Agile Software Development
Practices in Distributed Environments

Rabia Durrani
Mälardalen University,
Västerås, Sweden
rdi09001@student.mdh.se

Muhammad Atif Javed
Mälardalen University,
Västerås, Sweden
mjd09002@student.mdh.se

ABSTRACT

Agile is rapidly becoming one of the most popular software development methodologies across the globe. Companies throughout the world are making considerable efforts to shift from the traditional software development methods towards the agile practices. Distributed software development is an increasingly vital development approach for software companies throughout the world, as it brings additional opportunities. Moreover, the limitations of the localized projects can be overcome in the distributed projects. Distributed development is already burdened with several challenges and agile methodology brings further challenges. Till now, there has been a little empirical research on distributed agile software development. The paper reviews various sources and case studies for distributed agile software development, and discusses the challenges faced by such development teams. Possible recommendations for these challenges are presented as well.

Keywords

Agile software development, distributed systems development.

1. Introduction – Agile Processes

Agile processes are a group of similar development processes, including eXtreme Programming (XP), Scrum, Feature Driven Development (FDD), and Dynamic Systems Development Method (DSDM). Agile life cycle development processes evolved in the late 1990s. The term ‘Agile’ was not applied to them until the Agile Manifesto was written and signed in 2001 [1]. The agile manifesto stresses upon the value of “individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and agility in responding to change over following a prescribed plan”. Like other development lifecycles, agile development aims to improve ‘customer satisfaction and employee empowerment’, rather than improving the processes [3]. Agile processes are iterative in nature, utilizing a combination of certain project management and software engineering methods. In result, these processes continue to deliver additional software functionality(s) ranging from every one to four weeks.

Agile methodologies are used to produce higher quality software in a shorter period of time. Agile methodologies were developed to streamline the development process and remove barriers to accepting business requirement changes during the development process. Agile methodologies do not require that business requirements and design details be locked in for the duration of development. Agile SDMs share several features including prototyping, iterative development, and minimal documentation [8].

This paper aims to discuss the role of agile software development methodology in a
distributed environment, the benefits which make the agile development methodology preferable over traditional development lifecycles; the extended the use of agile practices from smaller to large-scale distributed projects, and the key issues of outsourced software projects and their possible solutions. This paper assumes the reader is familiar with the agile practices.

The next section characterizes what we mean by distributed development. That section is followed by a discussion the choice of appropriate software development methodology, challenged faces in distributed projects, agile practices in distributed environment, and key challenges. The paper ends with lessons learned and conclusions from the use case-studies and literature.

2. Distributed Software Development

In the modern software industry, distributed software development (DSD) has evolved and is becoming a common practice. Various factors, such as such as constant and parallel development across different locations, ease of formation of virtual teams and other business market benefits, have resulted in a rising need for DSD [5]. In the modern software industry, distributed development is not considered a new practice. Usually large software systems, constructed at several locations, consist of various teams participating and working from different areas. They usually work in parallel on some part(s) of the distributed project. For the integration purpose, an integration team oversees the assembly of each part. Developers across different destinations “work as editors”, evaluating various work “patches”, and decide about the final merging of these patches into the next releases work product [9].

3. Choice of an Appropriate Software Development Methodology

While developing software is not a simple routine itself, the choice of an appropriate software development lifecycle methodology makes it a more complicated one. Within large organizations, the choice of using the traditional software development life cycle has been unsuccessful and inadequate in preserving the quality of software product, as well as meeting the constraint of ‘time’ taken by the software development and marketing processes [2].

4. Benefits of Agile Development

Agile methodologies are said to have reduced the geographical and socio-cultural gaps in large scale, globally dispersed projects [4]. Agile methodologies stress upon ‘time-boxed’, short releases of work products in all iterations. It involves high levels of association and teamwork from virtually self-organizing teams. In distributed environment, such robust engineering practices which are fully adaptive to changes can prove to be very useful in catering to the limitations of the geographical and socio-cultural gaps [10].

Another case study describes some other benefits of using benefits of using agile process and distributed development. Good communication between all team members is vital in all agile methods projects. Distributed development comes with web-based collaboration activities, instant messaging services, etc. among the team members. As agile methods highlight “low-tech face-to face collaboration”, hence remote collaboration serves the purpose. Several configuration-management issues are minimized as agile practices always keep up with continuous integration. Developers always integrate their work items into main system, if they make any changes [9]. In distributed agile systems, normally, a “source code control system” is a
basic necessity, to carry out all the professional software development projects [9].

The table below shows results from a statistical analysis of the performance of distributed agile projects as compared to the non-agile projects. The distributed agile projects performed much better in terms of schedule and overall productivity [10].

<table>
<thead>
<tr>
<th>Table 1: Performance of Distributed Agile projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule Deviation</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>Distributed Agile projects</td>
</tr>
<tr>
<td>Distributed Non-agile projects</td>
</tr>
<tr>
<td>Confidence level</td>
</tr>
</tbody>
</table>

5. Problems Faced in Distributed Systems Environment

While adopting any development practices for globally dispersed projects, the following factors need to be considered first. These include size and nature of the project, the cultural differences and gaps distance among all the teams and the geographical distances [3]. While communication & coordination, software quality and schedule & cost overruns are some of the problems hindering both one-site and distributed software projects [5]. The geographical distance can highly affect the communication, as in distributed projects, the use of virtual modes of communication become more useful. While a track of all emails, faxes and memos has to be maintained, use of instant messaging software and video conferencing helps in reducing the distance. While agile practices focus more on mutual interactions and people, which relates to the question if the hindrance of geographical distances can be overcome while using agile development practices in a distributed environment.

Similarly, the factor of cultural differences needs to be significantly considered while applying agile methods in this scenario. Agile principles have said to adjust and settle well in the more transparent, open and more flexible American culture. Hence, these practices could not be easily applicable in geographic locations such as Asia, Latin America, and certain parts of Europe where more hierarchal culture exists in the organizations. Time differences in various zones have also been reported to cause hurdles in distributed projects. Communication forms the basis of agile project development, and any barriers to communication would definitely affect the working. Agile development methodologies have been reportedly successful in relatively medium and small scale projects run by comparatively small teams.

Other problems faced by distributed development projects include lack of clarity on project status, delays in feedback, loss of business or technical perspective, decrease in communication volume, higher documentation overhead and probable lack of trust [7].

6. Challenges for Large Dispersed Systems Using Agile Methods:

Large and dispersed systems face a lot of complexities due to the overall scope and complexity level. Normally, in agile practices, pair programming is used to transfer the system knowledge among the team members. A comprehensive set of domain and related technical knowledge and skill set comes with large scale distributed projects. In reality, any individual, solely, cannot grasp maximum area of the project. Hence, these systems do require documentation to transfer these complexities. Also, agile practices welcome ‘changes’, while in these large scale projects, project plans or direction is non-agile and difficult to change [6].

For various contractual and legal reasons, these projects ought to maintain track the
implemented project work and its details, whereas agile practices normally ignore these artifacts. Similarly, changes to the codebase require a whole lot of pre-requisites (changes need to co-relate to existing work-item, which is budgeted, approved, code review perfumed and code tested, etc.). Agile methods allow a more flexible way of carrying out changes, linking them closer to a release [6].

Numerous researchers have stated that “agile methodologies do not work effectively with large development teams”. From the results of research conducted in one study, no significant correlation was found between success level of implementation and the size of the project group [8].

7. **How to Face these Challenges Using Agile:**

These challenges can be addressed using some common agile practices in figure 1 [7]:

Distributed systems development comes with a lot of challenges itself. Even with agile development methods, the team has to be prepared to avoid potential pitfalls. The management and development teams, both, should be aware and prepared for these challenges, such as dealing with teams across multiple continents having different time-zones, the delays caused by the inability to have a live interaction, language and cultural differences, etc. The last of trust, which potentially rises due to the distance, cannot be left unchecked.

8. **Making Distributed Agile Work:**

To make agile practices successful in distributed projects, some areas of these techniques can be modified, so that they can be applied efficiently and easily in these systems. As studies show, some basic in-house agile techniques have to be applied as-it-is, others (for instance, volume & level of information sharing, documentation details, communication channels, project status tracking, and meeting frequency) can be modified [7].

8.1 **Establishing Several Communication Channels**

Instead of having a single communication channel, teams can have more than one communication channel. For instance, in Scrum having two channels of communication, with “status and task commitment” coming through the Scrum Master. The technical team leads can manage the design and technical communication.
This ensures efficiency in issues’ handling and constantly changing requirements [7].

8.2 Strict Communication Plan:
The daily scrums with the distributed teams can be reduced to reduce load of late nights/early mornings to manage real time overlap. In some cases, e-mails can be of good use for correspondence and quick feedback is considered mandatory, even if it does not directly respond to the request, and just acknowledges it.

8.3 Shared Electronic Work Spaces:
Personalized SharePoint site and project dashboard can be set up for all the updates about work and teams, contact information, status updates, announcements, discussions and documents. Mutual calendars can be established so that everyone is aware of all the local and off-shore deadlines. A common IT infrastructure can be established to make use of the shared workspace [7].

8.4 Dealing with Two Time Zones at Once
Studies have shown that if there are a lot of teams involved, belonging to different time zones, then it’s the best that at a time, only two teams collaborate. Work should be split accordingly, and hence, the teams can work more effectively.

8.5 Continuous Integration
For large and globally dispersed projects, integration turns out to be one big obstacle. The volume of code and the “variety of code sources (outsourced, packaged applications, open source, customizations)” is not easy to tackle. Therefore, integration should not, at all, be delayed. Following steps can help achieve early integration [6]:
  - “Organizing teams around building scenarios, not components”. This helps a lot, as if teams are focused in building their own components only, they become more concerned about their part of the system than the system’s overall success.
  - Keep the stakeholder involved in the feedback loop, as often as possible.
  - Differentiate between “release” and “early draft” of the release, so that customer’s expectations remain set, and early feedback can be received, after integration.
  - Simplify change management process.
  - Get all team members on one ground for early and continuous integration.

8.6 Document when Necessary
Though agile manifesto prefers working software over documentation, distributed agile teams should document “appropriately whenever necessary” [9]. It is not necessary to document to every minor details and work breakdown structures.

9. What to do Differently
“The agile manifesto provides is a good starting point to analyze software practices”. Today’s organizations are ‘customer-focused’ and ‘employee-focused’, having lesser organizational hierarchies, and more willing to adapt to the required changes [3]. According to a study, four principles of the agile manifesto are termed as feasible [3].

Team formation is a great challenge for remote agile teams. Since the aim is complete work synchronization, hence, work should be distributed across the team by “feature”, not by “discipline”. Remote test, development or management teams should be avoided. [9]. Teams should be kept together across the project.
The teams at various geographic locations need to accommodate for flexible hours when working in such projects [9]. Most of the practices of agile development run by a “common sense”. Hence, instead of stressing over the names and terminologies of agile methods, focus should be retained on attaining the main functionality (as sometimes, people used to the traditional methods get more nervous while working around agile practices). Moreover, testing activities can be run on a separate schedule. Ideal scenario is that the whole offshore team should be available to the onshore team for a portion of the day (at least two teams should have some portion of overlapping time slots) [9].

Fowler proposes “mutual seeding” and “maintaining visits between the distant locations”, which will aim in building and maintaining mutual trust and relationships as a basis for ‘efficient communication’ [11]. These short visits can help sustain a better communication and collaboration.

10. Conclusion

Agile development processes in a distributed environment require having the right people, in the right places, playing the right roles. Extending agile, from the history of its successful application at small one-site projects, to a distributed model can be a challenge. This tough goal can be attained by constructing the ‘right team’ with the ‘right skills’. With these and leveraging the proven agile tools and techniques, distributed projects can obtain incredible benefits in terms of cost and time management, productivity, risk minimization, improved quality [7].

Agile methods can prove to be of great competitive advantage by simplifying communication, reducing the time-to-market by facilitating the business, as it can respond quickly to the changing requirements by changing the software. Trying to distribute a development project by using agile development methodology is not an easy mission; yet, the advantages of deploying agile processes are far too many. Keeping in mind the nature and scope of project, and the size and number of development teams, modifications can and should be made in the agile manifesto.

11. References
