represents the interactions between them.

Lyytinen et al. [58] established a connection between the Leavitt's model and software risks. They argued that a change in any Leavitt's component in a system development process would create disturbances that may lead to failure in software change or associated loss. Such risky incidents have been found to fall into one or more Leavittian elements and their connections.

The components of Leavitt's model can easily be translated into well known elements of software development. 'Task' signifies the activities for deriving expected outcomes in terms of goals and deliverables. 'Actors' signify all stakeholders including users, managers and designers. 'Structure' denotes the project organization and other institutional arrangements related to communication, authority and workflow. 'Technology' means development tools, methods and hardware and software platforms.

In this study, the risk categories that emerged after performing qualitative data analysis were mapped to the components of Leavitt's model. Besides the fact that our risk categories align very well with the components of Leavitt's model, the ability of the Leavitt's model to accommodate the most essential aspects of software development and its extensive use in the information system literature [58], motivated us to use it as a base for our risk factor categorization.

3.3.4. Analysis of the conflicting DSD properties and agile principles of the risk factors

It was observed that risks in DAD projects occur primarily because distributed development and agile practices differ from each other in their key tenets [80]. Agile methods use more informal approach for coordination, while distributed development needs more formal processes and hence, do not align with each other. In this section, we identify the DSD properties, which are in contrast with the principles and practices of agile methods, leading to significant challenges in DAD projects. Later, for each risk factor identified, we will be identifying the properties of DSD and that of agile methods, which together pose risk to DAD project.

3.3.4.1. Properties of Distributed Software Development (DSD). Following properties of distributed development come to light from the responses of professionals that directly or indirectly cause risks in distributed agile development:

1. **Spatial Distance:** In traditional, co-located projects, teams are able to coordinate well and are able to build up a shared view of their work. On the contrary, spatial distance limits face-to-face communication, frequency of travel and weakens communication and team collaboration [36].
2. **Temporal Distance**: Temporal distance is related to the differences in the time-zones in which the multiple teams work in distributed development environment. It makes multisite virtual meetings hard to plan and increases the complexity of coordination and communication [29].
3. **Language Barrier**: Language Barrier arises in cross national projects when sites and participants do not share a common language or norms of communication. This results in misinterpretation and poor communication of information between the teams [38,85].
4. **Development/Work Culture**: Difficulties in distributed development arise when sites are different in terms of team behavior, perception of authority, hierarchy, planning, punctuality and organizational culture [38,50]. Further, the teams may have different development tools, processes and practices [36].
5. **Large Project Scope**: Distributed development projects tend to be larger, more complex and have more contributors [44]. This multiplies the difficulty in controlling defects in product and there is an apparent tendency of the work going out of control [22].

Table 2
Agile principles and the corresponding keywords.

No.	Agile principles	Keyword
1	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software	Customer Satisfaction
2	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage	Embrace Change
3	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale	Frequent Software Delivery
4	Business people and developers must work together daily throughout the project	Customer Involvement
5	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done	Team Motivation, Training and Support
6	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation	Face-to-Face Communication
7	Working software is the primary measure of progress	Working Software
8	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely	Sustainable Pace
9	Continuous attention to technical excellence and good design enhances agility	Technical Excellence
10	Simplicity—the art of maximizing the amount of work not done—is essential	Simplicity
11	The best architectures, requirements, and designs emerge from self-organizing teams	Self-Organized Teams
12	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly	Inspect and Adapt

Table 3
Relevant agile practices applicable for DAD risks identified and corresponding keywords.

No.	Relevant agile practices applicable on identified DAD risks	Keyword
1	The size of the team should be seven people, plus or minus two [89]	Small Teams
2	As the team grown in capacity, keep its workload constant but gradually reduce its size. This frees people to form more teams [6]	Shrinking Team
3	Develop an open space big enough for the whole team. Meet the need for privacy and "owned" space by having small private spaces nearby [6]	Sit together
4	Osmotic communication means that information flows into the background hearing of members of the team, so that they pick up relevant information as though by osmosis. This is normally accomplished by seating them in the same room [15]	Osmotic Communication
5	Make your workspace about work such that the project status and progress can easily be observed. Teams use spatially sorted story cards and big visible charts for this [6]	Informative Workspace
6	Development work happens in time-boxed cycles typically of 1–4 weeks [94]	Short Iterations
7	The system is put into production in a few months, before solving the whole problem. New releases are made often—anywhere from daily to monthly [94]	Small Releases
8	The team gets feedback much more frequently, which helps them to identify the cause for their success or failure and further refine their plans [94]	Frequent Feedback
9	When customers work out requirements incrementally, programmers are able to work on established user stories while customers figure out the details for future stories [94]	Incremental Requirements
10	An evolving prioritized queue of business and technical functionality that needs to be developed in the system [89]	Single Product Backlog
11	A subset of Product Backlog that is selected for a Release [89,17]	Release Backlog
12	A customer sits with the team full-time [7]	Onsite Customer
13	Customer Representative (product owner) is a single person, not a team of members. This helps to avoid conflicts in the content and priority of the requirements [89]	Single Customer Representative
14	At the end of the sprint, the team gets together with the management to inspect the product increment the team has built [89]	Product Demonstration
15	Is a short, simple description of a feature told from the perspective of the user of the customer of the system. User stories are often written on index cards or sticky notes and are arranged on walls or table to facilitate planning and discussion [17]	User Stories
16	In every sprint, 5–10% of the time should be spent to review, cleans and re-organize the product backlog [17]	Product Backlog Grooming
17	Write an automated test, then write just enough code to make that one test pass. Build the code in small test-code-test increments. This leads to development of good quality code which provides functionality close to the customer needs [21]	Test Driven Development (TDD)
18	All production code is written by two people at one screen/keyboard/mouse [7]	Pair Programming
19	New code is integrated with the current system after passing all the tests after no more than few hours [7]	Continuous Integration and Testing
20	The design of the system is evolved through transformations of the existing design that keep all the tests running [7]	Refactoring
21	Agile teams need to heavily rely on test automation in order to deliver value quickly [17]	Test Automation
22	Everyone shares responsibility for the quality of the code. No single person claims ownership over any part of the system, and anyone can make any necessary changes anywhere [94]	Collective Code Ownership
23	Scrum projects do not have up-front design phase; all work occurs with repeated cycles of sprints. Design is both intentional and emergent [17]	Incremental Designs
24	At every moment, the design runs all the tests, communicates everything the programmers want to communicate, contains no duplicate code, and has the fewest possible classes and methods [7]	Simple Designs
25	Team should follow coding standards which acts as guidelines, to which all developers agree to adhere when programming [94]	Coding Standards
26	The scrum team meets daily for a short status meeting to review the team progress and identify the impediments for removal by management [89]	Daily Stand-ups
27	Maintain only code and tests as permanent artifacts. Generate other documents from the code and tests. Any artifacts contributing to these sources of value are themselves valuable. Everything else is waste [6]	Minimal Documentation
28	Is a common artifact that defines the list of things that must be true for a backlog item to be considered as finished [87]	Definition of Done
29	Keep effective teams together. Value on software is created not just by what people know and do but also by their relationships and what they accomplish together [6]	Team Continuity
30	The team possesses all the skills needed to produce a quality code that delivers the features needed by the organization. Any task might be completed by any team member, or a pair of members [21]	Whole Team
31	Leadership in agile teams is meant to be light-touch and adaptive providing feedback and subtle direction [4]	Servant Leadership
32	Is a meeting held at the end of the sprint by the team in order to review how the team worked during the previous sprint and make improvements to its processes and practices [87]	Retrospective Meetings

3.3.4.2. Principles of agile software development. We referred to the twelve principles and practices of agile and have related the risk factors to them. Table 2 lists twelve principles of agile development as given in the 'Agile Manifesto' [15], along with shortened keyword for their identification and further reference. Agile practices, which can be derived from agile principles, are also considered separately since they have an impact when combined with DSD properties. These practices are enlisted in Table 3.

4. Results and discussion

Qualitative analysis of interview and project document data resulted into identification of five risk categories, namely, 'Software Development Life Cycle', 'Project Management', 'Group Awareness', 'External Stakeholder Collaboration' and 'Technology Setup', that could be mapped to the four components of Leavitt's model. Following section explains the categorization and mapping.

4.1. Categorization of risk factors for distributed agile development

One of the five derived risk categories is 'Software Development Life Cycle' that comprises the risk factors related to various activities of software development like customer specification of requirements and planning, modeling, construction and deployment of software application [77]. This category maps to the 'Task' component of the Leavitt's model as the risks are related to the activities performed for delivering product or service to customers.

Distributed agile teams require intensive communication, coordination, collaboration and trust amongst the team members and other stakeholders, which forms important elements of 'Group Awareness' [34,84]. Risks arising due to the lack of group awareness and trust have been discussed under this category. It maps to the 'Structure' component of Leavitt's model as it deals with the communication and authority systems.

Along with software development activities, 'Project Management' related tasks like project planning, project organizing,

Table 4
DAD risk factors and risk management methods for SDLC risk category ('Task' in Leavitt's model).

Risk factor	Risk management method	DSD properties	Related agile principles	Related agile practices	Source
Unclear objectives of project [103,47]	Product owner to discuss the vision of the project with the team in sprint planning meetings [103]; Periodic co-location of team to get common vision [103,99]	Spatial Distance	Customer Satisfaction. Customer Involvement. Face-to-Face Communication. Inspect and Adapt	Onsite Customer. Single Customer Representative. Product Demonstration. Frequent Feedback	I
Requirement unclear to the team [25,63]	Facilitate white-board sessions and group discussions; Show demonstrations to the client; Record the minutes of meetings with the customer; Performing acceptance test after each build; Usage of richer communication medium [25]	**Spatial Distance** Temporal Distance	Customer Satisfaction. Customer Involvement. Face-to-Face Communication. Inspect and Adapt	Single Customer Representative Onsite Customer Small Teams. Sit Together. Frequent Feedback. Incremental Requirements	I,D
Requirements conflicts amongst multiple product owners [45]	Daily meetings of product owners to resolve the conflicts [103]	**Large Project Scope** Spatial Distance Temporal Distance	Customer Involvement. Customer Satisfaction	Single Customer Representative	D
Inadequate communication about end-user requirements [25,13]	Team members visit end-user location [13,25]	Spatial Distance	Customer Involvement	User Stories. Frequent Feedback	I
Inadequate Prioritization of Requirements [17,52,40]	Backlog grooming to reduce project scope by story re-prioritization; Consolidate multiple product backlogs into one; Have a separate backlog for each team working on large independent component. A single overall backlog can be maintained to guide the team about the overall product features and the interdependencies involved [17,103]	Large Project Scope	Customer Satisfaction. Customer Involvement. Face-to-Face Communication. Embrace Change	Single Customer Representative. Onsite Customer. Single Product Backlog. Incremental Requirements. Product Backlog Grooming	D
Frequent Architectural Changes [8]	Builds flexible designs to cope up with the frequent requirement changes. Representatives from feature teams conduct 'Joint Design Workshops' repeatedly to resolve broad architectural issues [52]	**Spatial Distance** Temporal Distance	Embrace Change. Technical Excellence	Incremental Designs. Test Driven Development. Refactoring	D
Inconsistency in design standards of distributed teams [80]	Balance team skills by including architects/ designers at each location; Form architectural board to resolve design conflicts; Split Sprint planning meeting to discuss design issues [103]	Work/Development Culture	Face-to-Face Communication	Sit Together. Incremental Designs. Minimal Documentation	D
Poor technical debt management [5]	Repaying technical debts as the first class item on product backlog; Spend at least 10% of the iteration time on technical debts; Having an independent testing to remove the technical debts [17,94] **RA**: Architecture Envisioning and Architecture Prototyping [5]	**Spatial Distance** Temporal Distance	Technical Excellence. Working Software. Simplicity. Sustainable Pace	Refactoring. Simple Designs. Coding Standards	I, D f
Issues with pair programming [41]	Create pairs of developers carefully to ensure compatibility; Use of good tools to implement pair programming [39]	**Spatial Distance** Temporal Distance	Technical Excellence. Team Motivation, Training & Support	Pair Programming. Sit Together	D
Unavailability of requirements documents for testing [70]	Have onsite representative from vendor at client location; Involve testing team in the project right from the beginning; Use of exploratory testing as it needs less documentation [20]	**Spatial Distance** Temporal Distance Language Barrier	Face-to-Face Communication. Working Software	Minimal Documentation. Incremental Requirements. Test Driven Development	D
Cross Functional teams insufficient for Testing of Large Projects* [3]	Independent test team augments developers for complex testing [3]	Large Project Scope	Frequent Software Delivery. Working Software. Self-Organized Teams	Short Iterations. Test Automation. Whole Team	I
Test Data Management* [10]	Automate test data management by using appropriate tools [5]	Large Project Scope	Frequent Software Delivery. Technical Excellence	Short Iterations. Test Automation	D
Code integration across multiple sites [93,38]	Teams clarify the interface requirements for product integration [38]	**Temporal Distance** Spatial Distance	Face-to-Face Communication. Working Software. Frequent Software Delivery	Short Iterations. Small Releases. Continuous Integration and Testing. Minimal Documentation	D
Loosing on Time on End-to-End extensively Interdependent Transaction rich Test cycle*	Divide the transaction test cycle in small phases. Execute small batches of transactions sets such that the phases of test cycle of each transaction set can be overlapped to bring in parallelism; Use automated testing in certain test cycle phases; Utilize time zones which are exactly overlapping to reduce test cycle lag time	Temporal Distance	Working Software. Frequent Software Delivery	Continuous Integration and Testing. Short Iterations. Test Automation	D
Ineffective standup meetings [8]	Well trained scrum master for effective scrum meeting; Have a larger time-box (45–60 min) instead of 15 min [103]	Spatial Distance. Temporal Distance. **Language Barrier.** Work/Development Culture	Face-to-Face Communication. Inspect and Adapt	Daily Stand-ups	I, D

Table 4 (*continued*)

Risk factor	Risk management method	DSD properties	Related agile principles	Related agile practices	Source
Differences in agile practices and standards of processes followed by multiple teams [81,93,47]	The core team decides upon standards practices to be followed; Document the interactions and meetings between the teams; Teams share their daily work through presentations; Raise issues about teams not following common practices in retrospective meetings [81]	Work/Development Culture	Face-to-Face Communication. Technical Excellence. Inspect and Adapt	Definition of Done. Coding Standards. Retrospective Meetings	I
No common definition of done [82]	Distributed agile teams need to have a common definition of done for the story, for the iteration, for the release, for the project [82]	Work/Development Culture	Working Software. Technical Excellence	Definition of Done	
Difficulty in system release management and deployment* [55]	Frequent and periodic release dates; Keep infrastructure for component integration ready; Keep functionality flexible if dates are constrained [55]	**Temporal Distance.** Work/Development Culture	Working Software. Frequent Software Delivery	Short Iterations. Continuous Integration and Testing. Small Releases	D

Table 5
DAD risk factors and risk management methods for project management risk category ('Task-structure interdependencies' in Leavitt's model).

Risk factor	Risk management method	DSD properties	Related agile principles	Related agile practices	Source
Difficult to execute fixed price projects [31]	**RA**: Use time and material contracts instead of fixed price model [31]; In case the project is Fixed Priced: Team must take extra efforts for clarifying the scope. Collaborate with the customer to keep the scope variable, if cost and schedule are fixed, while ensuring high quality. Communicate with the customer and involve them for providing the most valuable feature in the earlier release. Identify the low priority features that can be dropped, if needed [72]	**Spatial Distance** Temporal Distance Language Barrier	Customer Satisfaction. Embrace Change. Customer Involvement. Face-to-Face Communication	Small Releases. Incremental Requirements. Product Backlog Grooming. Onsite Customer	D
Lower initial velocity [18,99]	Invest in travel of team members, especially the senior members [99]	**Spatial Distance** Temporal Distance	Face-to-Face Communication. Team Motivation, Training and Support. Sustainable Pace	Sit Together. Small Teams	I
Working with component teams instead of feature teams [52]	Use feature teams at each site who can deliver end-to-end solution [52]	Spatial Distance	Self-Organized Teams. Working Software	Whole Team	I, D
Growth in team size or development sites [35]	Elaborate training plans should be made to help new members	**Spatial Distance** Temporal Distance	Face-to-Face Communication. Team Motivation, Training and Support	Sit Together Small Teams. Shrinking Teams. Team Continuity	I
Team reorganizing in every sprint [52]	Majority of team members should be core members; Assigning buddy or mentor to new team members to speed them up [52]	**Spatial Distance** Temporal Distance	Face-to-Face Communication. Self-Organized Teams. Team Motivation, Training and Support	Small Teams. Team Continuity	I, D
Higher Inter-Dependencies between the Teams [18,37]	**RA**: Write user stories such that teams can develop independently; Team identifies potential dependencies during Scrum-of-Scrum and brings them in Sprint planning. [103]	**Spatial Distance** Temporal Distance. Language Barrier Large Project Scope	Face-to-Face Communication. Working Software	Small Teams. Sit Together. Product Backlog Grooming	I,D
Lack of uniformity in multisite team's capabilities [20,47,17]	Team is a good mix of technically strong and weak members; Encourage team to be cross-functional; Build a good knowledge sharing environment like document management system, community of practice; Agile Coaches provide training to the team members to build their skills. [17,47]	Work/Development Culture	Team Motivation, Training and Support. Self-organized Team. Inspect and Adapt	Informative Workspace. Whole Team	I
Unavailability of business analyst*	Separate business analyst for each team; Swap business analyst amongst multiple teams [102]	Work/Development Culture	Customer Involvement. Self-organized Teams	Onsite Customer. Incremental Requirements. Whole team	D

project staffing, project directing and control [14] aim to achieve the goals of software development project [65]. This risk category maps to the 'Task-Structure Interdependencies' of the Leavitt's model as it involves risks due to interactions between development activities and the institutional arrangements including workflow, communication and authority systems.

In agile development environment, otherwise external players like customers and vendors play the role of internal actors for their direct and continuous contribution to the development of software. In DAD projects, risk factors related to the customers, third party developers and vendors have been put together under the risk category 'External Stakeholder Collaboration'. Further, authors assert that the risk factors related to such external people, maps to the 'Actor' component of Leavitt's model.

Distributed agile teams use many tools for executing rigorous engineering practices in a distributed environment [75]. Risk factors which arise due to inappropriate tool usage are grouped in the risk category 'Technology Setup', which are related to the

Table 6
DAD risk factors and risk management methods for group awareness risk category ('Structure' in Leavitt's model).

Risk factor	Risk management method	DSD properties	Related agile principles	Related agile practices	Source
Lack of communication between the team members [74,42,64]	Team members use tools like video conferencing and desktop sharing; Use of scrum-of-scrum meetings between teams; Use of tools which support formal and non-verbal communication [42 98]	**Spatial Distance** Temporal Distance Language Barrier	Face-to-Face Communication	Sit Together. Osmotic Communication. Small Teams. Informative Workspace	I,D
Poor communication skills in the team members [62,44]	Impart training to team members to improve their communication skills [62]	**Work/ Development Culture**. Language Barrier	Face-to-Face Communication Customer Involvement. Team Motivation, Training and Support	Product Demonstrations. On-site Customer	I
Uncommon Language [29,47]	Organization to introduce language trainings for the team members; Establish English as the official language in the organization; Use of tools which support spell checkers and translators [29]	**Language Barrier** Spatial Distance Work/ Development Culture	Face-to-Face Interactions. Customer Involvement	Sit Together. Onsite Customer	D
Lack of Documentation [80,47,50]	Team create minimum amount of documentation and check in regularly in the version control system [42]	**Spatial Distance** Temporal Distance	Face-to-Face Communication. Working Software	Minimal Documentation. User Stories. Informative Workspace. Test Automation. Test Driven Development	I, D
Lack of communication between team and the client [80,99,47]	Team to communicate with the customer at least once in a week; Make customer a part of the standup meetings; Make travel plans for product owner to visit offshore locations to meet the teams [99]	**Spatial Distance** Temporal Distance. Language Barrier	Customer Involvement	Onsite Customer. Single Customer Representative. Product Demonstration	I, D
Unsuitability of agile approach for large organization [9]	Combine agile and traditional methods to resolve this problem; Formation of self-organizing communities called Community of Practice [52,17]	Large Project Scope	Face-to-Face Communication. Customer Involvement. Self-Organized Teams	Small Teams. Sit Together. Team Continuity. Onsite Customer. Single Customer Representative. Whole Team	D
Underinvestment on travel by the management [17]	Teams to insist on collocation at least for the initial iterations; Management include travel expense in the overall budget of the project [74,95]	Spatial Distance	Face-to-Face Communication	Sit Together. Small Teams Onsite Customer	I, D
Poor Coordination between Multiple Teams [17,42,47]	Use of Scrum-of-Scrum between multiple teams [68]	Spatial Distance **Temporal Distance**	Face-to-Face Communication. Self-Organized Teams	Sit Together. Small Teams. Whole Team. Informative Workspace	I
Poor collaboration between different sites [42,43,93,85]	Scrum-of -scrum meetings should be conducted regularly between multiple teams; Train the team members for effective communication and collaboration [47]	**Spatial Distance** Temporal Distance Work/ Development Culture	Face-to-Face Communication. Self-Organized Teams. Team Motivation, Training and Support	Osmotic Communication. Sit Together. Daily Stand-ups. Whole Team	I
Lack of collaboration between developers and quality assurance members [17]	Testers should be full-fledged members of the team; Provide trainings to developers on Test driven development, pair programming while testers should get trainings on pair testing; Automation to enhance collaborations; Involve the testers in the project activities from the beginning [17,52]	Work/ Development Culture	Self-Organized Teams. Face-to-Face Communication. Team Motivation, Training and Support	Whole Team. Sit Together	I, D
Lack of Trust between the Client and the Offshore Team [64,102]	Management allows team members to visit client location; Use of richer means of communication like video conferencing [42]	**Spatial Distance** Work/ Development Culture	Face-to-Face Communication. Customer Involvement	Sit Together. Onsite Customer	I
Lack of trust between onshore and offshore team [80,102,17,47]	Co-locate the team at least for initial iterations to build trust; [17]; Having a 'Traveling Ambassador', who can make short and frequent visits to 'other' locations helps to build trust [17]	**Spatial Distance** Work/ Development Culture	Face-to-Face Communication. Team Motivation, Training and Support	Small Teams Sit Together. Team Continuity	I, D

'Technology' component of the Leavitt's model. The mapping of five risk categories to the components of Leavitt's model is shown as boxes with thickened lines in Fig. 1.

The risk categorization created by analyzing the qualitative data pertaining to risks in DAD projects as obtained through in-depth interviews and work document analysis is presented in this section. Tables 4–8 summarize the risk factors under specific risk category, along with the respective risk management methods suggested by practitioners and the citation of literature to support both.

Finding a support in literature for a practitioner's view was an interesting and educating experience for us and the outcome listed would help the industry and academia as well. As mentioned earlier, research studies have stated risk factors specific to agile or distributed software development separately along with a few others that have scanty mention of the risk factors in distributed agile development. The risk factors which have very less evidence in the literature related to DAD project, have been marked with an asterisk (*) and have been boldfaced as well, in Tables 4–8.

Corresponding to each risk factor, we also present the DSD property and agile principles and practices which are in contrast with each other. The identification of these properties for each risk factor is based on the understanding about that risk factor obtained through the consolidation of in depth-interviews, work document analysis and review of literature. In case of certain risk factors, there are multiple DSD properties which are conflicting with agile principles, as can be seen in Tables 4–8. In such cases, we have bold-faced the DSD property, which holds the primary responsibility for the occurrence of the risk factor.

In addition to DAD risk factors, Tables 4–8 also convey the methods used by the practitioners for managing risks. The stated risk management methods can be mapped to the high level abstractions concerning the four cornerstones of project

Table 7
DAD risk factors and Risk management methods for external stakeholder collaboration risk category ('Actor' in Leavitt's model).

Risk factor	Risk management method	DSD properties	Related agile principles	Related agile practices	Source
Unavailability of product owner [80,17,45,47]	Encourage product owner (PO) to be available for team interactions; **RP**: Surrogate product owner to interact with PO on the client side; The team improves quality of solution in the absence of PO [45]	**Spatial Distance** Temporal Distance	Customer Involvement	Onsite Customer. Single Customer Representative	D
Poor Coordination between Multiple Vendors* [84,26]	Having a 'Distributed Agile Governance' team working with common set of parameters [5]	**Work/Development Culture** Spatial Distance	Working Software. Frequent Software Delivery Customer Involvement	Short Iterations. Small Releases. On-site Customer. Small Teams	D
	Client must allocate work to vendors with minimum dependencies				
Inappropriate user story estimates by multiple vendors* [26]	Multiple vendors use common standards for user story estimation	Spatial Distance Temporal Distance Language Barrier **Work/Development Culture**	Working Software. Frequent Software Delivery	Single Product Backlog. Product Backlog Grooming. Short Iterations. Small Releases	I
Risk in code integration with multiple vendors* [55]	Good communication between vendors to reduce integration problems	**Spatial Distance** Temporal Distance Large Project Scope	Working Software. Frequent Software Delivery	Short Iterations. Small Releases. Continuous Integration and Testing	I, D
Dependency on third party* [57]	Involve third party while making release plans; Make the product owner aware of such problems beforehand. **RA**: Contract with third party should include schedule constraints	**Spatial Distance** Temporal Distance	Working Software. Frequent Software Delivery	Short Iterations. Small Releases	D

Table 8
DAD risk factors and risk management methods for technology setup risk category ('Technology' in Leavitt's model).

Risk factor	Risk management method	DSD properties	Related agile principles	Related agile practices	Source
Lack of communication infrastructure [74,27,42]	Provide the team with multiple, rich sources of communication [69,5]	Spatial Distance	Customer Involvement. Face-to-Face Communication	Onsite Customer. Sit Together. Osmotic Communication. Informative Workspace	I, D
Inappropriate tool selection [38,85]	The team to choose the tool after careful tool study and analysis; The team to refer to existing guidelines for tool selection, tool comparison studies [100]	Work/ Development Culture	Working Software. Technical Excellence	Coding Standards. Test Driven Development. Continuous Integration and Testing	I
	Use right tools for engineering activities as well as for enhancing team communication across all the sites [103]				

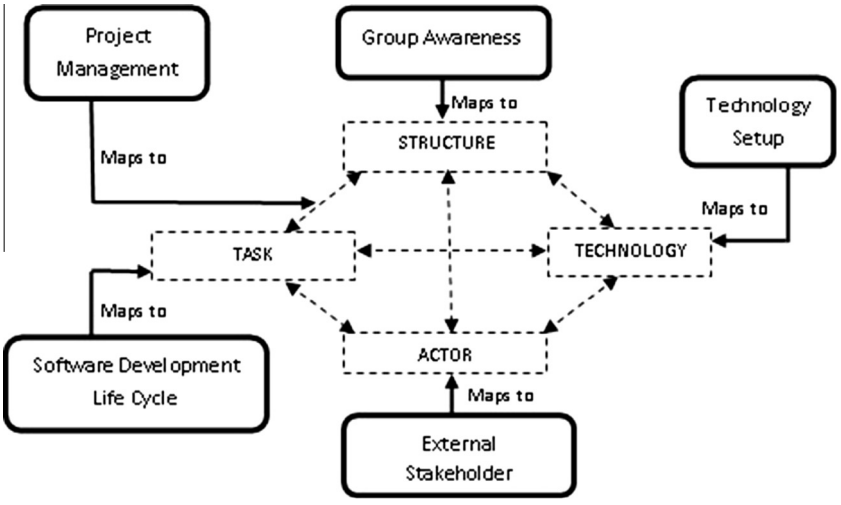

Fig. 1. Mapping of risk categories to the components of Leavitt's model of organizational change.

management, people, process, product and technology. Practitioners suggest different arrangements that ensure timely interactions amongst the distributed team members, customers, product owners, external vendors and eventually, all project stakeholders. Such facilitated communication along with selective documentation allows continuous communication and long-term social exchange, which are elemental to stakeholder management [83]. That further involves customer expectation management, achieved

by resolving the conflict between the scope of the project and pricing mechanism (fixed price versus time and material) [31]. Practitioners have recommended the standardization of practices (*use of common standard for user story point estimation by different vendor*) for facilitating the governance in multi-stakeholders situation, a universal feature of DAD. The other ways of managing risks involve the management of product evolution and quality (*identify potential dependencies in user stories; feature teams*), team skills and capability (*cross-functional team; training; buddy and mentor system*) and product delivery speed (*frequent and periodic release, flexibility in functionality delivered*). The need for underlying automation to support the management of stakeholders, expectations, standardization, product evolution and quality, team skills and capabilities, and product delivery speed (*multiple, rich tools for communication; guidelines for tool selection*) has also been cited. Although, the responses do not hold explicit directives for the senior management, except for the suggestion to encourage travel of team members at distributed locations to enhance trust, but the recommended approaches are insightful for understanding the specific prerequisites of a successful DAD project.

Despite first-hand interactions with the practitioners and project documentation, surprisingly, the responses do not feature the use of metrics for risk management [51]. The reasons for this absence of metrics in data could be the limited use of agile metrics owing to their ongoing evolution [92] or the lack of knowledge about such metrics in the respondents or respondents concealed the relevant information for maintaining the confidentiality. We suspect the latter, considering the importance of metrics for project success and confidentiality issues that were raised at times during the interactions.

Further, the Tables 4–8 also indicate the source of data collection that led to the identification of risk factor and management method. In column 'Source', I signify 'In-Depth Interview'; D indicates 'Work Document Analysis', and (I, D) represent both as the source of data. Along with that, the Tables 4–8 also mention about the 'Risk Mitigation' approaches that can be used to control the risks. For some of the risks, we provide the methods for Risk Avoidance, Risk Acceptance indicated by **'RA'** and **'RP'** respectively.

It is worth mentioning that risks, which occur when multiple vendors are involved in developing systems using distributed agile are not directly addressed in any research study to the best of our knowledge. The impact of involvement of multiple vendors has been discussed in research studies on outsourcing but not with any specific mention of development approach [84]. Also, some specific issues related to testing, integration and release in distributed agile environment remained unnoticed for reasonably long period. This exploratory finding asserts that organizations are involving vendors specialized in some area of software solution development in distributed agile environment. They are experiencing and managing the related risks. More than one respondent in this study suggested the authors to do further research related to multiple vendor involvement in distributed agile projects. Hence, the authors categorically specify that the risk factors with negligible support from literature need serious consideration from the research community.

5. Limitations and threats to validity

The present research has been designed [104] with an objective of eliminating biases and deriving dependable results from exploration. One of the major strength of this work lies in creation of comprehensive set of risk factors in DAD projects and the corresponding risk management methods. Further, the segregation of risk factors in organizational context through Leavitt's model and later, according to the mention in earlier research literature made the findings relevant for the practitioners as well as researchers. However, as a part of the rigorous research process, researchers faced some insurmountable challenges as described below.

First, it was difficult to take verbatim notes while conducting in-depth interview, which may have influenced the correctness and completeness of the collected data. In order to reduce the impact of this problem, majority of interviews that were conducted, involved face-to-face interaction. Moreover, the transcripts of the interviews were sent to the respective respondents after the interview, to ensure that the responses have been understood and interpreted correctly. In order to achieve construct validity, we got the questionnaire used for in-depth interviews validated from ten experts who had managed distributed agile projects.

Second, the unwillingness of the practitioners to provide us with project work documents due to the confidentiality constraint was a threat to the internal validity of the work. Although, we took special efforts to convince the respondents for providing the documents, still, out of 28 projects, the work documents of 18 projects were not handed over to us. In such cases, we had to view them at the respondent's location in limited time and had to hold simultaneous discussion on the project specific risks with the project team members.

Third, the time and budget constraints restricted the scope of data collection for the study to India. This may have an impact on the extent of generalization and hence the external validity [60]. We tried to reduce the impact by collecting data from 13 organizations operating in 4 different cities within India and one city in Australia. As the data was collected using two different approaches and iterative comparison of response contents led to the emergence of the repetitive patterns [32] indicating saturation of categories and inadequacy of further addition of inputs [86], a substantial level of generalization could be achieved.

Hence, considering certain limitations in this exploratory work, efforts have been taken for minimizing the effect and enhancing the reliability and dependability of results.

6. Future scope

Research on risks in DAD projects is in its nascent stage. Present study is an initiation of a comprehensive work in the discipline. Exploratory findings have unveiled many aspects like risk factors, their one possible categorization to facilitate organization's preparation for risk management and respective risk management techniques. Further work may focus on identification of the most critical risk factors and the most appropriate corresponding risk management techniques. As we understand, the findings of study roll out a vast canvas to create knowledge that helps practicing professionals.

7. Conclusion

It is possible to tap into new global markets and make best use of the globally available talent, while potentially reducing the cost by using distributed agile development (DAD) approach. However, these benefits come with various risks like reduced likelihood of project success, increased delivery time, reduced team performance and increased dysfunction [99]. The result of the present exploratory work not only identifies and classifies the risk factors in DAD projects and respective management techniques but also gives an in-depth account of the conflicting properties of agile and distributed development, which eventually are instrumental in producing those risks.

Geographic separation (spatial distance), time-zone differences (temporal distance), language barrier, work/development culture and large project scope are inherent to distributed development. These characteristics, when combined with principles and prac-

tices of agile approach like face-to-face communication, short iteration, continuous integration, customer involvement and others, create unique adversities in DAD projects. Software development fraternity has to be exceptionally apt to find the solutions to the difficulties that have been created or enhanced by DAD approach. The study consolidates some in-practice risk management techniques that have been stated by the practitioners and could help the teams in understanding the feasible approaches of addressing the risks in DAD projects.

Authors contend that the exploration of practitioners' views in this study has rendered some very important insights on DAD projects. Complexity of DAD approach has many layers contributed by the difficulty of software development, agile and then distributed development. Traditional view of managing risks will be inapt to deal with this complexity and so, software engineering community has a mammoth work to understand, organize and formulate the mechanisms that can rule over this new world of software development.

Acknowledgements

We would like to thank all the participants from the corporate who participated in the interview session and gave their valuable inputs for this research work. Our special thanks to the Dr. R Raman (Director, SCIT) for his critical inputs and the support which he has extended to us for carrying the research. This work has been funded by Project Management Institute Educational Foundation, PMIEF (USA).

Appendix A

This questionnaire is created to perform an exploratory survey on Risk Management in Distributed Agile Development (DAD) Projects as a part of the Doctoral work. The aim of this research work is to obtain insights on the risk factors affecting the development of DAD projects as experienced by the practitioners and to understand the strategies used to handle those risks in practice. We request the respondents to share their experiences openly and provide their honest response to the questions. We assure the respondents that their responses will be used for academic research purpose only and the confidentiality will be maintained with utmost care.

Date: **Time:**

Risk Management for Distributed Agile Development Projects - Open-ended questionnaire

Personnel Information

Name: E-mail:
Telephone number: Total Experience:
You're Profile in the company: Total IT Experience:
Name of the company / organization:

Definition of Terms: Agile process model, Software Development Project, Risk Factor, Risk Management, Strategies, status of the project (successful, challenged, failed) *(Provided to the Respondents)*

1. **What type of products/services your company develops/provides?**

2. **Does your organization use any Agile Process Model? If Yes, Please mention broadly, the percentage of projects which were executed using agile process model**
   XP----------
   Scrum ----------
   Hybrid of two or more agile process models ----------
   Any other----------

3. **What is the project team organizational structure (team composition) in general for Agile software development**

____ Product Owner	____ Module Lead	____ Team Member	____ Tester
____ Scrum Master	____ Project Manager	____ Sr. Developer	____ Quality Assurance (QA)
____ Program Manager	____ Project Lead	____ Developer	Any other:

4. **Does your company require a classification or taxonomy of the risk which are involved in executing DAD projects and a guideline for handling those risks?**

5. **What are the major risks (general software development risks) faced by the project managers or the team members while handling DAD projects? Please provide at least 10 major risks associated with Distributed agile projects.**
   SWD: Software development risk
   AG : Agile risk
   DSD : Distributed software development risk
   DAD : Distributed agile development risk

S.No.	Risk item	Source of the Risk (SWD, AG, DSD, DAD)	Cause of the Risk

S.No.	Risk item	Risk management technique used : risk mitigation, risk avoidance, risk transfer, , risk acceptance (contingency plans)

6. **Please quote instance of risk items encountered in a specific project, not usually encountered in most of the project? What were the causes for the occurrence of the risk? What strategy was used to reduce its impact?**

7. **Miscellaneous information about the risks or risk management activities being carried while executing DAD projects? Specific activities carried by Project managers or team members to manage risks?**

Thank you for your participation.

References

[1] S. Ambler, Agile Success Factors, 2012. <http://www.drdobbs.com/architechture-and-design/agilie-success-factors/232601858>.
[2] S. Ambler, Agility at Scale Survey: Results from the summer 2012 DDJ State of the IT Union Survey, 2012. <http://www.ambysoft.com/surveys/stateOfIT Union201209.html> (18.08.13).
[3] S. Ambler, Generalizing Specialists: Improving Your IT Career Skills, 2012. <http://www.ambysoft.com/essays/agileTesting.html on March 2013>.
[4] S. Augustine, Managing Agile Projects, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.
[5] R. Bavani, 10 Principles for success in distributed agile delivery, Cutter IT J. LLC 26 (11) (2013) 30.
[6] K. Beck, C. Andres, Extreme Programming Explained. Embrace Change, Addison Wesley, Boston, 2005.
[7] K. Beck, Embracing change with extreme programming, Computer 32 (10) (1999) 70–77. doi: 10.1109/2.796139.
[8] A. Begel, N. Nagappan, Usage and perceptions of agile software development in an industrial context: an exploratory study, in: First International Symposium on Empirical Software Engineering and Measurement, September 2007, pp. 255, 264, doi: 10.1109/ESEM.2007.12.
[9] B. Boehm, R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Addison-Wesley Professional, 2003.
[10] S. Borghers, W. Demey, The test data management framework, Testing Experience, 2013. <http://testingexperience.com/issues/testingexperience 22_06_13.pdf>.
[11] G.A. Bowen, Document analysis as a qualitative research method, Qual. Res. J. 9 (2) (2009) 27–40.
[12] W.W. Burke, Organization Change: Theory and Practice, third ed., Sage Publication, California, 2011. pg. 190, chapter 9.
[13] S. Chamberlain, H. Sharp, N. Maiden, Towards a framework for integrating agile development and user-centred design, in: Extreme Programming and Agile Processes in Software Engineering, Springer, Berlin Heidelberg, 2006, pp. 143–153.
[14] S. Clegg, D. Courpasson, Political hybrids, Tocquevillean views on project organization, J. Manage. Stud. 41 (4) (2004) 525–547.
[15] A. Cockburn, Agile Software Development, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
[16] A. Coffey, P. Atkinson, Making Sense of Qualitative Data: Complementary Research Strategies, Sage Publications Inc., 1996.
[17] M. Cohn, Succeeding with Agile: Software Development Using Scrum. Pearson Education in South Asia, 2012.
[18] M. Cohn, User Stories Applied: For Agile Software Development, Addison-Wesley Professional, 2004.
[19] D.R. Cooper, P.S. Schindler, Business Research Methods, Tata McGraw Hill, 2006.
[20] G.M. Cottmeyer, The goods and bad of Agile offshore development. in: Proc. AGILE 2008 Conference, IEEE Computer Society, Toronto, Canada, 4–8, August 2008, pp. 362–367.
[21] L. Crispin, J. Gergory, Agile Testing: A Practical Guide for Agile Testers and Agile Teams, Addison Wesley. Pearson Education, 2010.
[22] M.A. Cusumano, Managing software development in globally distributed teams, Commun. ACM 51 (2) (2008) 15–17.
[23] F.Q. Da Silva, C. Costa, A.C.C. França, R. Prikladinicki, Challenges and solutions in distributed software development project management: a systematic literature review, in: 2010 5th IEEE International Conference on Global Software Engineering (ICGSE), IEEE, August 2010, pp. 87–96.
[24] D. Damian, D. Moitra, Global software development: how far have we come?, IEEE Softw 23 (5) (2006) 17–19. doi: http://dx.doi.org/10.1109/MS.2006.126.
[25] D. Damian, D. Zowghi, The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization, in: Proceedings of the IEEE joint International Requirements Engineering, September 2002, pp. 319–328. doi: http://dx.doi.org/10.1109/ICRE.2002.1048545.
[26] S. Dhar, B. Balakrishnan, Risks, benefits, and challenges in global IT outsourcing: perspectives and practices, J. Global Inf. Manage. (JGIM) 14 (3) (2006) 59–89.
[27] B. Drummond, J.F. Unson, Yahoo! distributed agile: notes from the world over, in: Agile, 2008. AGILE'08. Conference, IEEE, August 2008, pp. 315–321.
[28] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: A systematic review, Inf. Softw. Technol. 50 (9) (2008) 833–859, http://dx.doi.org/10.1016/j.infsof.2008.01.006.
[29] J.R. Evaristo, R. Scudder, K.C. Desouza, O. Sato, A dimensional analysis of geographically distributed project teams: a case study, J. Eng. Technol. Manage. 21 (3) (2004) 175–189.
[30] J. Fereday, E. Muir-Cochrane, Demonstrating rigor using thematic analysis: a hybrid approach of inductive and deductive coding and theme development, Int. J. Qual. Methods 5 (1) (2008) 80–92.
[31] T. Franklin, Adventures in agile contracting: evolving from time and materials to fixed price, fixed scope contracts, in: AGILE '08 Conference, 2008, pp. 269–273. doi: http://dx.doi.org/10.1109/Agile.2008.88.
[32] B.G. Glaser, The Constant Comparative Method of Qualitative Analysis, Social Problems, University of California Press, 1965. vol. 12, no. 4 (Spring, 1965), pp. 436–445.
[33] B.A. Glenn, Document analysis as a qualitative research method, Qual. Res. J. 9 (2) (2009).
[34] C. Gutwin, R. Penner, K. Schneider, Group awareness in distributed software development, in: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work, ACM, November 2004, pp. 72–81.
[35] M. He, M. Li, Q. Wang, Y. Yang, K. Ye, An investigation of software development productivity in China, Making Globally Distrib. Softw. Dev. Success Story, Springer, Berlin Heidelberg, 2008 (pp. 381–394).
[36] J.D. Herbsleb, Global software engineering: the future of socio-technical coordination, in: 2007 Future of Software Engineering, IEEE Computer Society, pp. 188–198. doi: 10.1109/FOSE.2007.11.
[37] J.D. Herbsleb, A. Mockus, An empirical study of speed and communication in globally distributed software development, IEEE Trans. Softw. Eng. 29 (6) (2003) 481–494.
[38] J.D. Herbsleb, D. Moitra, Global software development, Software, IEEE 18 (2) (2001) 16–20.
[39] C.W. Ho, S. Raha, E. Gehringer, L. Williams, Sangam: a distributed pair programming plug-in for Eclipse, in: Proceedings of the 2004 OOPSLA Workshop on Eclipse Technology eXchange, ACM, 2004 October, pp. 73–77.
[40] R. Hoda, J. Noble, S. Marshall, Supporting self-organizing agile teams, Agile Process. Softw. Eng. Extreme Programming, vol. 77, Springer, Berlin Heidelberg, 2011, pp. 73–87.
[41] H. Holmström, B. Fitzgerald, P.J. Ågerfalk, E.Ó. Conchúir, Agile practices reduce distance in global software development, Inf. Syst. Manage. 23 (3) (2006) 7–18.
[42] E. Hossain, M.A. Babar, H.Y. Paik, J. Verner, Risk identification and mitigation processes for using scrum in global software development: a conceptual framework, in: Software Engineering Conference, 2009. APSEC'09. Asia-Pacific, IEEE, 2009 December, pp. 457–464.
[43] S. Jalali, C. Wohlin, Agile practices in global software engineering – a systematic map, in: 2010 5th IEEE International Conference on Global Software Engineering (ICGSE), IEEE, August 2010, pp. 45–54.
[44] M. Jiménez, M. Piattini, A. Vizcaíno, Challenges and improvements in distributed software development: a systematic review, Adv. Softw. Eng. 2009 (2009) 3.
[45] K.H. Judy, I. Krumins-Beens, Great scrums need great Product owners: unbounded collaboration and collective product ownership, in: Proceedings of the 41st Annual Hawaii International Conference on System Sciences, IEEE, 2008, January, pp. 462–462.
[46] T. Kahkonen, P. Abrahamsson, Digging into the fundamentals of extreme programming building the theoretical base for agile methods, in: Proceedings. 29th Euromicro Conference, 2003, IEEE, September 2003, pp. 273–280.
[47] M. Kajko-Mattsson, G, Azizyan, M.K. Magarian, Classes of distributed Agile development problems, in: Agile Conference (AGILE), IEEE, August 2010, pp. 51–58.
[48] M. Kajko-Mattsson, G.A. Lewis, D. Siracusa, T. Nelson, N. Chapin, M. Heydt, J. Nocks, H. Snee, Long-term life cycle impact of agile methodologies, in: 22nd IEEE International Conference on Software Maintenance, 2006. ICSM'06, IEEE, September 2006, pp. 422–425.
[49] P.G. Keen, Information systems and organizational change, Commun. ACM 24 (1) (1981) 24–33.
[50] S. Krishna, S. Sahay, G. Walsham, Managing cross-cultural issues in global software outsourcing, Commun. ACM 47 (4) (2004) 62–66.
[51] M. Kunz, R.R. Dumke, N. Zenker, Software metrics for agile software development. In 19th Australian Conference on Software Engineering, 2008. ASWEC 2008, IEEE, March 2008, pp. 673–678.
[52] C. Larman, B. Vodde, Scaling Lean and Agile Development: Thinking and Organizational Tools for Large-scale Scrum, Pearson Education, 2011.
[53] L. Layman, L. Williams, D. Damian, H. Bures, Essential communication practices for Extreme Programming in a global software development team, Inf. Softw. Technol. 48 (9) (2006) 781–794.
[54] H.J. Leavitt, Applied organization change in industry: structural, technical and human approaches, New Persp. Org. Res. 55 (1964) 71.
[55] D. Leffingwell, Scaling Software Agility: Best Practices for Large Enterprise, Pearson Education, 2007.
[56] T.O. Lehtinen, R. Virtanen, J.O. Viljanen, M.V. Mäntylä, C. Lassenius, A tool supporting root cause analysis for synchronous retrospectives in distributed software teams, Inf. Softw. Technol. 56 (4) (2014) 408–437.
[57] T. Little, Context-adaptive agility: managing complexity and uncertainty, Software, IEEE 22 (3) (2005) 28–35.
[58] K. Lyytinen, L. Mathiassen, J. Ropponen, Attention shaping and software risk—a categorical analysis of four classical risk management approaches, Inf. Syst. Res. 9 (3) (1998) 233–255.
[59] C. Marshall, G.B. Rossman, Designing Qualitative Research, Sage publication Inc., Thousand Oaks, 2011.
[60] J.A. Maxwell, Understanding and validity in qualitative research, Harvard Educ. Rev. 62 (3) (1992) 279–301.
[61] N.B. Moe, A. Aurum, T. Dybå, Challenges of shared decision-making: a multiple case study of agile software development, Inf. Softw. Technol. 54 (8) (2012) 853–865, http://dx.doi.org/10.1016/j.infsof.2011.11.006.
[62] S. Moore, L. Barnett, Offshore Outsourcing and Agile Development, Forrester Research inc., 2004.
[63] V. Mudumba, O.K. Lee, A new perspective on GDSD risk management: agile risk management, in: 2010 5th IEEE International Conference on Global Software Engineering (ICGSE), IEEE, August 2010, pp. 219–227.
[64] S. Nerur, R. Mahapatra, G. Mangalaraj, Challenges of migrating to agile methodologies, Commun. ACM 48 (5) (2005) 72–78.

[65] S. Nokes, The Definitive Guide to Project Management, second ed., Financial Times/Prentice Hall, London, 2008. ISBN 978 0 273 71097 4.
[66] J. Nyfjord, M.M. Kajko-Mattsson, Outlining a model integrating risk management and Agile software development, in: Proceedings of the 34th EUROMICRO Conference on Software Engineering and Advance Applications IEEE Computer Society, 2008, pp. 476–483.
[67] E. O' Conchuir, H. Holmstrom, P.J. Agerfalk, B. Fitzgerald, Exploring the assumed benefits of global software development, in: ICGSE'06. International Conference on Global Software Engineering, IEEE, October 2006, pp. 159–168.
[68] M. Paasivaara, S. Durasiewicz, C. Lassenius, Using scrum in distributed agile development: a multiple case study, in: ICGSE 2009. Fourth IEEE International Conference on Global Software Engineering, 2009, IEEE, July 2009, pp. 195–204.
[69] M. Paasivaara, S. Durasiewicz, C. Lassenius, Distributed agile development: using scrum in a large project, in: Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on (pp. 87–95). IEEE.
[70] Paetsch, F., Eberlein, A., & Maurer, F. (2003, June). Requirements engineering and agile software development, in: 2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, August 2008, IEEE Computer Society, pp. 308–308.
[71] C.R. Pandian, Applied Software Risk Management: A Guide for Software Project Managers, CRC Press, 2006.
[72] J. Patton, Unfixing the fixed scope project: using agile methodologies to create flexibility in project scope, in: Proceedings of the Agile Development Conference, 2003. ADC 2003, IEEE, 2003, June, pp. 146–151.
[73] M.Q. Patton, Qualitative Research and Evaluation Methods, third ed., A: Sage, Thousand Oaks, 2002.
[74] J.S. Persson, L. Mathiassen, J. Boeg, T.S. Madsen, F. Steinson, Managing risks in distributed software projects: an integrative framework, IEEE Trans. Eng. Manage. 56 (3) (2009) 508–532.
[75] R. Phalnikar, V.S. Deshpande, S.D. Joshi, Applying agile principles for distributed software development, in: ICACC'09. International Conference on Advanced Computer Control, IEEE, 2009 January, pp. 535–539.
[76] A. PMI, Guide to the Project Management Body of Knowledge (PMBOK), 2004 Edition. Project Management Institute.
[77] R.S. Pressman, Software Engineering. A Practitioner's Approach Sixth Edition, McGrawHill, 2005.
[78] R. Prikladnicki, D. Damian, J.L.N. Audy, Patterns of evolution in the practice of distributed software development in wholly owned subsidiaries: a preliminary capability model, in: IEEE International Conference on Global Software Engineering, 2008. ICGSE 2008, IEEE, pp. 99–108, pp. 17–20, doi: 10.1109/ICGSE.2008.36.
[79] R. Prikladnicki, J.L.N. Audy, R. Evaristo, A reference model for global software development: findings from a case study, in: ICGSE'06. International Conference on Global Software Engineering, 2006, IEEE, October 2006, pp. 18–28.
[80] B. Ramesh, L. Cao, K. Mohan, P. Xu, Can distributed software development be agile?, Commun ACM 49 (10) (2006) 41–46.
[81] J.M. Robarts, Practical considerations for distributed projects, in: Proc. AGILE 2008 Conference, IEEE Computer Society, Toronto, Canada, 4–8, 2008, pp. 327–332.
[82] J. Rothman, S. Hastie, Lessons learned from leading. Workshops about geographically distributed agile teams, IEEE Softw. 30 (2) (2013) 7–10.
[83] S. Rowlinson, Y.K.F. Cheung, Stakeholder management through empowerment: modelling project success, Constr. Manage. Econ. 26 (6) (2008) 611–623.
[84] S. Sakthivel, Virtual workgroups in offshore systems development, Inf. Softw. Technol. 47 (5) (2005) 305–318.
[85] S. Sarker, S. Sahay, Implications of space and time for distributed work: an interpretive study of US–Norwegian systems development teams, Eur. J. Inf. Syst. 13 (1) (2004) 3–20.
[86] K. Saumure, L.M. Given, Data saturation, The Sage Encyclop. Qual. Methods 1 (2008) 195–196.
[87] J. Schiel, The ScrumMaster Study Guide, CRC Press, Taylor & Francis Group, LLC, 2012.
[88] R. Schmidt, K. Lyytinen, M. Keil, P. Cule, Identifying software project risks: an international Delphi study, J. Manage. Inf. Syst. 17 (4) (2001) 5–36.
[89] K. Schwaber, M. Beedle, Agile Software Development with Scrum, Prentice Hall, Upper Saddle River, NJ, 2002.
[90] W.R. Scott, Organizations: Rational, Natural and Open Systems, Prentice Hall, Englewood Cliffs, 1992.
[91] C.B. Seaman, Qualitative methods in empirical studies of software engineering, IEEE Trans. Softw. Eng. 25 (4) (1999) 557–572.
[92] H. Sedehi, G. Martano, Metrics to Evaluate & Monitor Agile Based Software Development Projects – A Fuzzy Logic Approach, in: 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement (IWSM-MENSURA), October 2012, pp. 99, 105, 17–19.
[93] B. Sengupta, S. Chandra, V. Sinha, A research agenda for distributed software development, in: Proceedings of the 28th International Conference on Software engineering, ACM, May 2006, pp. 731–740.
[94] J. Shore, S. Warden, The Art of Agile Development, first ed., O'Reilly Media, Inc., 2007.
[95] D. Šmite, C. Wohlin, A whisper of evidence in global software engineering, IEEE Softw. 28 (4) (2011) 15–18. doi: http://dx.doi.org/10.1109/MS.2011.70.
[96] G. Stoneburner, A. Goguen, A. Feringa, Risk management guide for information technology systems, Natl. Inst. Stand. Technol., Spec. Publ. 800 (30) (2002) 800–830.
[97] K. Sureshchandra, J. Shrinivasavadhani, Adopting agile in distributed development, in: IEEE International Conference on Global Software Engineering, 2008. ICGSE 2008, August 2008, pp. 217–221. doi: http://dx.doi.org/10.1109/ICGSE.2008.25.
[98] J. Sutherland, G. Schoonheim, E. Rustenburg, M. Rijk, Fully distributed scrum: the secret sauce for hyperproductive offshored development teams, Agile, 2008 Conference, 4–8 August 2008, pp. 339,344. doi: http://dx.doi.org/10.1109/Agile.2008.92.
[99] E. Therrien, Overcoming the challenges of building a distributed Agile organization, in: Proc. AGILE 2008 Conference, IEEE Computer Society, Toronto, Canada, 2008, pp. 368–372.
[100] J. Vanthienen, S. Poelmans, A general framework for positioning, evaluating and selecting the new generation of development tools. EUROMICRO 96. Beyond 2000: Hardware and Software Design Strategies, in: Proceedings of the 22nd EUROMICRO Conference, September 1996, pp. 233,240. doi: http://dx.doi.org/10.1109/EURMIC.1996.546387.
[101] VersionOne, 7th Annual State of Agile development Survey. <http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf> (accessed September 2013).
[102] D. West, Case Study: Thoughtworks Makes Distributed Agile Work, 2011. <http://www.thoughtworks.com/sites/www.thoughtworks.com/files/files/Forrester-Case-Study-ThoughtWorks-Makes-Distributed-Agile-Work.pdf>.
[103] E. Woodward, S. Surdek, M. Ganis, A Practical Guide to Distributed Scrum, Pearson Education, 2010.
[104] R.K. Yin, Case Study Research: Design and Methods, third ed., Sage Publications, Thousand Oaks, CA, 2003.
[105] G. Zikmund William, Business Research Methods, Thomson Learning, 2006.