Analysis of Agile in Distributed Teams

Syed Nisar Hussain Bukhari, Ashaq Hussain Dar, Muneer Ahmad Dar
National Institute of Electronics and Information Technology, Srinagar/Jammu
Govt. of India.

Abstract

An agile methodology relies on getting releases out to users, on fixed dates, hopefully frequently. This has a number of benefits. You get feedback from users, and your product improves. You fix bugs before each release, so quality stays high. And, you establish credibility that you will deliver which reduces pressure on the development team and makes it easier to manage user requests. Unfortunately most agile methodologies or approaches assume that the team is located in a single team room. Until recently there has been little guidance about how to apply these approaches with a geographically dispersed team. Most software teams are now distributed. For example several teams working at different locations may be working on different parts of the same software product, perhaps with an integration team somewhere with the responsibility of overseeing the assembly of each part into a cohesive product. The overwhelming majority of software produced in the open source world is created by individual hobbyist developers working from home offices or student terminal rooms. This type of development is characterized by the fact that almost every developer is physically separated and that a small number of trusted developers are responsible for integrating submissions from a wide pool of contributors. These trusted developers effectively work as editors deciding which submitted "patches" to accept, and they decide how each accepted patch will be worked into the released product. The developers rarely meet, and integration of each patch or commit happens relatively infrequently, certainly by extreme programming standards. In this paper motivations to implement agility in geographically distributed teams will be analyzed. Also the factors that need to be addressed in implementing distributed agile techniques will be analyzed.

Keywords: distributed agile, artifacts, product, scrum, remote teams

1. Introduction

One of the fundamental tenets of any agile software methodology is the importance of communication between the various people involved in software development. Furthermore agile methods put a large premium on improving communication through face-to-face communication. As the agile manifesto states "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."[5] Extreme Programming emphasizes this with its practice of a single open development space where the team can work closely together. [6]

Another trend that's been grabbing the software development world recently is the move to offshore development, where much of the development work is done in lower paid, ahem more cost effective countries. Offshore development seems opposed to agile development in a couple of ways. For a start it immediately goes against the notion of physical proximity, since by definition offshore developers are a long way away. Secondly most offshore organizations favor the plan-driven approach where detailed requirements or designs are sent offshore to be constructed. Even the largest or most distributed teams can achieve the faster time to market, higher productivity, and higher team morale that the agile methods provide. [2]

Although agile processes are being used increasingly in many software development environments, some enterprises still havenȇt adopted Agile because of various concerns, especially about using it with distributed teams.

2. Motivation for distributed Agile

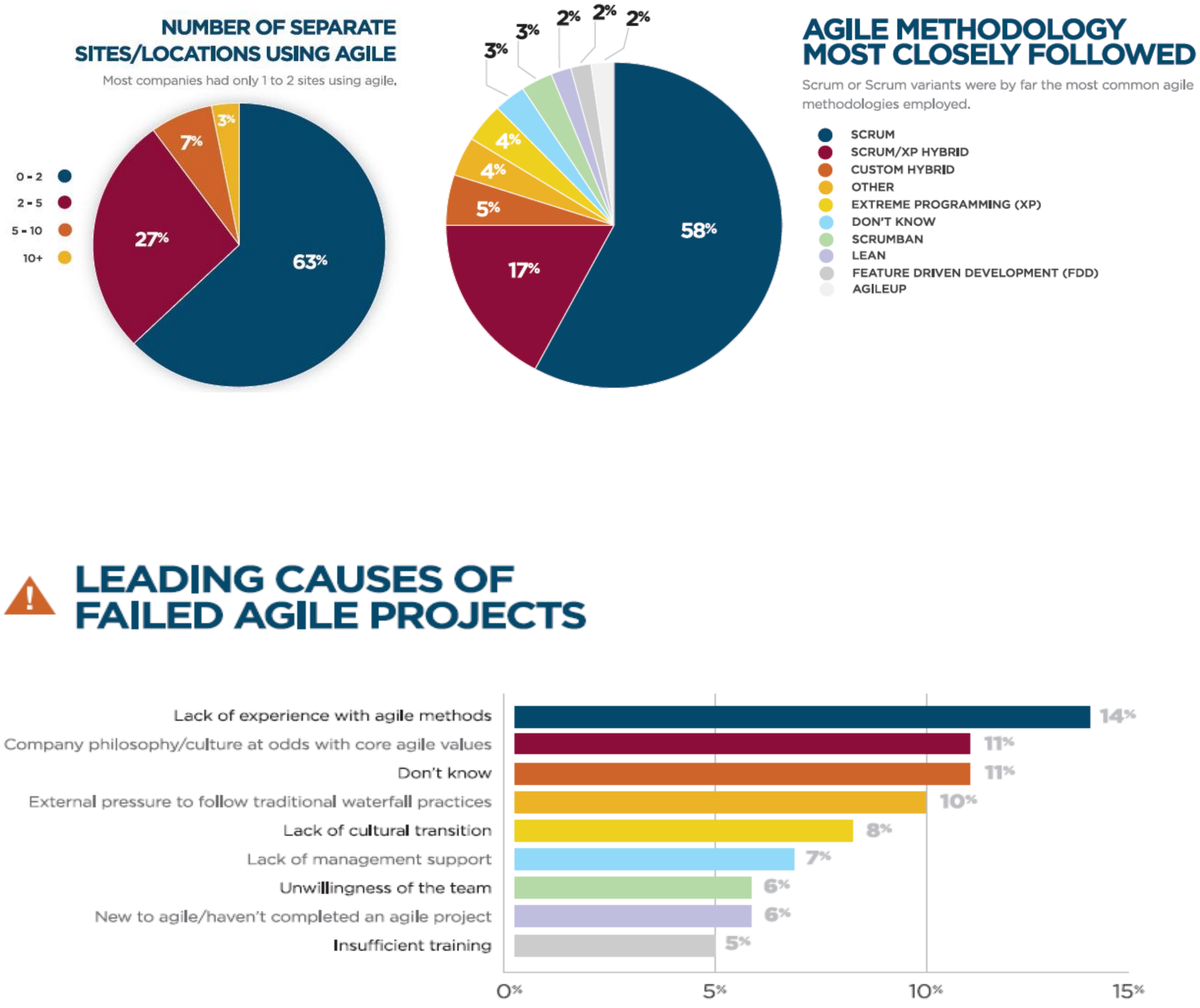
Teams co-locate because it maximizes their ability to communicate in person. Working in the same room is core to all the agile methodologies, including scrum. Scrum is unique as the minimum plan necessary to start a Scrum project consists of a vision and a Product backlog [1]. The best communication is face to face, with communications occurring through facial expression, body language, intonation and words. When a white board is thrown in and the teams work out design as a group, the communication bandwidth absolutely sizzles. [1] Given that communication is such a significant part of the efforts involved in delivering software, why then are distributed teams so prevalent? The answer speaks to the reality of doing business today: a companyȇs need to have a global presence, to access global talent and to develop outsourced options.

Effective distributed agile development is about minimizing the impact of distribution. Mainstream Agile processes work well for teams of 10 to 15 people, but what if your team is much larger? Paper-based, face-to-face strategies start to fall apart as the team size grows and the team distribution begins to span multiple time zones. You can still use agile processes for larger, distributed teams. You just have to be willing to adjust your processes to accommodate the reality of your teamȇs environment.

Finding the right mix of talent at every location is critical when the team is distributed. If you centralize one type of work at one location, you can lose opportunities to benefit from local talent. For example, you might know of an excellent developer you canȇt hire because your development team is in another country. Lost talent adds to your companyȇs total recruitment costs.

3. Surveys

In a survey conducted by VersionOne, a leading project management tool designed specifically for agile software development in 2010, more than half the survey respondents said they are currently using Agile with both co-located and distributed teams, or planning to do so in the future. Although co-locating your team is the recommended, optimal approach for implementing agile processes. Many teams are unable to do so for critical business reasons and must learn to follow agile principles and practices in a distributed development.



Figures (source: Version 1)

4. Research findings and observations

One of the projects ñiControl Systemò I have worked on used distributed agile methodology and I observed the factors that should be considered while implementing the agile in distributed mode.

1. Distribute workload evenly: Make sure all team members understand their role and have fairly equal workloads. In a distributed team, if the workload is uneven, you may risk the overall project delivery schedule if you ignore the imbalance. Uneven workloads can also cause bottlenecks. You should take complaints about workload seriously and be ready to balance the distribution of work carefully and quickly.

Maintaining relatively even workloads across the team is also an important part of keeping the team rolling. If someone is burning out because of an unrealistically high workload, move some tasks to another developer if possible. If someone doesnôt have enough work, find ways to keep the person engaged with the project. People at both extremes of the workload range usually lose focus and eventually become disengaged.

2. **Do pair programming:** You do need to modify some agile practices, of course, by moving agile process artifacts from their physical environment (white boards, sticky notes) to an online one. Microsoft SharePoint, commercial tools like Version 1, Microsoft Visual Studio Team System 2008/2010 and other online tools can be used to keep a team/project on track. Visual Studio 2010 includes project templates for Agile and scrum you can use to organize your user stories, product backlog and so on. It also includes a number of predefined reports, such as burn-down charts and overall status reports.

The way you communicate will change as well, as you supplement face to face meetings with IM, VOIP, videoconferencing, and wikis. It is important to get all team members or at the least the key members to meet each other during the project kickoff, and from time to time. [7]

Pair programming, where two team members sit side by side and work on the same code, is a challenge for distributed teams. When pair programming simply isnôt possible, you need to adapt the ñside-by-sideò part of it so that you can replicate the experience virtually, for example, by using a video-conferencing solution such as Skype or LiveMeeting.

If this solution isnôt possible, you need to replace pair programming with an equivalent practice, such as a show-and-tell hour or a daily developer scrum. In a show-and-tell hour, every team member demonstrates (using screen sharing) his or her work to the entire team and receives feedback. In a daily developer scrum, developers meet briefly to collaborate on technical issues or approaches. This practice fosters code sharing and reusability, and it usually results in greater team bonding

3. Strengthen the use of agile practices: By definition, agile approaches rely on a set of mutually supporting practices. When one practice doesnôt make sense for one team, the practice should be either abandoned or replaced. With distributed teams, enforcing and reinforcing a practice takes time and effort. Product owners and scrum masters who push a practice even when itôs clearly not working are seen as bullish or aggressive.

In a distributed environment, best practices around tool use play a vital role in enforcing Agile practices. For example, you can enforce code check-in comments by setting up a Team Foundation Server check-in policy, or you can use a centralized bug-tracking tool and mandate status updates before the daily scrum call to track the up-to-date status. For the first few weeks after you establish procedures like this, youôll notice that everyone follows the practices. As you get closer to release, people tend to bypass them. As a scrum master, you need to reinforce the use of these practices to make sure the project stays on track and to make them automatic for developers.

4. Give importance to Time Differences: Teams whose members work in various time zones face unique communication challenges. Even though regular scrums and overlapping work hours mitigate this problem to some extent, part of the work often gets delayed because clarification is needed or something requires reworking because one personʼs changes affected the work of others in a different location or changes were not propagated correctly end to end. Regular reworking like this can impact team health; morale can suffer when people are repeatedly asked to redo work. Sometimes more disciplined developer-to-developer handshakes and end-of-day status notes to the entire team are required.

Developer-to-developer handshakes, means that development teams should communicate the code check-in status and code areas they have changed during their work hours that the other side should be aware of. Software versioning and revision control tools like TortoiseSVN can be used here. They should share issues such as classes or methods changed, files changed and if the build was successful post check-in. They should also mention if there were conflicts in the code and whether they have shelved the code or checked it in. This way of working calls for more discipline since it has to be a regular activity, and developers tend to bypass this e-mail communication, assuming people already know what has changed.

In the iControl system, everyone exchanged end-of-day notes that contained the code check-in status, build status and any known issues that needed to be worked on or ignored when the developers in other locales started their day.

Help your team work with those across the world by first making everyone aware of the time zones team members are working in. Utilities or tools in Microsoft Outlook 2010, such as Time Zone Data Updater. Enabling multiple calendars in Outlook 2010, you can see the time in other time zones. Also you can use multiple clocks for the same purpose in windows 7. In e-mail messages, you should note the time zone you are in or referring to, for example, by including something like, ñWe shall discuss the update transaction API tomorrow at 7:00 a.m. ISTò

5. Make Remotely located teams more adaptable to communication: Agile teams typically rely on intensive person-to-person communication, both within the team and with the customer. Remote team members miss out on these face-time encounters, and their understanding consequently suffers. When distributed team members are working through a communication channel in which they canʼt see the person (or people) they are talking with, they need to be able to convey their agreements, disagreements and mood through their voice, tone, careful listening and general communication. You cannot do distributed projects without honest communication. It is a matter of establishing [the teamʼs] trust [7]

You can enhance the audio communication abilities of your distributed teams by providing the tools they need, whether instant messaging, VOIP, wikis or mobile phones. You can also make tools that allow for virtual face-to-face communication available, such as videoconferencing with Microsoft Lync 2010 and Microsoft Lync Server 2010.

5. Do’s and Don’ts for distributed agile

Do’s	Don’ts
Do work to maximize the available communication bandwidth available to your team. Provide communications toolsð like conference phone, Web cams, and hands free headsetsð for your team and help them adapt their existing practices to distribution.	Donôt continually reorganize your teams for each new project. Building teams takes time, building distributed teams takes even longer. Maximize your investments in team building by minimizing churn on teams.
Do plan to travel, especially at the projectôs pivotal points. Bring everyone together for the first couple of iterations, periodically during the project, and right before final release.	Donôt distribute the work by system components, focus on user stories. Avoid organizing distributed teams by functionð for example, the offshore test team.
Do provide tools to augment or replace those that only work within a team roomð like a work item tracking system to replace sticky notes on whiteboards.	Donôt let remote team members be forgotten in team meetings. Pair them up with a buddy and try putting everyone on the same footing by having all members call into conference calls at least occasionally.
Do focus on coaching. Make sure everyone understands why agile practices need to be adapted for distributed development.	Donôt forget to include everyone in frequent team retrospectives to identify what does and does not work for the team.

6. Conclusion

With distributed agile development it is possible to tap into new global markets and make best use of globally available talent, while potentially reducing costs. With organizations going global, distributed teams are becoming the norm. With additional efforts and some modifications, it can work well with distributed teams. It might even prove more successful than current software development methodology. The decision to distribute your project should be a conscious one .The risk/reward tradeoff needs to be clearly understood before deciding to distribute your team(s). Distributed agile development requires significant effort on the part of the team and support from management in order to be truly successful. One of the key success factors in an agile project is the high level of communication made possible by having the team sit together. If your team is distributed they need to make deliberate efforts to replace as much of this lost communication bandwidth as possible and adapt and augment their practices to account for this loss. Agile development is hard and requires a great deal of discipline. Distributed development is harder still and requires yet more resolve to stay on track. Make sure that your team has someone with a clear mandate to coach them. Provide distributed teams with the right tools to work as effectively as possible and remove as many of the barriers created by distribution as possible. Expect to experiment with how you use these tools, be it conference phones, collaboration software, or work item tracking tools.

References

[1] Ken Schwaber, The Enterprise and Scrum, Microsoft Press, ISBN 073561993X
[2] Dean Leffingwell: ñScaling Software Agility, best practices for large enterprises
[3] http://www.versionone.com/state_of_agile_development_survey/10/page3.asp
[4] Craig Larman & Bas Vodde, Scaling Lean & Agile Development: Successful Large, Multisite & Offshore Products with Large-Scale Scrum, Addison-Wesley Professional, ISBN 0321480961
[5] http://www.dilitech.com/index-Approach-agilepwsl.html
[6] http://www.buffalosoft.com/webdevelopment/tag/conventional-software-development
[7] Alberto Sillitti, Xiaofeng Wang, Angela Martin, Elizabeth Whitworth ,ISSN 1865-1348, Agile Processes in Software Engineering and Extreme Programming, 11th international conference,XP 2010 Trondheim,Norway,June 2010 proceedings.