Can a distributed project be agile?

Tobias Pfeiffer - Blekinge Institute of Technology
Email: topf11@student.bth.se

***Abstract*—Both agile software development and distributed software development are recent trends in software engineering. Both approaches essentially aim at a shorter time to market, more quality and less costs. The question, which remains to be answered, is whether these two approaches can be effectively combined to research an even greater benefit. In this paper ideas and facts of current research will be presented .**

***Index Terms*—agile, distributed, global, software development**

I. Introduction

AGILE and distributed development are two of the most recent developments in software engineering. Both are perceived to achieve great benefits, so the desire to combine both approaches in order to gain even greater benefits exists.

Agile and distributed projects are not just a theoretical concept but rather reality. In the annual "The State of Agile survey", conducted by VersionOne[1], the percentage of companies reporting that teams in their company are working distributed rose from around 57% in 2007 and 2008 [1], [2] to 58% in 2009 [3] and finally to 65% in 2010 [4].

At first this paper will briefly introduce agile development and distributed development. Afterwards the challenges of combining the two methodologies will be elaborated. Then the current approaches to distributed agile software development will be analysed. Then a new idea for starting an agile distributed project will be presented. The findings of this paper will then be concluded in the last section.

II. Agile Software Development

Agile Software Development is a development philosophy driven by the thought that requirements always change and that huge upfront specifications (as practised in the waterfall model[2]) are therefore wrong. Agile development processes aim at being lighter processes in order to respond to change much faster and better than traditional development processes. Agile comes in many flavours, the most well known ones are Scrum [5], Extreme Programming [6] and Lean. The core principles of these development methodologies were summarized and defined during a famous gathering in 2001 in a document known as "The Agile Manifesto"[3], where the following values are emphasized:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

[1]http://www.versionone.com
[2]http://www.waterfall2006.com
[3]http://agilemanifesto.org

The agile manifesto also highly emphasizes the value of face-to-face communication and co-location. So it is stated in the agile manifesto, that "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation." Moreover in order to realize the value of customer collaboration a customer should always be nearby, the so called on-site customer.

Agile stresses the value of fast feedback. The development principles of agile development processes also include developing software in iterations (usually one week to four weeks) and delivering incremental value. Therefore they are also often referred to as iterative and incremental development, however those principles are not as young as many think, they date back to at least 1968 [7].

III. Distributed Software Development

Distributed Software Development refers to the development of a software on more than one site. Usually there is on-shoring, where the all teams still work in the same country, and off-shoring, where teams work together across country boundaries. The latter is usually referred to as "Global Software Development", which will be analysed in more detail here.

Global Software Development (GSD) is an effect of globalization as a whole. Companies are looking to outsource (working with a subcontractor) or insource (working with/founding a subsidiary) software development activities to other countries for a multitude of reasons. Mostly the following perceived advantages of GSD are named [8]:

- Reduced Development Costs
- Cross-site modularization of Development Work
- Access to large skilled labor pool
- Closer proximity to market and customer
- Innovation and shared best practice
- Leveraging Time-zone Effectiveness

Whereas the last two are assumed to be mythical benefits, the others are perceived as just being partially realized [8]. A big reason for this are communication problems, mostly emerging from the distance between the teams. Distance in this context not only means the geographical distance but also the temporal distance (time zone difference) and the socio-cultural distance [8]. Of course this also results in a greater effort for coordination, as delivering messages in this environment is much more difficult than in a face-to-face communication. This can especially be observed with modification requests, where research has found that modification requests, that are spread across more than one site, take more than 2.5 times the time to complete compared to a single site modification request [9].

Of course one of the main issues for distributed development lies in its very nature, as the team is not co-located. Co-location has been proven to be a major success factors for software development [10]. The top issues in distributed development were identified as follows [11]:

- *Communication* Lack of effective communication mechanisms
- *Cultural* Conflicting behaviours, processes, and technologies
- *Project and process management* Difficulty synchronizing work between distributed sites.
- *Security* Ensuring electronic transmissions confidentiality and privacy.
- *Strategic* Difficulty leveraging available resources
- *Technical* Incompatible data formats and exchanges.

IV. Challenges of Combining Agile and Global Software Development

Agile methodologies heavily rely on co-location, shown with practices like pair programming and the daily stand-up meeting (also called "Daily Scrum"), which is one of the most widely adopted techniques. It is a short meeting where everyone in the team answers three simple questions [5]:

- What did I do yesterday?
- What will I do today?
- What impediments are in my way?

This practice is difficult in a distributed team setup, as not everybody can be physically present and also the time difference is a big problem.

Moreover the focus is turned towards working software and informal communication with co-workers rather than documentation. This lack of documentation can be a handicap when working in a distributed environment as information has to be made accessible to all team members [12]–[14].

At least when outsourcing there is another problem: Some practices require all the teams to use the same tools. For instance the practice of a shared code base requires all teams to use the same version control system and the practice of continuous integration requires all teams to use the same build tool [14]. Furthermore these systems have to be set up in a way that they are accessible fast and easy for every developer, in order not to restrain their work flow. Additionally the temporal distance may impose the challenge of a twenty-four hour availability, complicating maintenance [15].

V. Current approaches for combining Agile and Global Software Development

However there are not just challenges when trying to combine the two approaches. Some agile practices can greatly help with problems in GSD. It has been found that the most commonly used practices in a distributed agile environment are Continuous Integration, daily stand up meetings and pair programming [16].

The practice of continuous integration for instance can help with the problem of the "Big Bang" at the end of the project. By continuously integrating the code of all sites it can be made sure that no major issues occur [14], [15]. Furthermore the practice of Test Driven Development (TDD, also called Test First Development) can ensure a high test coverage of the code instantly notifying everybody of a problem in the form of a failing test [14]. Both technologies give a fast feedback on changes, TDD for instance makes it apparent when expectations, that were set by the stakeholders or a developer, were broken by a recent change.

Furthermore it is suggested, that agile methodologies increase the quality of communication at least inside of a team [17], [18]. A key aspect of agile methodologies is reflecting about the work. It is common to have such reflections at the end of a meeting and at the end of an iteration (often called "retrospective"). These reflections help the team to identify problems in the way they work and allow them to adjust accordingly.

The short iterations of agile development processes make the progress apparent. This benefit makes it obvious when a project isn't progressing as expected and measures can be taken [14]. As such the daily stand up meetings have been proven effective [16], however this often comes at the price of shifting working times so all sites can attend [13].

It is recommended for loosely coupled teams to have one or more ambassadors from the remote team at the home site and from the home site at the remote site [15], [19]. These ambassadors are there to facilitate communication and also to establish trust between the remote team and the home team. Ambassadors however should be exchanged every few months, so that one person does not have to stay away from his family for too long and does not loose the connection to his home team.

It is apparent that the level of documentation of an agile development process has to be increased in order to fully work in a distributed team set-up. The informal information exchange emphasized by agile processes normally just works for the people at one site, it is rarely shared with the other sites. As a result of this, practices that facilitate knowledge sharing are emphasized and official communication channels become more important [13]. Therefore Fowler recommends a wiki as its unstructured nature allows the teams to apply a structure of their liking for sharing information [15].

As mentioned before, communication is crucial in a setup with distributed teams. Therefore communication channels should be plentiful, ranging from instant messaging, over a phone at every work place to good video conferencing tools [13], [15]. There should not be a significant overhead in setting up a meeting across all locations [12]. This of course is another additional cost distributed software development often introduces.

Travelling so that teams can meet and engage in face-to-face communication has been proven significant for the success of distributed projects, so it is suggested to travel often [12], [15]. Especially important phases are the start of the project, in order to get to know each other, and the end of the project in order to smoothly finish it. However visits every few months are also encouraged.

The following two subsections will shortly present the findings of two case studies, that employed a successful distributed agile set up while employing approaches not found to be very

common in the reviewed research papers.

A. Distributed Scrum Type C

Sutherland et al. describe a project which they describe as: "For a globally dispersed team, it is one of the most productive projects ever documented at a run rate of five times industry average." [20]. This was accomplished using a distributed Scrum Type C, which means that unlike most implementations of Scrum in a distributed environment the distributed teams weren't loosely coupled teams but rather virtual teams.

Each team consisted of members from almost all remote sites. However the people mainly responsible for coordination were co-located or close. All Product Owners were based in Utah whereas all Scrum Masters were based in Utah, Colorado or Canada [20]. This way they could conduct Scrum of Scrums [5] and other planning activities in a face-to-face manner. The teams had stand up meetings every day. However these were modified, the team members in the remote location had a local stand up meeting at the start of the day and a global meeting at the end of the day. For communication purposes however they found it to be necessary, to write their answers to the questions down before the meeting and distribute them.

B. The Distributed Agile Transition Model

A four-stage model towards adapting Agile in a distributed environment has been proposed [19]:

1) *Evaluation*, different aspects of the project are analysed in order to determine whether to distribute the project and if so how.
2) *Inception*, infrastructure is set up and training on agile and the problem domain is provided. Many representatives from the remote teams are at the home location and the remote team has a low degree of self-organization.
3) *Transition*, some remote team members move back from the home location to the remote location. They are now familiar with the customer and the home location team, which facilitates communication. The degree of self-organization of the remote team increases.
4) *Steady State*, more team members move back to the remote location, which is now completely self-organizing.

At Wipro, this approach has shown to lead to less schedule derivation and more overall quality, whereas other metrics haven't been impacted, compared to conventional plan-driven methods [19].

VI. A new take: The agile inception deck

Looking at the current approaches I found that it seems to be a common practice to bring the team together at least for the start of the project. Remembering my favourite technique for beginning a project, the agile inception deck, I thought that it might be very well combined with GSD.

A. What is the agile inception deck?

The agile inception deck is "a collection of ten tough questions and exercises you'd be crazy not to do and ask before starting any project.", as described by Jonathan Rasmusson in his book "The agile Samurai" [21, part II].

It is designed for anyone directly involved in the project (like the stakeholders and the development team) to get together and define the core characteristics of a project to ensure that everybody has the same idea of what the project is like, why the project is done and what to expect from the project. Creating an agile inception deck can take from a couple of days to two weeks [21]. Some of the exercises conducted are the following:

- *Ask: Why are we here?* Define clearly why the project is done and what the main drive that lead to a need for this system was.
- *Create a NOT list* Be clear on what is in scope, and be even clearer on what is not in scope and therefore will not be developed.
- *Size it up* Give a rough estimate of how long the project is going to need to be completed. This is not a commitment, just a guess so that everybody has an idea of how complex the project will be.
- *Ask what keeps us up at night* The goal of this exercise is to identify major project risks early on and talk about it, getting to know your co-workers better.
- *Design a product box* If the product you built could be bought in a store, what would it look like? This helps identifying the main features of the product.

B. How to combine the agile inception deck and global software development

I imagine that the agile inception deck could be conducted during the first weeks of the project when the teams are co-located. The exercises not only helps everybody to get the same understanding of the project but it can also be a great technique to get to know each other much better. Some of the exercises, like "Design a product box", are rather informal and allow the team to creatively work together, having fun in the meantime.

Once the inception deck is completed it represents a set of shared goals and information which should be available to everybody and updated when needed. Ideally it should also be part of a visible workspace [21, chapter 11], a workspace where everybody can see the most important information like shared values, story board, burn down chart and of course the inception deck.

VII. Conclusion

In a review of empirical research papers it has been found that over 80% of the examined agile distributed projects have been a success, whereas under 5% have been a failure (the success of the other projects is listed as NA) [16]. This statistic certainly suggests that distributed development can be done agile. Also other sources document a successful incorporation of agile distributed development in a multitude of different companies like Microsoft [12], [18], Thoughtworks [15] and many other companies [13], [19], [20].

In those studies it was said that an agile approach showed an improvement over traditional plan driven methods. However

agile methods have to be adopted in order to be successful in a distributed environment, these adjustments work but are not always easy to achieve. For instance the recommended high amount of travelling (see V) comes with major costs, although it is said that these efforts are really worth their cost.

Personally I highly believe that agile methodologies can be effective in distributed environments and I think that distributed agile projects can still do better, as agility is all about adopting to change. I would like to see some new ideas and approaches being explored, like the incorporation of the agile inception deck or the highly loosely coupled "2-Pizza"-teams at Amazon[4].

All in all I think that agile distributed projects may be conducted effectively in many environments. However more research in this sector is needed and there is still much room for improvements and new ideas.

REFERENCES

[1] VersionOne. 2nd annual survey the state of agile development. 2007.

[2] VersionOne. 3rd annual survey : The state of agile development. 2008.

[3] VersionOne. 4th annual state of agile development survey final summary report. 2009.

[4] VersionOne. 5th annual state of agile development survey final summary report. 2010.

[5] Jeff Sutherland and Ken Schwaber. The scrum papers. 2007.

[6] Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, us ed edition, October 1999.

[7] Craig Larman and Victor R. Basili. Iterative and incremental development: A brief history. *IEEE Computer*, 36(6):47–56, June 2003.

[8] Pär J. Ågerfalk Eoin ÓConchúir, Helena Holmström Olsson and Brian Fitzgerald. Global software development: where are the benefits? *Commun. ACM*, 52(8):127–131, 2009.

[9] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. Distance, dependencies, and delay in a global collaboration. In *CSCW*, pages 319–328, 2000.

[10] Stephanie D. Teasley, Lisa Covi, Mayuram S. Krishnan, and Judith S. Olson. How does radical collocation help a team succeed? In *CSCW*, pages 339–346, 2000.

[11] Kenneth E. Nidiffer and Dana Dolan. Evolving distributed project management. *IEEE Software*, 22(5):63–72, 2005.

[12] Ade Miller. Distributed agile development at microsoft patterns & practices. 2008.

[13] Balasubramaniam Ramesh, lan Cao, Kannan Mohan, and Peng Xu. Can distributed software development be agile? *Communications of the ACM*, 49(10):41–46, October 2006.

[14] M Paasivaara and C Lassenius. Could global software development benefit from agile methods? *2006 IEEE International Conference on Global Software Engineering ICGSE06*, pages 109–113, 2006.

[15] Martin Fowler. Using an agile software process with offshore development. Last updated on 18th July 2006, retrieved 19th September 2011 from http://martinfowler.com/articles/agileOffshore.html.

[16] Samireh Jalali and Claes Wohlin. *Agile Practices in Global Software Engineering - A Systematic Map*, pages 45–54. IEEE, 2010.

[17] Minna Pikkarainen, Jukka Haikara, Outi Salo, Pekka Abrahamsson, and Jari Still. The impact of agile practices on communication in software development. *Empirical Software Engineering*, 13(3):303–337, 2008.

[18] Andrew Begel and Nachiappan Nagappan. Usage and perceptions of agile software development in an industrial context: An exploratory study. In *ESEM*, pages 255–264. IEEE Computer Society, 2007.

[19] Kalpana Sureshchandra and Jagadish Shrinivasavadhani. Adopting agile in distributed development. *2008 IEEE International Conference on Global Software Engineering*, pages 217–221, 2008.

[20] Jeff Sutherland, Anton Viktorov, Jack Blount, and Nikolai Puntikov. Distributed scrum: Agile project management with outsourced development teams. 2007.

[21] Jonathan Rasmusson. *The Agile Samurai: How Agile Masters Deliver Great Software*. Pragmatic Programmers, 2010.

[4]http://tech.groups.yahoo.com/group/leanagile/message/2908