Applying Agile Principles for Distributed Software Development

Prof. Rashmi Phalnikar  
IT Dept.  
MIT COE, Pune, India  
rashmiphalnikar@yahoo.co.in

Prof. V.S. Deshpande  
HOD, IT Dept.  
MIT COE, Pune, India  
vsd_deshpande@rediffmail.com

Prof. S.D. Joshi  
Bharati Vidyapeeth, COE, Pune India  
sdj@live.in

Abstract

The necessity of finding right skilled people, sharing resource and limitation on cost has made distributed software development indispensable. In a distributed development project, but are working collaboratively toward the outcome. Such offshore service providers follow the traditional process models. Agile practices promote development iterations, open collaboration, and process adaptability throughout the life cycle of the project. Adapting these practices in a distributed environment can help distributed development tackle the challenges of cultural incompatibility, leadership struggle and lack of trust. This paper describes the benefits of using Agile process and Scrum the iterative incremental process in Distributed Software development, and proposes two team structures for its implementation.

1. Introduction

Distributed development is not a new practice. Large software systems built across multiple sites has several teams, hobbyist and developers working from different locations on some part of the distributed project. An integration team will have the responsibility of overseeing the assembly of each part into a cohesive product built. This type of development is characterized by the fact that the distant developers effectively work as editors deciding which submitted "patches" to accept, and decide how each accepted patch will be worked into the released product.

Such development can be categorized into:

i. Distributed development: This involves co-operation between several teams located at different sites. It will comprise efforts where the bulk of the work is outsourced to developing countries with a small team of consultants working on site with the business stakeholders.

ii. Dispersed development: Dispersed development refers to individual developers located separately, working together over a network. This includes project teams that may include small group of experienced developers working from home and specialists working from private office.

2. Challenges to distributed development

The reason why most people look to offshore and distributed development is to reduce costs, noting the significantly lower rates that one gets from offshore vendors. Another benefit of offshore that's coming up is the use of 24 hour development to reduce time to market. The benefit is that having the code base at all hours of the day, helps functionality gets written faster.

However, offshore makes communication harder both due to the distance, which makes it difficult to meet face-to-face, and the time zone offset. Both of these increase the likelihood of building the wrong functionality as miscommunications occur over requirements. The distance between development and business also reduces the motivation of the development team, since they have no personal relationship to build on.[8]

The pitfalls of offshore software development are many. These are some of the most significant challenges to overcome:

i. Decreased communication bandwidth. Due to time zone differences between western business centers and Asian counterparts, there are very few hours in the day when project participants are in both office locations at the same time. This factor, as well as the current cost and quality of telecommunications, serves to significantly decrease the volume of communication between offshore and onshore teams.

ii. Decreased visibility into project status. It's often difficult for project managers and business sponsors to get an accurate sense of project progress and status. Measuring project progress is a problem when the project team is collocated and onshore, and is only
made worse when the project is on the other side of the world.

iii. Configuration management. "Bringing it all together" for implementation is one of the most difficult parts of any software project. Many teams that have built components offshore have been beset by problems when the time came to integrate the offshore and onshore pieces into a working system.

iv. Disconnection on project estimation. Customers, managers, and developers all estimate projects differently. In an offshore situation, where the development team may never meet the project manager or customers, the standard deviation of traditional project estimates can vary wildly. Also, it can be very difficult to assess the accuracy or reasonability of estimates as the project progresses.[7][8]

3. Agile Manifesto

Agile methods are a response to the drastic degree of change in the modern business and IT environments. These highly dynamic environments demand software development teams that can respond to change and continuously deliver business value. Some of the key points of Agile Manifesto are:

- Customer satisfaction by rapid, continuous delivery of useful software.
- Working software is delivered frequently and is the principal measure of progress.
- Late changes in requirements are welcome.
- Close, daily cooperation between business people and developers.
- Face-to-face conversation is the best form of communication.
- Projects are built around motivated individuals, who should be trusted.
- Continuous attention to technical excellence and good design.
- Simplicity.
- Self-organizing teams.
- Regular adaptation to changing circumstances.[10]

The most important of these principles is: “The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.”[11] The manifesto was put together in 2001; when, offshore development (the primary scenario for distributed development) was just beginning to gather momentum. Distributed development however is unable to support this. 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.

The waterfall model used by distributed development is the inflexible division of a project into separate stages, so that commitments are made early on, and it is difficult to react to changes in requirements. Iterations are expensive. Agile methods, in contrast, produce completely developed and tested features (but a very small subset of the whole) every few weeks. The emphasis is on obtaining the smallest workable piece of functionality to deliver business value early, and continually improving it/add further functionality throughout the life of the project.

Agile development has little in common with the waterfall model.

Scrum is an Agile software development process designed to add energy, focus, clarity, and transparency to project teams developing software systems. It allows teams to operate in close proximity to foster rapid system evolution. It was designed to achieve a hyperproductive state where productivity increases by 5-10 times over industry averages and many collocated teams have achieved this effect.[10] The question is whether distributed, outsourced teams can consistently achieve the hyperproductive state.

Other challenges when combining Agile and offshore development include distributed communications, time zone differences, language and cultural barriers, security problems, lack of direct interaction with the business subject matter experts. All of these challenges are significant and require directed effort if they are to be overcome.

4. Benefits of agile distributed development

Few practical considerations when combining Agile process and distributed development are:

i. Communication: Good communication between all participants is vital in all Agile methods projects.

ii. Security: Confidential data and intellectual property are prime concerns of Agile project team members and use of broadband routers to create a secure, private network will be helpful.

iii. Remote Collaboration: As agile methods emphasize low-tech face-to-face collaboration, in distributed environment, these are replaced with a kind of server set aside for common file storage, web-based collaboration and a private instant messenger service. Also some form of project planning and tracking tool is required.

iv. Code Repositories: A source code control system is a requirement for all professional software development projects.
v. Use of Technology: VOIP, Webcams, Screen Sharing Software. Technology, it seems keeps advancing. At work, we use a combination of Skype/Vonage, WebCams, and GoToMeeting/Webex, to achieve this on a daily basis.

vi. Spread Cultural Awareness: Regular visits back and forth between the two teams are advised. It really helps to know other team members as teams may be spread across the world at two places that are culturally very different. It avoids awkward situations during communication.

vii. Advanced Features - Certain features that can be used to increase efficiency for example, a rules engine can automatically create a task for code review or a list of commonly used acronyms will help developers before committing code.

viii. Proxy/Caching - The source control system have an ability to manage replicated servers on distributed sites. This speeds up operations when interacting with the source control system.

Offshore is wrought with challenges and parts of agile development appear to make offshore even harder to manage. Fortunately, some aspects of agile development will tackle this.

If practiced with discipline, continuous integration— wherein developers integrate their code and build the entire system whenever they have made changes— should reduce or even eliminate configuration-management issues. Check-in is preceded and followed by build verification tests (unit tests) as well as automated functionality testing. This solves small integration headaches instead of trying to solve huge ones at the end of the project.

Iterative development with frequent delivery to the customer is a core practice in agile software development, and one that directly addresses one of the major challenges of offshore development—decreased visibility into project status. Frequent deliveries allow project managers, customers, and users to measure project progress by the amount of working software they have received.

5. Proposed team structure

The key challenges for designing a team for Agile offshore/distributed distributed are:

i. Complete work synchronization is expected. Process is to be managed and coordinated.

ii. Distribute work across the team by feature not by discipline. Avoid remote test/development/management teams.

iii. Use continuous integration to anchor team. Utilize defense in depth and Run additional builds to account for people in different time zones.

iv. Source code control systems also suffer from network latencies caused by remote teams. Make sure teams have good bandwidth or use proxies.

v. Keep teams together across projects.

vi. Distribute teams to increase talent not cut costs.

vii. Be prepared to do additional work to negate the impact of team distribution.

viii. New scrum teams are required for documentation, integration and validation.

ix. Documentation team can participate in all planning meetings.

x. Integration scrum team is made responsible for nightly builds and installation.

xi. Scrum masters are allocated to each location.

xii. A routine meeting of scrum members will help solve any ambiguity.

xiii. Validation team takes care of testing and validation

Offshore companies need to accommodate for flexible hours when working on a Agile distributed developmental project to accommodate for the time zone differences. Activities such as testing and deployment can be seen as separate and therefore can be run on a separate schedule, but that does not scale well within the Agile team. And so, effort should be taken to make the whole offshore team available to the onshore team for a portion of the day.

A typical team that works on a distributed Agile process may look like the one shown in Fig 1.

All members of the team reside in the same location and in most cases they sit physically close to each other. Colocation improves intra-team communication and eases the flow of information. In addition, the degree of ceremony is reduced and project management overhead is kept under check.

Fig. 1: Model for a process using Agile principle

To overcome the drawbacks of the standard agile model, we propose two team structures to exploit the advantages of Agile process in distributed software development.
- Partial Offshoring
- Complete Offshoring

Model 1: Partial Offshoring:

In this model the design team is located onshore and it sends the requirements overseas.

The offshore team responds with fully-tested and working code. The onshore team then performs the user acceptance tests and validates the requirements and the defects.

The partial offshoring model aims at creating quality requirements and application architecture by keeping key members of the business analysis and architecture team close to the customer (onshore) while shipping lower-level tasks such as programming and testing overseas. In this model the onshore team sends requirements and architectural designs along with other supporting documentation to the team overseas and expects application code which complies with the requirements is fully tested. Because the offshore team lacks architects, business analysts and most importantly the onsite customer, there is plenty of expensive inter-team communication between the onshore design team and the offshore implementation team.

This communication carries a high-price tag because it involves people from multiple disciplines and specializations and the extra overhead of putting everything in writing as opposed to verbal communication. For example architect-to-developer communication is more expensive than the architect-to-architect communication as it involves architects creating elaborate design documents and other supporting materials to ensure proper coding of their designs overseas.

Model 2: Complete Offshoring:

This model proposes to provide as much colocation as possible by shipping the entire production team overseas.

The only person left onshore is the customer. The customer sends features and the offshore team responds with fully tested working code. This scenario is arguably the most cost-effective among all.

The complete offshore model overcomes the communication latency problem by placing key members of the production team together at the offshore location. This reduces multidisciplinary inter-team communication while improving intra-team communication. In this model the only key person onshore is the customer and the primary communication takes place between customer and the offshore business analysts and architects.

Since the inter-team channels of communication are kept to a minimum and the intra-team communication is boosted, this model is better suited for a long-term Agile development.

One disadvantage of this model is the multidisciplinary communication taking place between the customer and the offshore business analysts and architect team. This communication is the most important and studies have shown that misunderstood or misinterpreted requirements are among the chief causes of offshore project failures. Subtle differences of terminology or cultural nuances can become an obstacle in an otherwise well-executed project.
6. How to succeed at offshore agile development

We discuss some additional tips to exploit the advantages of Agile in distributed software development

1. Documentation: Document appropriately whenever necessary. Document the features that are queued up for the upcoming iteration—to a level of detail that will enable the development team to understand the features enough to break them into tasks, estimate them, and get started.

2. Proxy Customers: Some projects large enough to support proxy customers can benefit by the cross-functional collaboration. A good proxy is one with interests and abilities in technical and business spheres. He should interact and communicate equally well with the technical and business project members.

3. Getting Questions Answered - the First Time: One issue is the response time required on posing questions to people who operate in radically different time zones. Adjusting work schedules to ensure at least a few hours of to discuss time-critical issues in real time can reduce this.

4. Usage of message boards or Wiki instead of email may considerably reduce time. Collaboration systems, besides IM tools, net meetings and video conferencing can really help bridge the communication gap. Audio recordings can be published online across the world. Regular scrum meets can be adjusted to meet all time zones.

7. Conclusion

Although, nothing can replace a true ‘face-to-face’ conversation (yet), the true message of the models proposed is to eliminate the inefficiency of 'non-face-to-face' communication: which is slow and prone to misinterpretations. But a mix of technology and best practices can definitely come very close to eliminating the inefficiencies of ‘non-face-to-face’ communication as discussed above.

Agile methods can provide a competitive advantage by delivering early, simplifying communication and allowing the business to respond more quickly to the market by changing the software. Trying to distribute a development project in an agile way isn’t easy and will involve compromises but the advantages are far too many.

Clearly, a number of challenges are introduced by offshore development, and some of them are likely to be made worse by using an agile approach. Even if you allow that agile teams are able to produce software more rapidly, there is some question as to whether the inevitable delays caused by reduced communication bandwidth counterbalance the speed of agile teams. This may depend entirely on the nature of the project and team structure.

8. References


