Chapter 13
Using Distributed Agile Patterns for Supporting the Requirements Engineering Process

Maryam Kausar and Adil Al-Yasiri

Abstract This chapter discusses the challenges practitioners face while choosing to develop their projects at offshore locations. As offshore development introduces new challenges in the software development process such as trust, socio-cultural, communication and coordination and knowledge transfer issues, it has been observed that these challenges affect how requirements are defined and managed while using agile practices in offshore software development. Using the notions of Distributed Agile Patterns we discuss how they can facilitate the requirements engineering process in offshore software development. We present a catalogue of the complete set of patterns, but only gave details of selective patterns from the catalogue that are related to the requirements engineering process. The whole catalogue is available online for anyone interested in it. At the end we developed a process flow showing the distributed agile patterns mapped onto the traditional requirements engineering process to show how these patterns address and improve the requirements engineering process for agile offshore projects.

Keywords Distributed agile patterns · Global software engineering · Requirements engineering

13.1 Introduction

The process of requirements elicitation is one of the most challenging tasks in software development. In traditional software development methodologies, the client would predefine all their requirements to the development team before the start of subsequent phases. The team would then analyse the requirements and finalise a software requirements specification (SRS) document. Once the client has approved the document, they would start the development phase. This process of requirements elicitation has problems such as; a long time is spent in preparing this

M. Kausar · A. Al-Yasiri (✉)
School of CSE, University of Salford, M5 4WT Salford, Greater Manchester, UK
e-mail: a.al-yasiri@salford.ac.uk

M. Ramachandran and Z. Mahmood (eds.), *Requirements Engineering for Service and Cloud Computing*, DOI 10.1007/978-3-319-51310-2_13

documentation, which causes issues in dealing with future requirements change requests from the client once the actual development phase starts. Over decades of software development we have learnt that requirements change is inevitable during the development stage, because neither the client nor the developers are 100% sure of all the requirements of the system at the start of the project.

In agile software development, this problem is solved with the use of story cards, which is a lighter process for the definition of very high-level requirements and is an artefact of methods such as SCRUM and XP. They contain just enough information for the developer to be able to estimate how much effort and time will be required to develop them and can handle change requests with little effort. There is also an agile requirements change management process to control changing requirements throughout the software development lifecycle. Based on this process, new requirements can be added and reprioritized based on the client's request [1]. Figure 13.1 illustrates the agile requirements change management process:

However, in distributed agile software development, as the team is distributed over different time zones, the process of gathering and documenting requirements becomes more complex. As any change in the requirements need to be communicated over different locations and due to cultural and language differences, requirements can be misunderstood.

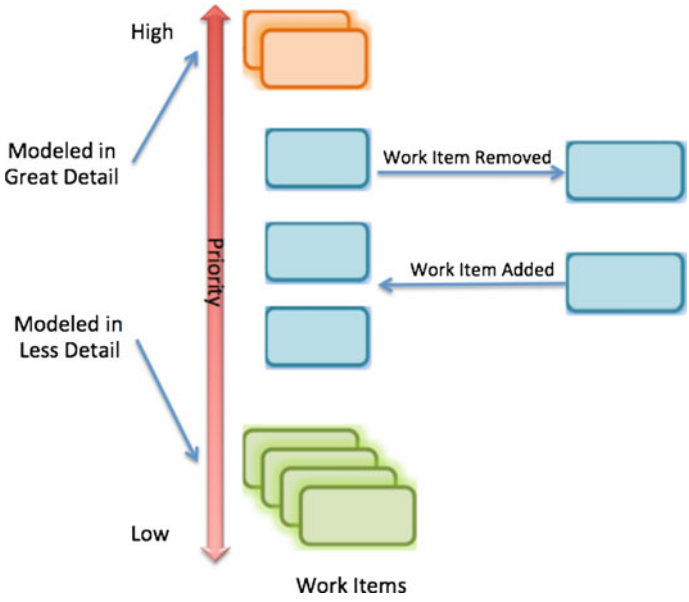

Fig. 13.1 Agile requirements change management process

In this chapter we will discuss how agile methods are used in offshore software development, what are the challenges facing the requirements elicitation process in agile offshore software development and how we can use distributed agile patterns to overcome these challenges.

13.2 Agile Offshore Software Development

Since the creation of the agile manifesto it has brought unprecedented changes to how software is being developed [2]. The manifesto focuses on four points, which are:

i. Individuals and interaction over processes and tools
ii. Working software over comprehensive documentation
iii. Customer collaboration over contract negotiation and
iv. Responding to change over following a plan.

Nowadays many companies are using agile for developing their offshore projects [3]. However, the use of agile methods on offshore projects is not a straightforward process. Taylor et al. [4] claim that projects that use agile for offshore development go through many problems because of the differences in the development practices and the complex development environments. Table 13.1 shows a characteristic comparison of agile and offshore development, done by Šmite et al. [5], showing the differences in the development styles.

Table 13.1 shows that the application of agile methods is not a straightforward process in offshore development. Based on extensive literature review on offshore software development four key challenges have been identified that affect the adoption of agile practices. Table 13.2 shows the results of how these challenges affect agile practices.

Table 13.2 summarises the agile practices that cannot be used as they are in offshore software development and that the requirements engineering process is affected in offshore development. In the next section we have identified how these challenges affect the requirements engineering process in agile offshore development.

Table 13.1 Comparison of agile development versus offshore development [5]

Characteristics	Agile development	Offshore development
Communication	Informal Face-to-face Synchronous Many-to-many	Formal Computer-mediated Often synchronous Tunnelled
Coordination	Change-driven Mutual adjustment, Self management	Plan-driven Standardisation
Control	Lightweight Cross-functional team	Command-and-control Clear separation of roles

Table 13.2 Agile practices affected by offshore challenges

No.	Offshore challenge	Agile practice	Affect of challenge on agile practice
1.	Trust	Collective ownership	Dispute over code ownership among the onshore and offshore team members
		Sustainable pace	Difficulties in maintaining a sustainable pace of project development
2.	Socio-cultural	Iterative and incremental development	Delays in frequent delivery of code
		Self-organising teams	Problems in understanding each other's cultural and social values can cause a barrier in the formation of self-organising teams
3.	Communication and coordination	Sprint planning	As the team is distributed, due to lack of sufficient communication, it can cause problems is designing a correct sprint
		Continuous integration	Multiple versions of code developed at different location can cause any build to break due to errors being integrated in the code
4.	Knowledge transfer	Product backlog	Any change in the product backlog not documented correctly can cause a project to fail
		Sprint review	As the sprint is being developed at multiple locations it causes problems in determining the progress of the work done

13.3 Challenges in Requirements Engineering Process in Agile Offshore Development

This section discusses the challenges and effect of agile offshore development on the process of gathering and documenting requirements. Table 13.3 shows the effect of agile offshore development challenges on the requirements elicitation process. As the table demonstrates the four challenges identified in our study have direct impact on the gathering, analysis and change management of requirements.

13.4 Distributed Agile Patterns

Based on the previous section we can see that applying agile on offshore project is not a straightforward process. We studied over 200 cases from the literature and interviewed practicing professionals involved in distributed teams, which resulted in observing a number of solutions addressing common agile issues in offshore software development settings, which we presented as Distributed Agile Patterns.

Table 13.3 Requirements engineering process affected by challenges of agile offshore development

No.	Challenges in agile offshore development	Affect on requirements engineering process in agile offshore development
1.	Trust	Establishing correct user story estimates as onshore team does not have clear understanding of the skills of the offshore team
		Getting the main objective and core functional requirements of the project clear to all the team members located at different sites
2.	Socio-cultural	Differences in cultural values and language can cause misunderstanding of requirements
		As at the beginning of the project the client is not sure of all the requirements, which can result in vague requirements, which need to be later clarified to the teams who are not on-site
3.	Communication and coordination	Changes in the requirements, needs to be communicated to all the distributed teams. Any mistake in recording the change can cause problems in the development of the system
4.	Knowledge transfer	As the project is being developed at different locations, there can be inconsistencies in the work done and documented user stories
		Maintaining bidirectional traceability of requirements across different sites is difficult as each site is working on different use stories
		Managing requirements, on-site customer and daily meetings become difficult with distributed teams

The term "pattern" is commonly referred to as a reusable solution for a recurring problem within a given context [6]. Based on this definition, we defined **Distributed Agile Patterns** as, adaptation of an agile practice that is being repeatedly applied in order to solve a recurring challenge in a distributed project scenario.

Generally a pattern has four essential elements [6]:

- The **pattern name**: to give a high level of abstract to the pattern. It gives us the idea of what problem the pattern is providing a solution for in a word or two. Giving a name to a pattern makes it easy for us to talk about it with people and for documentation.
- The **problem**: helps in describing when the pattern can be applied. It provides details of the problem and its context. It may include lists of conditions or scenarios, which must be met in order to apply the pattern.
- The **solution**: describes the elements that make up the agile patterns, their relationships and responsibilities. The solution does not describe a particular concrete agile practice or implementation, because a pattern is like a template that can be applied in many different scenarios. A pattern does provide an

abstract description of an agile practice problem and how a general arrangement of elements/practices can solve it.
- The **consequence**: describes the outcome of applying the pattern. They are critical for evaluating a pattern and for understanding the benefit of applying a pattern to see if it helped in solving the problem and if yes, up to what extent. For software the consequence often refer to space and time trade-offs. In distributed agile patterns, the consequence includes flexibility, extensibility and team coordination and collaboration.

The Distributed Agile Pattern's catalogue is developed based on literature and interviews and adopted Gamma's pattern template in order to preserve familiarity, as they are perceived as the first pattern catalogue documented by the software community. A customised template was then developed to capture the specific findings related to distribute agile practices. The distributed agile patterns template contains the following sections:

- **Pattern Name**: As patterns represent generic knowledge it is vital to give a good name that would make it recognisable and reusable. A good name also helps in facilitating communication among practitioners about the pattern.
- **Intent**: A short statement that highlights the issues and problems that are required to be solved by applying the pattern.
- **Also known As**: The pattern's other well-known names, if any are mentioned in this section.
- **Category**: Based on the similarities of the patterns we grouped them into different categories to be able to provide an abstract view of all the patterns.
- **Motivation**: It consists of the description of the problem and why the pattern should be used in order to avoid the problem from recurring. It provides scenarios that help understand the abstract description of the pattern.
- **Applicability**: Under what conditions the pattern can be applied.
- **Participants**: The participants are those people that are required in applying the pattern.
- **Collaboration**: How participants will coordinate with each other in order to fulfil their responsibilities that are required to complete the projects.
- **Consequences**: Discuss the trade-offs of applying the patterns such as advantages and difficulties faced when applying it.
- **Known uses**: Examples of real scenarios found that follow the pattern in order to provide clarity of how the pattern can be used.
- **Related Pattern**: List of similar patterns in order to identify which patterns can be used together to improve a particular situation.

Following is the summary of list of the identified distributed agile patterns [7]:

1. **Distributed Scrum of Scrum Pattern**: To apply scrum, sub-teams are formed based on location. Each team has its own scrum. Scrum of scrum meetings are arranged to discuss the progress of the project, which is attended by key people.

2. **Local Standup Meetings Pattern**: To discuss daily updates on work done, each local team will conduct their own stand-up meetings.
3. **Follow the Sun Pattern**: Onshore and offshore teams will work 9 a.m–5 p.m according to their own time zones.
4. **Onshore Review Meeting**: The onshore team will present the demo as they are located where the client is.
5. **Collective Project Planning**: Both the onshore team and the offshore team will collectively work in the project planning phase.
6. **Project Charter Pattern**: Before starting the project planning activity, agile teams use project charter in order to have a central document between the onshore and offshore team that defines the project.
7. **Collaborative Planning Poker**: Only Key people will hold this activity from onshore and offshore team.
8. **Global Scrum Board**: There will be an online-shared Scrum board, which, both onshore and offshore team can use to view product backlog, storyboard, task board, burn down charts and other agile artefacts using online tools such as wikis.
9. **Local Sprint Planning**: Each team will have their own sprint planning meetings.
10. **Local Pair Programming**: Make pair programming teams from the same location.
11. **Central Code Repository**: The whole team will maintain a central code repository so that both team can see each other's code and see the progress of the work done.
12. **Asynchronous Retrospective Meetings**: Teams conduct separate retrospective meetings based on location and share the key information via email. The Scrum Masters discuss possible improvements with the team based on the feedback from the client.
13. **Asynchronous Information Transfer**: Due to the time difference between the onshore and offshore team use online tools to exchange information with each other. Each team should response to queries within 12 h.
14. **Synchronous Communication**: In order to discuss issues the teams used synchronous tools for voice, video conferencing, document sharing, application sharing, etc.
15. **Visit onshore–offshore Teams**: Both onshore and offshore teams should quarterly/annually visit each other in order to build trust, exchange cultural values and improve team coordination.

Fifteen distributed agile patterns were identified and organised into four categories based on the type of problem they solve, which are **management**, **communication**, **collaboration** and **verification patterns**. Table 13.4 shows the 15 distributed agile patterns according to their categories.

Table 13.4 Categories of distributed agile pattern

	Category			
	Management patterns	Communication patterns	Collaboration patterns	Verification patterns
Pattern names	Distributed scrum of scrum	Global scrum board	Collaborative planning poker	Project charter
	Local stand-up meeting	Central code repository	Follow-the-sun	Onshore review meeting
	Local sprint planning	Asynchronous information transfer	Collective project planning	
	Local pair programming	Synchronous communication	Visit onshore–offshore	
	Asynchronous retrospective			

- **Management patterns** help in managing the onshore and offshore team members and their activities to effectively apply agile in a distributed environment.
- **Communication patterns** focus on providing solutions to how distributed team members can maintain an effective communication channel in an agile setting using different online tools which provide both synchronous and asynchronous method for communication.
- **Collaboration patterns** provide solutions regarding which activities the onshore and offshore team members should conduct together to improve team coordination and project progress.

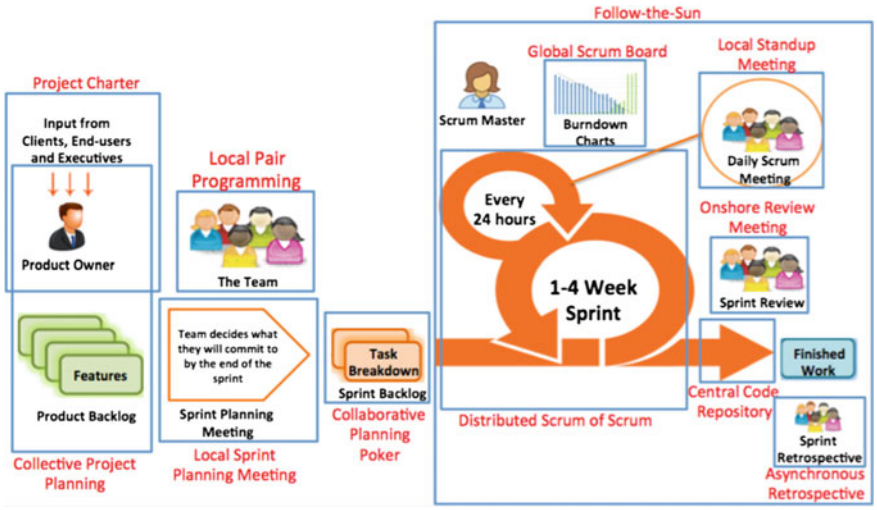

Fig. 13.2 Distributed agile patterns application of software development lifecycle

- **Verification patterns** focuses on how efficiently the clients can get a distributed project developed according to their requirements and monitor the progress of what has been developed.

The identified distributed agile patterns are spread across the Scrum software development lifecycle as shown in Fig. 13.2. In this chapter we have only presented the patterns that map to the requirements engineering process, however, the full catalogue is available online [8].

13.5 Distributed Agile Patterns Used for Requirements Engineering Process

In this section we present the selective Distributed Agile Patterns that are used in the Requirements engineering process, which are further explained in the following section showing how they are used to gather and document requirements.

13.5.1 Project Charter Pattern

In project management, a project charter is a statement that defines the scope, objectives and participants of a project. It is used to explain the roles and responsibilities, outline of the project objectives and identify main stakeholders. It has been observed that while starting a distributed project using agile many organisation use project charter to clarify the goals and objectives of the project to both onshore and offshore team [8] (Table 13.5).

13.5.2 Collective Project Planning Pattern

Agile focuses on individuals and interactions over processes and tools. While planning for the project the whole team is present. Unlike the traditional development where a project manager hands a project plan to the team, in agile the whole team takes part in the planning activity in order to determine when and how the project will be developed. It has been observed that even if the project is of a distributed nature it is better to co-locate the onshore and offshore teams for the project planning activity. The motivation of this pattern is to address the trust, socio-cultural, communication and coordination and knowledge transfer challenges. For example, consider a team that is divided into sub-teams that are located on

Table 13.5 Project charter pattern

No.	Pattern element	Detail
1.	Pattern name	Project charter pattern
2.	Intent	Before starting the project planning activity, agile teams use project charter in order to have a central document between the onshore and offshore team that defines the project
3.	Also known as	Project definition or project statement
4.	Category	Verification category, as this pattern helps the onshore and offshore team to have a central document clarifying the project goals and objectives, which is written by the product owner/client
5.	Motivation	The motivation of this pattern is to address the trust, communication and coordination and knowledge transfer challenges. For example when a project is distributed to a team that is divided over different time zones, a central document is written known as the project charter, which clarifies the onshore and offshore goals and objectives of the project. It also identifies the roles and responsibilities of the onshore and offshore team. The purpose of this activity is to have a document that helps the team in the project planning task
6.	Applicability	Use project charter pattern when: • Team is distributed over different time zones
7.	Participants	• Distributed onshore and offshore agile team • Client
8.	Collaboration	The client gives the project charter to the onshore team and offshore team to clarify the goals of the project
9.	Consequences	The project charter pattern has the following benefits: 1. It allows the onshore and offshore teams to understand the project. This helps overcome communication and coordination, and knowledge transfer challenges 2. Since it is a single document stating the goals and objectives of the project it helps establish trust between the onshore and offshore team members 3. It is intended to clearly set the stage for the project by aligning the team and settings goals and expectations
10.	Known uses	IONA technologies used project charter for their distributed projects in order to have a central document that clarifies the goals of the project to both onshore and offshore team members [17]. Similarly in a case study conducted by Brown [18] on Agile-at-Scale Delivery it was observed that organisations use project charter
11.	Related patterns	Project charter pattern is often used with visit onshore–offshore team pattern

different time zones and both the teams at the beginning of the project come to one location to do the project planning activity; this helps the team members to understand each other and establish working standards for the project. The detail table for this pattern can be found in Appendix 1.

13.5.3 Local Sprint Planning Meeting Pattern

In agile, a scrum consists of many sprints. The duration of a sprint varies from 1 to 4 weeks depending on the size of the project. At the start of every sprint the team has a sprint planning meeting in which the team defines the goal of the sprint and prepare the sprint backlog. When the team is divided and is working on different modules of the project it has been observed that the onshore team members and offshore team members conduct their own separate sprint planning meetings. The motivation of this pattern is to address the communication and coordination and knowledge transfer challenges. For example when a project is distributed to a team that is divided over different time zones, it is better that each location has their own sprint planning as it helps each team to decided their own tasks without having to wait for the other teams to be present. The detail table for this pattern can be found in Appendix 2.

13.5.4 Collaborative Planning Poker Pattern

An agile team plays a planning poker to put point estimation on each story card. The product owner also takes part in this activity. He/She tells the team the intent and value of a story card based upon which development team assigns estimation on the card. Based on the points assigned, the team members who assigned the lowest and highest estimation will justify their reasons. The team will have a brief discussion on each story and assign a fresh estimation upon which the whole team agrees on.

It has been observed that even when the team is distributed the planning poker activity is conducted when both teams are co-located for the project planning activity. The motivation of this pattern is to address the trust, socio-cultural, communication and coordination and knowledge transfer challenges. For example when a project is distributed over different locations, team members located at different locations, do not know each other's skill set so with the help of collaborative planning poker, each member can assign estimation points according to their skills. The detail table for this pattern can be found in Appendix 3.

13.5.5 Global Scrum Board Pattern

Agile has many artefacts such as product backlog, sprint backlog, storyboard, task board, team velocity and burndown charts which help the team in managing the project. It has been observed that when the team is divided to different locations they maintain an online record of all these artefacts so that they can share them with each other using online tools such as Wiki's, Rally and Jira [9–11]. The motivation

of this pattern is to address the trust, socio-cultural, communication and coordination and knowledge transfer challenges. As the team is distributed over different time zones, any change in the requirements and sprint can be viewed in real time with the help of the global scrum board. The detail table for this pattern can be found in Appendix 4.

13.5.6 Central Code Repository Pattern

In agile, when a team is using Scrum and XP, the team members are divided in pairs of two and are working on different tasks during a sprint. When a task is completed the team members commit their code to a share repository for continuous integration of the code. It is observed that even when the team members are geographically apart they still use a share code repository where they commit their code so that all the team members can see the code as well as determine the progress of the project. The motivation of this pattern is to address the communication and coordination and knowledge transfer challenges. As the team members are distributed over different locations, with the help of the central code repository, all the team members can see the progress of the project. The detail table for this pattern can be found in Appendix 5.

13.5.7 Asynchronous Information Transfer Pattern

Agile emphases on close face-to-face communication between the team members rather than detailed documentation. When a team is distributed on different time zones it has been observed that the teams adopted asynchronous tools for sharing information with each other such as emails, Wikis, SharePoint. The motivation of this pattern is to address the communication and coordination and knowledge transfer challenges. Due to different time zones, the team members can use asynchronous communication methods to share information with each other. The detail table for this pattern can be found in Appendix 6.

13.5.8 Synchronous Communication Pattern

Agile emphases on close face-to-face communication between the team members rather than detailed documentation. When a team is distributed on different time zones it has been observed that the teams adopted asynchronous tools for sharing information with each other such as emails, Wikis, SharePoint. The motivation of this pattern is to address the trust, socio-cultural, communication and coordination and knowledge transfer challenges. For example, in case of any confusion, team

members can communicate with each other using real time synchronous tools for communication. The detail table for this pattern can be found in Appendix 7.

13.5.9 Visit Onshore–Offshore Team Pattern

As agile emphases on close face-to-face communication between the team members it has been observed that when the team is divided on different time zones, the team members travel quarterly or annually to visit each other. This activity helps build trust among the team members and helps them understand each other's cultural differences [12–15]. The motivation of this pattern is to address the trust, socio-cultural, communication and coordination and knowledge transfer challenges. For example, in order to establish trust and understand each other's culture, team members should quarterly/annually travel and do team activities. The detail table for this pattern can be found in Appendix 8.

13.6 Use of Distributed Agile Patterns in Requirements Engineering Process in Agile Offshore Development

In order to verify and validate the identified distributed agile patterns, we conducted a reflection workshop based on Norm Kerth [16], '**The keep/try reflection workshop**'. Based on the workshop we designed Table 13.6, which shows how Distributed Agile Patterns address requirements engineering challenges in offshore development, which were mentioned in Table 13.3, to the relevant distributed agile pattern addressing them.

13.7 Mapping Distributed Agile Patterns on the Requirements Engineering Lifecycle

In this section we map the distributed agile patterns onto the traditional requirements engineering lifecycle, to show how these patterns facilitate the requirements engineering process. As shown in Fig. 13.3, the four key features of this process are as following:

- **Feasibility Study**: In this process, we decide whether or not the proposed system is worthwhile. By writing down the Project Charter document in the beginning of the project, we can achieve this. As stated in the Project Charter we define the aim, objectives and core functional requirements of the proposed system.

Table 13.6 Using distributed agile patterns to address requirements engineering challenges in agile offshore development

No.	Requirements challenge in agile offshore development	Distributed agile pattern	Solution
1.	User story estimation	Collective planning poker	By doing the planning poker activity together all team members will get a better understanding of each other's skills
2.	Objective and core functional requirements	Project charter	Having a project charter document written at the start of the project, will give all the team members a clear understanding of the project's objective and core functional requirements
3.	Misunderstanding of requirements	Collective project planning	Since the whole team will be part of the planning activity, the chances of misunderstanding a requirement will be reduced
		Asynchronous information transfer	In case of any misunderstanding, the team members can communicate with each other using asynchronous tools
		Synchronous communication	For a real time response to any misunderstanding, the team members can use synchronous methods for communication
4.	Vague requirements	Visit onshore–offshore	To clarify vague requirements, onshore team members should visit offshore team members and vice versa to discuss the requirements in order to avoid defects
5.	Changes in the requirements	Global scrum board	Any change in the requirements will be updated on the global scrum board, which is accessible by all team members in real time.
6.	Inconsistencies	Central code repository	To avoid any inconsistency between the work done and the user stories, all the team members use a central code repository, which is accessible by the whole team. All the code is committed in that repository, enabling the whole team to see what work is done and what is remaining
7.	Bidirectional traceability of requirements	Central code repository	With the help of a central code repository, we can map each requirement with its code, allowing traceability of all the requirements being developed at different locations
		Global scrum board	Status of all requirements being developed can be recorded using a global scrum board, which helps in maintaining traceability of requirements
8.	Managing requirements	Local sprint planning	Each site can manage their requirements and daily meetings using local sprint planning

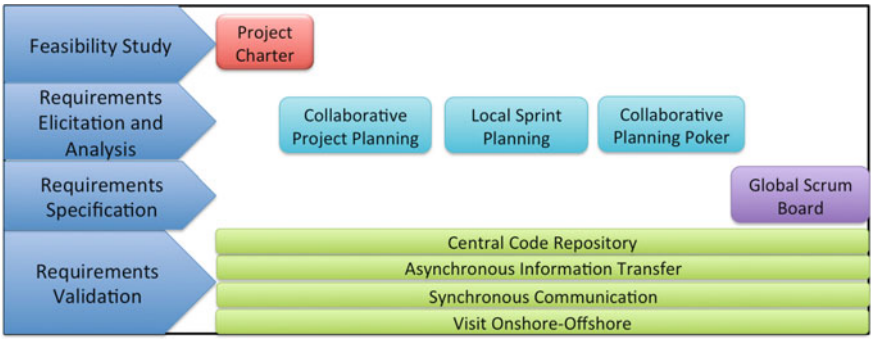

Fig. 13.3 Mapping distributed agile patterns on traditional requirements engineering process

- **Requirements Elicitation and Analysis**: The purpose of this process is to identify the application domain; the services it will provide and what are the limitations. Three distributed agile patterns are applied in these phases. Collaborative project planning helps the team to find out what the requirements of the application to be developed. Local sprint planning helps team members in their respective locations to discuss in detail the user stories allocated to them and determine the constrains associated with them. Collaborative Planning Poker, allows the team members to discuss the services that the system will provide and estimate how much effort is required to develop them.
- **Requirements Specification**: Once the requirements have been identified, we document them in the SRS document. As in agile software development, requirements are documented using user stories, in distributed agile software development; we use a Global Scrum Board, where all the story cards are placed. Hence any change or modification in the requirements will be updated on the Global Scrum Board, which is accessible by all the team in real time.
- **Requirements Validation**: In this process, the requirements are validated by reviewing them to make sure that the identified requirements are in accordance with what the client wants. Four distributed agile patterns make sure that the correct requirements are identified. Central Code Repository helps the client verify that the code being developed is according to requirements. Asynchronous Information Transfer provides a platform to the team members and the clients to communicate and coordinate the progress of the system being developed and in case of any confusion, the team can either use asynchronous or synchronous tools for communication, according to the communication standards defined using

Synchronous Communication Pattern. For further clarification of requirements, distributed agile patterns suggest that both onshore and offshore team members should visit each other.

13.8 Conclusions

In this chapter we have discussed the requirements engineering process and highlighted the issues practitioners face in agile offshore development such as incorrect user story estimation, problems in identifying the core functional requirements and vague requirements. Based on our observation and literature, we found that these challenges affect the whole requirements engineering process. Using distributed agile patterns, practitioners can avoid these challenges. They can use them at the beginning of their offshore projects and make informed decisions about how to adopt agile approaches to gather and document requirements, as generalised patterns make it easier for other companies to reflect on and to apply the results to their own cases.

Appendix 1: Collective Project Planning Pattern

See Table 13.7.

Table 13.7 Detail of collective project planning pattern

No.	Pattern element	Detail
1.	Pattern name	Collective project planning pattern
2.	Intent	Both the onshore team and the offshore team will collectively work in the project planning phase. Once both teams have engaged in the project planning activity, the team will prepare the project backlog
3.	Also known as	Project planning or agile project planning
4.	Category	Coordination category, as this pattern helps the onshore and offshore team to work together and come up with a project plan
5.	Motivation	The motivation of this pattern is to address the trust, socio-cultural, communication and coordination and knowledge transfer challenges. For example consider a team that is divided into sub-teams that are located on different time zones and both the teams come to one location to do the project planning activity. In the beginning of any distributed project, the offshore team is invited to the onshore location so that they may work together and understand each other's requirements.

(continued)

Table 13.7 (continued)

No.	Pattern element	Detail
		While the teams are co-located they worked on preparing the product backlog and they spend at least one or two sprints together before the offshore team leaves and starts working on the project [9, 15]. This helps the onshore team by making the offshore team understand their working style and work standard
6.	Applicability	Use collective project planning pattern when: • Team is distributed over different time zones
7.	Participants	Distributed onshore and offshore agile team
8.	Collaboration	Onshore team and offshore team work together to make a product backlog
9.	Consequences	The collective project planning pattern has the following benefits and liabilities: 1. It allows the onshore and offshore teams to work together and understand each other. This helps build trust among the team members and overcome communication and coordination challenges 2. Onshore team works with the offshore team and makes them understand what type of work they want. This helps overcome the socio-cultural and knowledge transfer challenges 3. It adds additional cost of travel and stay of the offshore team at the onshore location
10.	Known uses	FAST, a search company with headquarters in Norway while building a search application on top of their core search platform used collective project planning to co-locate the team and make them work together in project planning activities [10]. Siemens also used collaborative planning for their distributed projects [19, 20] in which team members from multiple sites got involved in the early stages of the project in order to create an open communication channel and high level of trust among the distributed team members [21]
11.	Related patterns	Collective project planning pattern is often used with Project charter pattern as it provides a central document that consists of the goal and objectives of the project written by the client

Appendix 2: Local Sprint Planning Meeting Pattern

See Table 13.8.

Table 13.8 Detail of local sprint planning meeting pattern

No.	Pattern element	Detail
1.	Pattern name	Local sprint planning meeting pattern
2.	Intent	Each team will have their own sprint planning meetings
3.	Also known as	Sprint planning meeting or iteration meeting
4.	Category	Management category, as this pattern helps the onshore and offshore teams work on their separate modules and conduct independent scrum and sprint planning meetings

(continued)

Table 13.8 (continued)

No.	Pattern element	Detail
5.	Motivation	The motivation of this pattern is to address the communication and coordination and knowledge transfer challenges. For example when a project is distributed to a team that is divided over different time zones, and are working on different modules of the project and are conducting their own scrums. As the onshore and offshore teams conduct their separate scrums, they also conduct separate sprint planning meetings to decide what they will develop during a sprint. Both teams prepare their sprint backlogs, which are shared using online tools
6.	Applicability	Use local sprint planning meeting pattern when: • Team is distributed over different time zones and is working on different modules/subsystems of the project
7.	Participants	Distributed onshore and offshore agile team
8.	Collaboration	The onshore team and offshore team share sprint backlog with each other to show the work they will be doing over the next sprint
9.	Consequences	The local sprint planning meeting pattern has the following benefits: 1. It allows both teams to work independently without having to wait for the onshore team to be available to conduct the meeting, which helps overcome the communication and coordination challenges 2. It provides control to both onshore and offshore team to work on their scrum and conduct their own sprint planning meetings, which avoids the offshore team from having to adjust working hours based on the onshore team availability. This helps overcome the communication and coordination challenges 3. Both teams can share their sprint backlog with each other, which provides visibility of the project progress and helps overcome the knowledge sharing challenges 4. As both the teams are working independently, it can cause the teams to feel, as they are not part of one team, rather create an effect that they are two separate teams
10.	Known uses	When CheckFree decided to move their work to an Indian offshore consulting firm they used local sprint planning meetings to plan their sprint activities [9]
11.	Related patterns	Local sprint planning pattern is often used with global scrum board pattern as the meetings minutes of the planning meeting are shared with both onshore and offshore team members

Appendix 3: Collaborative Planning Poker Pattern

See Table 13.9.

Table 13.9 Detail of collaborative planning poker pattern

No.	Pattern element	Detail
1.	Pattern name	Collaborative planning poker pattern
2.	Intent	Only key people will hold this activity from onshore and offshore teams
3.	Also known as	Planning poker or scrum poker

(continued)

Table 13.9 (continued)

No.	Pattern element	Detail
4.	Category	Collaborative category, as this pattern helps the onshore and offshore teams to discuss the duration of a story card
5.	Motivation	The motivation of this pattern is to address the trust, socio-cultural, communication and coordination, and knowledge transfer challenges. For example when a project is distributed to a team that is divided over different time zones, it is important that all the team members agree on the time duration of a feature before they start developing the project. This helps estimate the duration of the project completion as well as it provides visibility of project progress. For this purpose the onshore and offshore team members play planning poker in order to collectively agree on the estimation of a story card. Once the estimation is decided they write it down and approved by the product owner/client and move on to the next story card, till all the story cards are estimated
6.	Applicability	Use planning poker pattern when: • Team is distributed over different time zones and will be working on different story cards in a sprint
7.	Participants	• Distributed onshore and offshore agile team • Product owner/Client
8.	Collaboration	The client approves the estimation made by the team members
9.	Consequences	The planning poker pattern has the following benefits: 1. It allows the onshore and offshore teams to agree on a story card estimation, which helps the team establish their team velocity. Since members from both locations are present during this activity, this helps overcome trust and socio-cultural challenges 2. It provides the product owner/client with estimation of project completion, which helps overcome the communication and coordination, and knowledge transfer challenges 3. If all team members do not agree on estimation on a story card it can lead to a long discussion, resulting the planning poker to prolong
10.	Known uses	US hardware has development centres across North America, South America and Asia. When transitioning to distributed agile environment they used planning poker for estimating their story cards [22]
11.	Related patterns	Planning poker pattern is often used with Collective Project Planning as its better to conduct this pattern when the whole team is co-located. The estimated story cards are then shared on the Global Scrum board so that whole team can view them during the project

Appendix 4: Global Scrum Board Pattern

See Table 13.10.

Table 13.10 Detail of global scrum board pattern

No.	Pattern element	Detail
1.	Pattern name	Global scrum board pattern
2.	Intent	An online-shared Scrum board, will be used by, both onshore and offshore teams to view the product backlog, storyboard, task board, burn down charts and other agile artefacts using online tools

(continued)

Table 13.10 (continued)

No.	Pattern element	Detail
3.	Also known as	Scrum Board or Agile Story Board
4.	Category	Communication category, as this pattern helps the onshore and offshore team communicate with each other using an online tool to view each other's work and understand the progress of the overall project
5.	Motivation	The motivation of this pattern is to address the trust, socio-cultural, communication and coordination, and knowledge transfer challenges. For example when a project is distributed to a team that is divided over different time zones, and are working on different modules of the project, to share their work they use an online tool to display agile artefacts. Based on the work done by both teams it is easier to see the progress of the project and it helps understand if there is a problem with a team
6.	Applicability	Use global scrum board pattern when: • Team is distributed over different time zones
7.	Participants	Distributed onshore and offshore agile team
8.	Collaboration	The onshore team and offshore team share agile artefacts with each other to show their progress
9.	Consequences	The Global Scrum board pattern has the following benefits: 1. It allows the whole team to discover the requirements, which creates visibility of the project and helps in overcoming trust challenges. The scrum board is designed keeping the socio-cultural differences in mind 2. It allows the onshore and offshore teams to understand the progress of the project, which helps overcome the communication and coordination challenges 3. It increases the visualisation of the work done by each team, which helps overcome knowledge transfer challenges
10.	Known uses	FAST, a search company with headquarters in Norway while building a search application on top of their core search platform experimented with a couple of online tools to keep both teams updated with the progress of the project. They tired XPlanner and Jira and settled for Jira, which is a web-based tool that allowed the remote team members to view the backlog and update tasks whenever they wanted [10]. Similarly in a study done by Cristal et al. [33] on an organisation that has development centres across North America, South America and Asia concluded with that the use of a global scrum board can help improve the productivity of global agile teams. Similarly companies like Valtech [11], Telco [12], BNP Paribas [23], Aginity LLC [24] and SirsiDynix [25] used online tools to share agile artefacts with their offshore team members
11.	Related patterns	Global scrum board pattern is often used with central code repository pattern as the team shares all the agile artefacts and code using an online tool

Appendix 5: Central Code Repository Pattern

See Table 13.11.

Table 13.11 Detail of central code repository pattern

No.	Pattern element	Detail
1.	Pattern name	Central code repository
2.	Intent	The whole team will maintain a central code repository so that both teams can see each other's code and view the progress of the work done
3.	Also known as	Source code repository or global build repository

(continued)

Table 13.11 (continued)

No.	Pattern element	Detail
4.	Category	Communication category, as this pattern helps the onshore and offshore team members to write code and share it on a central code repository where all team members can review the code and edit it if required
5.	Motivation	The motivation of this pattern is to address the communication and coordination, and knowledge transfer challenges. For example when a team is divided over different time zones and are working on different modules/subsystems of a project they use a central code repository to share their work with all team members. They can use online tools such as GitHub for committing their code and maintain versions of the project [26]. This helps the whole team to see the code and provides visibility of the project progress
6.	Applicability	Use central code repository when: • Team is distributed over different time zones and is working on different modules/subsystem of the project
7.	Participants	Distributed onshore and offshore agile team members
8.	Collaboration	The onshore team and offshore team members share a keyboard with a fellow team member from their respective location and once they have finished a task they commit their code to a central code repository
9.	Consequences	The central code repository pattern has the following benefits: 1. It allows the onshore and offshore team members to review each other's code, which helps overcome communication and coordination challenges 2. It helps in determining the progress of the project, which helps overcome knowledge transfer challenges 3. As all the team is committing to a central repository, if a team commits code with errors it can affect the whole build of the project
10.	Known uses	WDS global is a leading global provider of knowledge-based services to mobile operators, manufacturers and application and sales channels. In 2004 they combined their developments, which were located in UK, USA and Singapore. They shared their code on a central code repository to minimise duplications and reduce cost of maintenance [27]. Many companies use central code repository for their distributed projects such as Valtech [11], Manco [12]. Aginity LLC [24], SirsiDynix [25], Extol International [28], CE Informant [29] and ABC Bank [30]
11.	Related patterns	Central code repository pattern is often used with Global Scrum Board Pattern

Appendix 6: Asynchronous Information Transfer Pattern

See Table 13.12.

Table 13.12 Detail of asynchronous information transfer pattern

No.	Pattern element	Detail
1.	Pattern name	Asynchronous information transfer
2.	Intent	Due to the time difference between the onshore and offshore teams, they use online tools to exchange information with each other. Each team should respond to queries within 12 h
3.	Also known as	Information transfer or knowledge sharing

(continued)

Table 13.12 (continued)

No.	Pattern element	Detail
4.	Category	Communication category as this pattern helps the onshore and offshore team members to answer each other's queries within 12 h
5.	Motivation	The motivation of this pattern is to address the communication and coordination, and knowledge transfer challenges. For example when a team is divided over different time zones they may have queries about work but due to the time difference they cannot get a direct reply at that time so they use emails to communicate queries, which are then answered within 12 h max. Organisations have set standards for response time in order to avoid delays in work [31]
6.	Applicability	Use asynchronous information transfer when: • Team is distributed over different time zone
7.	Participants	Distributed onshore and offshore agile team members
8.	Collaboration	The onshore and offshore team members share information and ask queries using asynchronous tools
9.	Consequences	The asynchronous information transfer pattern has the following benefits: 1. It allows the onshore and offshore team members to exchange information when synchronous communication cannot be conducted due to working hour's time difference. This helps overcome the knowledge transfer challenges 2. It frees team members from waiting for an onshore team member availability to ask a query. This helps overcome the communication and coordination challenges 3. If the team members do not respond timely it can cause delays in the project
10.	Known uses	VTT Technical Research Centre of Finland and National University of Ireland conducted a research on two organisations that were developing a system together. One organisation was a customer organisation in U.S and the other organisation was a development organisation located in Ireland. Based on their findings the companies used asynchronous tools for communication. They used Wikis for storing documents and meeting minutes and used Emails for decisions and queries [32]. Similarly Valtech used Twiki for asynchronous communication [11]
11.	Related patterns	Asynchronous information transfer pattern is often used with global scrum board and synchronous communication pattern

Appendix 7: Synchronous Communication Pattern

See Table 13.13.

Table 13.13 Detail of asynchronous information transfer pattern

No.	Pattern element	Detail
1.	Pattern name	Synchronous communication pattern
2.	Intent	In order to discuss issues the teams use synchronous tools for voice, video conferencing, document sharing, application sharing, etc.
3.	Also known as	Synchronous knowledge transfer

(continued)

Table 13.13 (continued)

No.	Pattern element	Detail
4.	Category	Communication category, as this pattern helps the onshore and offshore team members to answer each other's queries within 12 h
5.	Motivation	The motivation of this pattern is to address the trust, socio-cultural, communication and coordination and knowledge transfer challenges. For example, when a team is divided over different time zones they may have queries about work but due to the time difference they cannot get a direct reply at that time so they use emails to communicate queries, which are then answered within 12 h max. Organisations have set standards for response time in order to avoid delays in work [31]
6.	Applicability	Use synchronous communication pattern when: • Team is distributed over different time zones
7.	Participants	Distributed onshore and offshore agile team members
8.	Collaboration	The onshore team and offshore team members share information and ask queries using asynchronous tools
9.	Consequences	The synchronous communication pattern has the following benefits: 1. It allows onshore and offshore team members to exchange information when synchronous communication cannot be conducted due to working hour time difference. This helps overcome knowledge transfer, communication and coordination challenges 2. Team members can ask each other questions which build trust and help understand each other's socio-cultural differences, which helps overcome trust and socio-cultural challenges 3. It frees team members from waiting for an onshore team member availability to ask a query. This helps overcome the communication and coordination challenges 4. If the team members do not respond timely it can cause delays in the project
10.	Known uses	CampusSoft is a UK based company that used synchronous communication when they moved to agile with their offshore suppliers in India and Romania. They used video conferencing facilities for planning sessions and later shifted to WebEx sessions and Go To Meeting so that they could share desktops with the remote team members. For daily Scrum meetings they preferred to use Skype call and made everyone wear headsets to make the meeting easier. For sprint review meetings they used sharing desktop tools as well as conference phones so that members from both end could talk with each other [13]
11.	Related patterns	Synchronous communication pattern is often used with global scrum board and asynchronous information transfer pattern

Appendix 8: Visit Onshore–Offshore Team Pattern

See Table 13.14.

Table 13.14 Detail of visit onshore–offshore pattern

No.	Pattern element	Detail
1.	Pattern name	Visit onshore–offshore team pattern
2.	Intent	Both onshore and offshore teams should quarterly/annually visit each other in order to build trust, exchange cultural values and improve team coordination

(continued)

Table 13.14 (continued)

No.	Pattern element	Detail
3.	Also known as	Travel onshore–offshore
4.	Category	Collaboration category, as this pattern helps the onshore and offshore team members to co-locate and understand each other and build a relationship, which improves team coordination
5.	Motivation	The motivation of this pattern is to address the trust, socio-cultural, communication and coordination, and knowledge transfer challenges. For example when a team is divided on different time zones they do not feel that they are both part of one team and they do not trust each other. They do not understand each other's cultural values and work ethics. In order to solve these issues the onshore and offshore teams visit each other to develop the feeling of trust and understand each other's cultural and working values. During these visits they attend training together as well as engage with informal activities to better understand each other. This helps build a bond between the team members, which results in good team coordination
6.	Applicability	Use visit onshore–offshore team when: • Team is distributed over different time zones
7.	Participants	Distributed onshore and offshore agile team members
8.	Collaboration	The onshore and offshore team members visit each other to improve team coordination
9.	Consequences	The visit onshore–offshore Team pattern has the following benefits and limitations: 1. It allows onshore and offshore team members to exchange cultural values with each other and work ethics. This helps overcome socio-cultural and communication and coordination challenges 2. It helps team members to feel they are part of one team, which develops trust among onshore and offshore team members. This helps overcome trust and knowledge transfer challenges 3. The travelling adds additional cost to the project budget
10.	Known uses	Ericsson is a Swedish multinational provider of communications technology and services. To build a XaaS platform and a set of services they used agile software development methodologies. The development team was distributed over 5 sites located in 3 countries. Four of the sites were located in Europe and one was located in Asia. They conducted workshops, which were attended by team members from different locations. The purpose of these workshops was to create a common vision for the whole organisation by setting common values as well also to improve the collaboration between the sites, thus build trust [14]
11.	Related patterns	Visit onshore–offshore team pattern is often used with collective project planning pattern as planning is better done when the whole team is co-located

References

1. Agile Requirements Change Management [Accessed on: 24th Oct 2016] http://agilemodeling.com/essays/changeManagement.htm.
2. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … & Thomas, D. (2001). Manifesto for agile software development.
3. Abrahamsson, Pekka, Juhani Warsta, Mikko T. Siponen, and Jussi Ronkainen (2003). "New directions on agile methods: a comparative analysis." In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pp. 244–254. IEEE.
4. Taylor, Philip S., Des Greer, Paul Sage, Gerry Coleman, Kevin McDaid, and Frank Keenan (2006). "Do agile GSD experience reports help the practitioner?." In *Proceedings of the 2006 international workshop on Global software development for the practitioner*, pp. 87–93. ACM.
5. Šmite, Darja, Nils Brede Moe, and Pär J. Ågerfalk (2010). "Fundamentals of Agile Distributed Software Development." In *Agility Across Time and Space*, pp. 3–7. Springer Berlin Heidelberg.
6. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: elements of reusable object-oriented software*. Pearson Education.
7. Kausar, Maryam and *Adil Al-Yasiri.* "Distributed Agile Patterns for Offshore Software Development" 12th International Joint Conference on Computer Science and Software Engineering (JCSSE), IEEE 2015.
8. Distributed Agile Patterns. [Accessed on: 24th Oct 2016] http://stp872.edu.csesalford.com/distributedagilepatterns.html.
9. Cottmeyer, Mike. "The good and bad of Agile offshore development." In *Agile, 2008. AGILE'08. Conference*, pp. 362–367. IEEE, 2008.
10. Berczuk, Steve. "Back to basics: The role of agile principles in success with an distributed scrum team." In *Agile Conference (AGILE), 2007*, pp. 382–388. IEEE, 2007.
11. Danait, Ajay. "Agile offshore techniques-a case study." In *Agile Conference, 2005. Proceedings*, pp. 214–217. IEEE, 2005.
12. Ramesh, Balasubramaniam, Lan Cao, Kannan Mohan, and Peng Xu. "Can distributed software development be agile?." *Communications of the ACM* 49, no. 10 (2006): 41–46.
13. Summers, Mark. "Insights into an Agile adventure with offshore partners." In *Agile, 2008. AGILE'08. Conference*, pp. 333–338. IEEE, 2008.
14. Paasivaara, Maria, Sandra Durasiewicz, and Casper Lassenius. "Using scrum in distributed agile development: A multiple case study." In *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*, pp. 195–204. IEEE, 2009.
15. Therrien, Elaine. "Overcoming the Challenges of Building a Distributed Agile Organization." In *AGILE*, pp. 368–372. 2008.
16. Kerth, Norm. "Project Retrospectives: A Handbook for Reviews." *Dorset House Publishing* (2001).
17. Poole, Charles J. "Distributed product development using extreme programming." In *Extreme Programming and Agile Processes in Software Engineering*, pp. 60–67. Springer Berlin Heidelberg, 2004.
18. Brown, Alan W. "A case study in agile-at-scale delivery." In *Agile Processes in Software Engineering and Extreme Programming*, pp. 266–281. Springer Berlin Heidelberg, 2011.
19. Avritzer, Alberto, and Daniel J. Paulish. "A comparison of commonly used processes for multi-site software development." In *Collaborative Software Engineering*, pp. 285–302. Springer Berlin Heidelberg, 2010.
20. Avritzer, Alberto, William Hasling, and Daniel Paulish. "Process investigations for the global studio project version 3.0." In *Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on*, pp. 247–251. IEEE, 2007.

21. Avritzer, Alberto, Francois Bronsard, and Gilberto Matos. “Improving Global Development Using Agile.” In *Agility Across Time and Space*, pp. 133–148. Springer Berlin Heidelberg, 2010.
22. Wildt, Daniel, and Rafael Prikladnicki. “Transitioning from Distributed and Traditional to Distributed and Agile: An Experience Report.” In *Agility Across Time and Space*, pp. 31–46. Springer Berlin Heidelberg, 2010.
23. Massol, Vincent. “Case Study: Distributed Agile Development.” TheServerSide.com (2004).
24. Armour, Phillip G. “Agile… and offshore.” *Communications of the ACM* 50, no. 1 (2007): 13–16.
25. Sutherland, Jeff, Anton Viktorov, Jack Blount, and Nikolai Puntikov. “Distributed scrum: Agile project management with outsourced development teams.” In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pp. 274a–274a. IEEE, 2007.
26. Räty, Petteri, Benjamin Behm, Kim-Karol Dikert, Maria Paasivaara, Casper Lassenius, and Daniela Damian. “Communication Practices in a Distributed Scrum Project.” *CoRR* (2013).
27. Yap, Monica. “Follow the sun: distributed extreme programming development.” In *Agile Conference, 2005. Proceedings*, pp. 218–224. IEEE, 2005.
28. Kussmaul, Clifton, Roger Jack, and Barry Sponsler. “Outsourcing and offshoring with agility: A case study.” In *Extreme Programming and Agile Methods-XP/Agile Universe 2004*, pp. 147–154. Springer Berlin Heidelberg, 2004.
29. Bose, Indranil. “Lessons learned from distributed agile software projects: A case-based analysis.” *Communications of the Association for Information Systems* 23, no. 1 (2008): 34.
30. Modi, Sunila, Pamela Abbott, and Steve Counsell. “Negotiating common ground in distributed agile development: A case study perspective.” In *Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on*, pp. 80–89. IEEE, 2013.
31. Vax, Michael, Stephen Michaud, “Distributed Agile: Growing a Practice Together,” AGILE Conference, pp. 310–314, Agile 2008, 2008.
32. Korkala, Mikko, Minna Pikkarainen, and Kieran Conboy. “Combining agile and traditional: Customer communication in distributed environment.” In *Agility Across Time and Space*, pp. 201–216. Springer Berlin Heidelberg, 2010.
33. Cristal, Mauricio, Daniel Wildt, and Rafael Prikladnicki (2008). “Usage of Scrum practices within a global company.” In Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on, pp. 222-226. IEEE, 2008.