Agile Practices in Offshore Outsourcing – An Analysis of Published Experiences

Joachim Sauer
Software Engineering Group, Department of Informatics, University of Hamburg and C1 WPS, Ltd.
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
sauer@informatik.uni-hamburg.de

Abstract. Agile practices are applied successfully in several of today's software development projects. Their use can lead to improved flexibility and better customer orientation. Another hot topic is offshoring. Development projects are carried out in low-wage countries, mainly to realize cost advantages and to benefit from bigger job markets. In this paper we evaluate several case studies and experience reports dealing with offshoring done in an agile way. This provides a summary of common challenges and solutions in agile offshoring. We especially analyze which agile practices work well in typical offshore settings. Based on these results we conclude for which type of projects agile offshoring is applicable and where problems still exist.

Introduction

Offshore outsourcing

Offshore outsourcing (short: offshoring) plays an important role in today's software development practice. Offshoring stands for the relocation of business functions and -structures to third party companies in low-wage countries. Contrary to other variants of outsourcing the geographical aspect is more

important than contract issues. Chief motive for this relocation is mostly cost reduction by lower wage levels, but also increased flexibility, concentration on a company's core business and the employment of qualified personal that is not available in one's own country in sufficient quantities (Schaaf and Weber, 2005). Prime area of offshoring is the development of application software.

Offshoring is discussed in the literature and in the public mainly from an economic point of view. It has not yet been picked out as central theme of software engineering. An analysis from this point of view seems necessary as long-lasting effects on employees and technology are expected, as described by Bertrand Meyer (2006).

The geographical distance, different time zones, diverse cultural, social and political backgrounds, and other factors that are inherent to offshoring lead to special challenges in software engineering. These manifest themselves in shifted needs in the areas of requirements engineering, knowledge transfer, task sharing and cooperation and in quality assurance. Project management is also affected by distributed teams and a lacking insight into the project status, cumbersome arrangements and complex controlling.

All these issues make offshoring projects inherently more difficult to master than traditional projects. Matthew Simons (2004) expresses it this way: "Offshore distributed projects belong at the top of the complexity scale because they must overcome not only the challenges of distance and of multiple organizations but also of language, cultural, and temporal inhibitors."

These issues boost the importance of adequate process models with accompanying practices and artifacts.

Agile practices in offshoring

Agile process models and practices like extreme programming (Beck, 1999) promise to constitute a good basis for the challenges imposed by offshoring. Agile practices emphasize flexibility by avoiding upfront design, bulky requirements engineering, voluminous documentation and rigorous rules. Instead they focus on an involvement of the costumer, flexible adaption to changes and an iterative proceeding with extensive communication, author-critics cycles and permanent feedback on all levels. See (Martin, 2003) for a good introduction to agile practices.

Agile practices are used successfully in several of today's development projects. If they could be utilized in offshoring projects, the main advantages of agility (great flexibility) and offshoring (substantial cost savings) could be combined. This does not seem unproblematic as agile approaches place emphasis on communication and feedback and therefore call for a development team that is sitting together and collaborates closely with its customer. This is not easily achievable in a distributed setting.

Research goal

In this paper we present the results from analyzing several recently published case studies, experience reports and documents of other types of agile offshoring and related projects. The evaluation uses quite an arbitrary sample of unequal publications and is by no means complete or even final. But we hope that it helps in bringing forward the research in this area by investigating how agile practices are suited for the offshoring challenge, which special enhancements have been made for practical application and where need for further studies and conceptual work exists.

Related work

To our knowledge not many attempts have been undertaken to analyze published experiences of agile offshoring and similar types of projects. Braithwaite and Joyce (2005) have begun to create a catalog of patterns for the special case of distributed extreme programming and have identified five widely shared patterns: Kickoff, visits build trust, virtual shared location, remote pair and multiple communication modes.
Moore and Barnett (2004) as well as Simons (2004) have assembled collections of common challenges and solutions in agile distributed and offshoring projects. Kircher et al. (2001) establish the research field of distributed extreme programming and give an overview of agile practices and their applicability for a distributed setting of which much also applies to agile offshoring.

In the next sections we present the choice of evaluated publications, common challenges found in the application of agile practices and established as well as innovative solutions. Then we address relevant remaining problems and conclude for which type of projects agile offshoring is applicable in its current form.

Analyzed publications

The publications have been selected for this paper by scope and availability. There were not many published case studies and experience reports to choose from and even less research papers. The publications can be categorized by organization type and document type.

The following organization types have been identified:

- **Agile Offshoring (AO):** Agile practices are applied to offshoring projects.

- **Agile Global Software Development (AGSD):** This is a more general organization type. Agile practices are used in globally dispersed software development. This includes geographically distant teams of equal status and is not limited to an offshoring setting.
- **Distributed Extreme Programming (DEP):** This is a common organization type where extreme programming is used as core agile process model. An offshoring setting is not required. The term is defined in Kircher et al. (2001) as "Extreme Programming with certain relaxations on the requirements of close physical proximity of the team members".

Agile Offshoring is the main area of focus. We also had a look at Agile Global Software Development and Distributed Extreme Programming publications as these are related organization types and most experiences gained there are also applicable to Agile Offshoring. AGSD emphasizes the geographical aspect with cultural and ideological differences, DEP – as advancement of XP – the distribution of teams which affects communication and coordination and AO the relationship and communication between unequally ranked teams and remote customers.

The publications that have been analyzed can be grouped in

- **Case Studies:** Detailed analyses of real or study projects that stress factors contributing to their successes or failures.
- **Interview Studies:** Results from interviews with participants of real world projects.
- **Experience Reports:** Reports from real world projects with a description of the experiences that were made.
- **Research Papers:** Scientific publications that present research results.
- **White Papers:** Overviews over concepts and practices of concrete approaches of individual companies.

This distinction serves to give a quick overview and not as an exact placement as some publications contain topics of more than one group, e.g. case studies containing several interviews.

A full list of all considered publications with their corresponding classification can be found in the appendix.

Common challenges

The challenges of implementing agile practices in offshoring projects that were found by analyzing the publications are classified by the four items in the Agile

Manifesto (Beck, 2001) that has become an often cited short description of the core subjects of agility: *Individuals and interactions*, *working software*, *customer collaboration* and *responding to change*. The most often mentioned and the reportedly most problematic issues are given in bold font.

1. Individuals and interactions

This group deals with people acting in different roles, communicating with each other and coordinating their work.

Communication is the main issue in offshoring. It is good practice in many agile teams to sit together and arrange things with other team members and resolve questions as soon as they arise. This is not possible in offshoring where teams are geographically distributed. This limits the team members' possibilities to talk with each other. Face to face conversation between on-site and offshore teams is not possible. Another limiting factor may be the time difference due to different time zones that hinders synchronous communication even with the usage of telephones or video conferencing systems. Issues that could be dealt with nearly instantly when the team is sitting together or all team members can be reached via telephone take a lot longer to solve this way.

Cost, limited bandwidth and varying availability of telecommunication channels prohibit rich conversations, especially in offshoring countries with poor infrastructure. This affects the **spreading of informal news** and gossip that is already very hard when team members do not meet regularly. Many news are interchanged during coffee breaks, informal meetings or after-work activities. A lack of this kind of communication may lead to a **lower team spirit** with no single team identity. Teams do not trust as easily members of the other team that they have never seen nor talked to in person, especially when cultural differences become apparent. Many publications report specific cultural issues that can arise when actions, statements or beliefs that are prevalent in one culture appear strange, disrespectful of even hostile to others.

With everyday communication being so difficult, **coordination** between team members is severely affected. This leads to less flexible processes and more formal than verbal agreements. A team's **common knowledge** must be build up and maintained explicitly. The same is the case with a **shared vision**, that is a common set of values and principles regarding the process and the collaboration. In a distributed environment this becomes even more critical as it is much harder to gain due to a lack of communication and cultural, ideological, and technical differences.

So more tasks and responsibilities have to be constantly taken care of. Many publications describe a wider **separation of roles** in the team with clearly defined responsibilities and functions to cope with these issues.

2. Working software

Topics in this group include integration problems, testing, quality assurance and configuration issues.

The **inadequate infrastructure** has been a problem in many projects. Offshoring projects have increased demands in the infrastructure. Not only does the hardware and software have to be similar and equally configured for all teams. Offshoring projects also require a high bandwidth, reliable connecting network for the transfer of source code and other documents in a shared version control system and the streaming of audio, video and program data for audio and video conferences and other software for remote collaboration.

Problems with a shared version control system can handicap the agile practice of **collective code ownership** severely. It is a also influenced by difficulties of dividing the work into separate tasks that can be assigned to different teams and handled sufficiently independent. This division is important as the remote setting raises the probability of parallel work on the same parts of the code that may lead to merging problems or repeated implementations of the same functionality. If the team was sitting together, mutual perception would be higher and situations like these could be avoided easier. Another problem can be **integration headaches** because implementations that work for their own do not function correctly when integrated. The root cause is that it is hard to specify the semantics of the interfaces sufficiently clear that development can be carried out without frequent coordination. **Configuration issues** can also lead to problems when implementations work on one machine but show errors on others.

An observation in some projects are shortcomings in **design and architecture skills** of offshore developers. Especially the practice of simple design leads to problems. Other publications report that offshore developers are as skilled as on-site ones. If the former is the case in projects, it can be quite hard to **train the offshore developers**. The common agile practice of coaching weaker or less experiences programmers with pair programming cannot be applied so easily. It can also be costly and time-consuming to communicate design and architecture specifications between on-site and offshore teams.

Quality assurance can also cause trouble when the teams don't share the same quality and schedule consciousness and some parts of an application are less tested than others. The distribution of teams across time zones can slow down progress in testing. Results of tests of the customers and feedback cannot be addressed directly but has to be send offshore and handled there.

3. Customer collaboration

This group deals with requirements engineering and the involvement of customers.

It is an established agile practice to bring a whole team together to accomplish the work and to communicate and give feedback permanently. Relating to the customer, this is called **customer on-site** in extreme programming. This is nearly impossible in an offshoring setting as the customer is residing far from the offshore team and normally is not willing to send coworkers to the offshore location. Another problem is the **language issue**, especially for non-English speaking countries. This makes it difficult for the offshore team to do **requirements analysis** and to contact the customer frequently. It is therefore possible that the customer and the offshore development team develop divergent understandings of the project requirements.

Establishing credibility and a friendly atmosphere with the customer is difficult. Personal talks and informal discussions are not possible in many settings.

4. Responding to change

This issue is applied mainly to challenges in project management. We have a look at leadership, controlling and process establishment and division of work.

The distributed team can lead to problems in **progress estimation**. The project status is less visible and controllable. More teams have to be coordinated, often from a distance. Therefore offshoring projects are riskier and project management and controlling is more difficult. Normally these projects involve more management functions than conventional agile projects. This rises the costs and stretches the time schedule of projects.

One principle in agile projects states that the developers and not managers should **estimate efforts** for upcoming tasks. In offshoring projects, the developers often are not available in the planning meetings or on-site and offshore developers develop diverging opinions about time and effort of tasks.

The division of work between the team can also be difficult as dependencies between work items cannot be resolved easily in a distributed setting. Offshore developers are usually less experienced in the application domain at the beginning of projects. So a **knowledge transfer** has to be established to upgrade them and to continually involve them more in the project to save costs by shifting more and more tasks offshore. The training of offshore developers is affected by high turnover rates of some countries, especially in India. Experienced developers fluctuate between companies to gain higher wages. This leaves projects with the problem of knowledge loss and can cause team synergy issues.

Practical solutions

As seen in the last section quite a few challenges arise when offshoring projects. Some agile practices, i.e. customer on-site, pair programming and planning game for extreme programming, are affected. In this section we list some of the adaptations and extensions of agile practices and other measures that have been used successfully in some of the projects.

We use the same classification in *individuals and interactions*, *working software*, *customer collaboration* and *responding to change* as above. The proposed measures are further grouped in

- *[o]rganization*: special organizational processes, roles, conventions, activities or tasks,
- *[d]ocument/artifact*: special documents or artifacts,
- *[t]ool*: application of special tools or applications.

1. Individuals and interactions

- *[o]* Dual-shore development with on-site and offshore team that split the work between them.
- *[o]* Team members are exchanged between the locations for some weeks to get to know their coworkers and their working conditions and to develop a better understanding.
- *[o]* "Ambassadors" / "Rotating gurus" travel around as experts and try to resolve misunderstandings between the teams.
- *[o]* Arrange a shared kickoff-meeting an the on-site location with all team members, also the offshore ones to build trust and establish a common team spirit. The kickoff can be followed by a shared boot camp.
- *[o]* Team members in different time zones adjust their working hours to reach maximum overlap.
- *[o]* Standup meetings (scrum meetings) are hold every day with a video conference system. Other meetings are scheduled weekly and monthly.
- *[o]* Development starts with only a small team that grows over time.
- *[o]* New team members are introduced to everyone through a web conference. Encourage non-business communication to build a common team spirit.
- *[d]* Wikis are used as central team repository, e.g. for use cases, story cards, tasks, questions and answers, tracking data, calendar of events and team and personal blogs.
- *[d]* Use the source code as main communications medium between sites.
- *[t]* Telephone, video conference systems, instant messaging, e-mail, virtual whiteboards and other technical means are used to connect the teams. The basic infrastructure uses a high-speed data link.

- *[t]* Application sharing is used to work as remote pair. Special applications help in developing together and in giving instant feedback.

2. Working software

- *[o]* Experienced coaches supervise all teams.
- *[o]* Code written offshore is reviewed in a formal detailed design review.
- *[o]* QA is also carried out offshore to shorten the testing cycle and get rapid feedback.
- *[o]* High-level design is carried out on-site.
- *[o]* The offshore team must not do too long periods of independent development.
- *[d]* Uniform standards, checklists and examples are given to all teams.
- *[d]* Unit tests are written by the offshore team and checked for accordance with the requirements before implementation (test-driven development).
- *[d]* UI prototypes are developed on-site, discussed with the customers and provided to the offshore developers as guideline.
- *[d]* The application is build on a common framework.
- *[d]* The core architecture of the application is solidified by a small and experienced on-site team during the first iteration. This way offshore developers can start up on an established code base and a solid architecture.
- *[t]* A shared source control system with collective code ownership is used. Special tools with local proxies can help to deal with network problems. Collective code ownership can be constricted to special modules or features.
- *[t]* Cruise Control with automated tests or similar tools for build management and quality control are used.
- *[t]* The same testing environment (similar hardware and software) is used on-site and offshore. Testing should be automated as far as possible.

3. Customer collaboration

- *[o]* Domain experts from the offshore team cooperate with on-site business analysts and prepare the requirements for the offshore team. The analyst part of the offshore team grows as they understand the business better.
- *[o]* “Proxy customers” take the role of real customers. They should have sufficient knowledge in the application domain and should be able to interact equally well with technical and business project members.
- *[o]* Requirements remain constant during an iteration. Changes are deferred until the next iteration, if possible.

- *[t]* Communication with the customer is carried out through video conference systems.
- *[t]* Application sharing can be used to work jointly on requirements and tasks.

4. Responding to change

- *[o]* Choose people for the project not only by technical but also by soft skills. Make clear that an adjustment of habitual working practices and travels or temporal relocations may be required.
- *[o]* A common vision with simple and clear vocabulary should be created and marketed to the development team.
- *[o]* Take care that each other is trusted as equal team members. Is its very important to encourage communication.
- *[o]* Increase the time for iterations and make them more flexible.
- *[o]* Provide the offshore team with more authority and the responsibility of making decisions to give them a sense of liberation and empowerment.
- *[o]* Divide the work by functionality and subsystems into independent subsystems.
- *[o]* Major technical changes can only be introduced by a well established formal process.
- *[o]* A risk assessment meeting is hold every week with the main actors from both sides.
- *[o]* Frequent deliveries allow a better measurement of the project progress by the amount of working software and already implemented requirements.
- *[t]* Track bugs and issues through a system that is accessible from all teams. This gives an estimate of the project progress and quality.

Conclusion and outlook

The analysis has shown that offshoring indeed poses special difficulties for development projects. Agile process models and practices seem to be appropriate for use in these contexts but have to be enhanced and adapted to work well.

The main problem lies in the restricted communication between on-site and offshore teams and between the offshore developers and the customer. This problem is not specific to agile approaches. But it is especially important in this context as communication and regular feedback form the basis of agility.

The solutions that were realized in many of the considered projects lie in a substitution of direct communication by technical means like video conference systems, telephone calls and instant messaging supplemented by an exchange of members between the teams. This is costly and time-consuming to set-up and

generates significant expenses during the projects' duration, especially by rotating resources. This may be too much for small projects. Kussmaul et al. (2004) state: "Avoid projects that are too small to amortize the overhead required for a effective distributed team. Very small projects are best done by local teams, unless an offshore team already has direct expertise." Another factor is that it takes some time to put the necessary infrastructure and organization into place.

Does this mean that offshoring is only affordable for bigger projects or a series of projects? Or is there a way for small projects to let on-site and offshore teams collaborate and communicate in a less cost-intensive fashion?

We believe that established practices and tools of software engineering can be employed to stronger formalize and structure agile offshoring without loosing too much flexibility of agile practices and falling back to a document-driven approach. We have performed a case study with this focus and expect from the results that a combination of component-based development, architecture-centric evolution, test-driven development and other practices of software engineering can disburden offshoring projects and enable small teams to integrate offshoring parts in an agile way without much of the initial and running costs and overhead. We plan to conduct further studies and research in this area.

Appendix: List of evaluated publications

Braithwaite, K., Joyce, T. (2005). "XP Expanded: Distributed Extreme Programming", in Baumeister, H., Marchesi, M., Holcombe, M. (eds.) *Extreme Programming and Agile Processes in Software Engineering, 6th International Conference, XP 2005*, 18-23, Springer **(Distributed Extreme Programming – Research Paper)**

Computer Enterprises (2004). *Global Agile Development – CEI's Approach to Successful IT Outsourcing, no. 09132004*, http://www.ceiamerica.com/cei/company/resources/ceinformant_09132004.pdf **(Agile Global Software Development – White Paper)**

Danait, A. (2005). "Agile Offshore Techniques – A Case Study", in *Agile 2005 Conference* **(Agile Offshoring – Case Study)**

Fowler, M. (2004). *Using an Agile Software Process with Offshore Development*, http://www.martinfowler.com/articles/agileOffshore.html **(Agile Offshoring – Experience Report)**

Kircher, M., Jain, P., Corsaro, A., Levine, D. (2001) "Distributed eXtreme Programming", in *Second International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP2001), Cagliari, Sardinia, Italy* **(Distributed Extreme Programming – Research Paper)**

Kussmaul, C., Jack, R., Sponsler, B. (2004). "Outsourcing and Offshoring with Agility: A Case Study", in Zannier, C. et al. (eds.) *XP/Agile Universe 2004, LNCS 3132*, 147-154, Springer **(Agile Offshoring – Experience Report)**

Murphy, B., Snyder, T. (2003). *TwoShore – The Valtech Agile Global Delivery Model, Part 1 – for the Project / Program Director*, Valtech Technologies **(Agile Offshoring – White Paper)**

Paasivaara, M., Lassenius, C. (2004). "Using Iterative and Incremental Processes in Global Software Development", in *Proc. of the Int'l Workshop on Global Software Development, International Conference on Software Engineering (ICSE 2004), Edinburgh, Scotland*, 42-47 **(Agile Global Software Development – Interview Study)**

Poole, C. J. (2004). "Distributed Product Development Using Extreme Programming", in Eckstein, J., Baumeister, H. (eds) *XP2004, LNCS 3092*, 60-67, Springer **(Distributed Extreme Programming – Experience Report)**

Simons, M. (2002). *Internationally Agile*, http://www.informit.com/articles/article.asp?p=25929 **(Agile Offshoring – Experience Report)**

Yap, M. (2005). "Follow the Sun: Distributed Extreme Programming Development", *Agile 2005 Conference* **(Distributed Extreme Programming – Experience Report)**

Other references

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA

Beck, K. et al. (2001). *Manifesto for Agile Software Development*, http://www.agilemanifesto.org

Martin, R. C. (2003). *Agile Software Development. Principles, Patterns, and Practices*, Prentice Hall PTR, Upper Saddle River, NJ, USA

Meyer, B. (2006). "Offshore Development – The Unspoken Revolution in Software Engineering", *IEEE Computer*, vol. 39, no. 1, 121-124

Moore, S., Barnett, L. (2004). *Offshore Outsourcing and Agile Development*, Forrester Research Report

Schaaf, J., Weber, M. (2005). *Offshoring-Report 2005: Ready for Take-off*, Deutsche Bank Research, BITKOM

Simons, M. (2004). *Distributed Agile Development and the Death of Distance*, Cutter Consortium Executive Report, vol. 5, no. 4