HAL
archives-ouvertes.fr

# When global process fails: A grounded theory study of a case from agile engagement to compulsive outsourcing

Jan Pries-Heje, Magnus Hansen, Sofia Bergbäck Knudsen

## To cite this version:

Jan Pries-Heje, Magnus Hansen, Sofia Bergbäck Knudsen. When global process fails: A grounded theory study of a case from agile engagement to compulsive outsourcing. Joint IFIP TC 8 and TC 6 International Conferences on E-Government, E-Services and Global Processes (EGES) / Global Information Systems Processes (GISP), / Held as Part of World Computer Congress (WCC), Sep 2010, Brisbane, Australia. pp.245-258, 10.1007/978-3-642-15346-4_20 . hal-01054637

**HAL Id: hal-01054637**

**https://hal.inria.fr/hal-01054637**

Submitted on 7 Aug 2014

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

# When global process fails: A grounded theory study of a case from agile engagement to compulsive outsourcing

Jan Pries-Heje, Magnus Hansen & Sofia Bergbäck Knudsen

janph@ruc.dk, magnuha@ruc.dk, samb@ruc.dk
Roskilde University, Denmark

**Abstract.** In a Scandinavian company developing a healthcare information system (IS) at three Scandinavian sites they succeeded in taking agile processes into use across the three sites. After a fourth development site in India was added the use of agile development processes gradually came to an end and plan-driven processes took over. In this paper we report from a month-long study where our analysis of the case shows that the cause for giving up agile was three-fold: (1) The cultural distance between India and Scandinavia was too great. (2) There were telling differences in competence and (3) the presence of knowledge asymmetry. From this analysis we develop a grounded theory explaining the necessary preconditions for succeeding with a global process for agile IS development.

**Keywords:** Global Process, Agile Processes, Outsourcing, IS development

## 1 Introduction

Information systems (IS) development typically unfolds as a process with stages such as analysis, design, coding and testing. Stages do not have to be carried out sequentially but can be done in parallel or iteratively. An IS development method is a prescribed process for carrying out development. The method description typically includes activities to be performed, artifacts resulting from the activities, plus some principles for organizing the activities including roles and responsibilities.

In the mid 1990s current IS development processes were typically effective in large-scale, long-term development efforts that employed stable and disciplined processes [1]. In contrast, many IS projects involved rapid changes in requirements and unpredictable product complexity. IS development approaches that achieves a balance between flexibility and disciplined method were needed [2-4] – so *agile processes* were born.

A comparative study [5] of the older traditional 'plan-driven' processes and agile processes found that IS developers in the 21st century environment in many cases adopted a considerably different set of agile practices from those associated with traditional IS development. The distinctive agile principles (those with no traditional counterpart) regard teamwork and on-the-fly software process adaptation, and they emphasize informal knowledge exchange, collaboration, and experience as important

elements in IS development, and acknowledge more sensitivity to tailoring project practices to environmental conditions. These principles are essential for managing software projects in volatile settings where fast changing technologies and markets drive fast changing skills and knowledge.

Prominent examples of agile processes (methods) are XP (eXtreme Programming), Crystal and SCRUM. The major similarity between XP and SCRUM is the focus on close collaboration and shared experiences in order to achieve a successful developing cycle. However, this close collaboration with shared experiences is at risk when the SD team is expected to outsource part of the development process to cut costs. Interestingly enough, a recent quantitative study found that the use of agile practices like stand-up meetings in an outsourced development context where management located in the US and development in the Czech Republic was successful due to informal communication presented daily [6]. The question remains as to how agile processes fare against a team that has to share its knowledge with new developers working globally? Recent studies indicate a 'yes', when characteristics of agile and global are blended properly [7].

Outsourcing traditionally is the practice where an organization purchases goods or services that were previously provided internally [8]. Outsourcing involves the movement of specific tasks or entire processes to one or more outsourcing vendors, typically to a place where wages are lower [9]. Organizations have claimed that IS outsourcing reduces cost and time, increases quality and reliability of products and services, improves business performance, and releases organizations to concentrate on core competencies [9].

The traditional way of thinking outsourcing is a vendor/client-relationship, where the client organization to the greater part decides what and when the vendor organization should deliver and how the process should be carried out [10]. However, an increasing number of client organizations aim for a closer collaboration with the vendor organization. It causes IS development to take place in a *distributed environment* where a team of developers working on the same project are distributed at several locations, sometimes across larger geographical distances.

The distributed project team is heavily dependent on a developing method which can support communicating and coordinating their daily work tasks [11]. The challenge in this is to be able to share knowledge and experiences through computer mediated communication-channels, a collaboration method that has proven less successful than working in collocated projects [12]. A suggestion is to use practices such as synchronizing head milestones, frequent deliveries, use of peer-to-peer communication links, problem solving practices, information- and monitoring practices and client/vendor-relationship building practices [13, 14].

These distributed team best practices are much alike those practices argued for in agile processes. Moore and Barnett (2004) found that labor gains from outsourcing combined with agile development practices can lead to lower overall costs along with a project team that can address a high rate of change [15]. A McKinsey study even shows a saving of 58 cents is achieved for every dollar mainly on wages, and further concludes "that offshore services are identical to those they replace …" [9, 16].

Can agile processes be the answer in achieving a successful distributed IS developing team? If we turn to scholars like Cramton [17], Olsen [12] and Cockburn [18], agile processes in a distributed environment is dismissed because of the lack of

face-to-face-communication due to the large proximity between team members. However, drawing on conclusions from Kiesler & Cummings [19], we seek out to investigate the antecedents of distributed IS developing in an agile setting so the work is not constrained to the effects of lack of face-to-face communication only.

This leads to the following related research question: *Why is it difficult to get agile processes to work in a distributed environment?*

The remainder of the paper is structured as follows. First we present our research method and then our case and our case analysis. Third we discuss our findings with an emphasis on the changes that outsourcing and distributed IS development caused. Finally, we use our findings to develop a theory on what makes the use of agile processes difficult.

## 2 Research Method

To answer our research question we decided to use a single-case study. The main reason being that case studies is a useful way of exploring unknown phenomena based on "why" and "how" research questions [20].

The case study was conducted using semi-structured interviews and participatory observations of the team's stand-up meetings over a course of three months around New Year 2008-09. In the beginning we observed the ending of one project iteration – observing the last stand-up meetings in that iteration. In the end of our case study we observed the beginning of a new project iteration aiming for a new project goal.

The reason for choosing to observe stand-up meetings was because this was an important daily communicative event where most team members took part in the ongoing communication and coordination of the daily work. It was also a unique opportunity for us to observe how agile principles were implemented in a particular context based on globally distributed worksites. We characterized the use of agile principles by using a theoretical framework by Conboy & Fitzgerald [21] of how successful agile teams respond to a variety of types and sources of change [21]: Over time the cost of defining and fulfilling changes decreases as the learning process of the team as a whole increases. From this definition we were able to compare the team's reluctance and enthusiasm towards delegating different type of tasks with different change requests to the developers over time.

We mixed the data collection processes to obtain different insights as to how the participants contemplated themselves and others (during interviews) and how they positioned themselves and others in a social context (during observations) [22]. We interviewed team members who themselves volunteered to participate in the study. This resulted in 11 interviews and six observed stand-up meetings. Observations were transcribed and recorded. Afterwards we made very thoroughly written summaries of the interviews.

In analyzing and reducing the data we listed important passages and quotes from the interviews as well as the observations and transcribed these in further detail. Our method of coding the data were based on categories identified through a literature study of distributed teams and agile development in outsourcing. As we perceive IS development as a social act with for example formal and informal power plays we

used discourse analysis in order to understand how the participants in the software team individually made sense and applied meaning to concepts, words and work structures. In the concrete we used the concepts of inter-textual and inter-discursive references in both interviews and observations to interpret how the participants made and remade values, norms and meaning to their respective work practices [23].

After having analyzed the data we decided to apply some coding procedures from Grounded Theory (GT) to develop a theory. Barney Glaser and Anselm Strauss originally developed GT [24, 25] as an inherently flexible methodology in which the researcher "should simply code and analyze categories and properties with theoretical codes which will emerge and generate their complex theory of a complex world" [26]. Glaser also makes theory generation versus theory verification the core theme of GT. As an alternative to this Strauss' version of GT emphasizes things like replicability, validation, and verification as important parts of conducting GT research. Use of GT in IS research is exemplified by a landmark paper by Orlikowski [27] on CASE tools and organizational change, over explorations on software requirements [28, 29], to a more recent study from Denmark of how the Internet is redefining information systems development methodology [30].

In the research reported in this paper we use the GT school of thought developed by Strauss [31]. According to Straus and Corbin [31], analysis in a grounded theory approach is composed of three groups of coding procedures called open, axial and selective coding. However, we only used the last third of these coding procedures called selective coding.

Selective coding involves the integration of the categories that have been developed to form the initial theoretical framework. Firstly, in selective coding, a story line is either generated or made explicit. A story is simply a descriptive narrative about the central phenomenon of study and the story line is the conceptualization of this story (abstracting). When analyzed, the story line becomes the core category, which is related to all the categories found earlier (as explained above), validating these relationships, and elaborating the categories that need further refinement and development.

## 3 Scandinavian IT company case

To answer our research question we selected a larger Scandinavian IT company which offers IT consultancy as well as development of their own IT solutions within a broad range of areas. The company has 16,000 employees primarily in Europe, and has a long tradition of virtual collaboration across geographical distances. In the concrete we were able to obtain access to the company via a healthcare IT project.
The particular project team which we examine in this case is developing a module to an Electronic Patient Journal-system with the scope of documenting patient treatment and care in a Swedish hospital. The team consists of seven developers, two software testers, a systems architect and a project manager. They are one of two teams developing the system and are nearing their first delivery phase in the middle of December 2008 and are about to start a new phase when we leave the team in January 2009.

Like the company in general, this team consists of co-workers spread out over several worksites in Sweden, Denmark and Norway. The majority of the team's software developers work in Malmö, Sweden. The *director of development* is in charge of several projects throughout Scandinavia: He works from Copenhagen, Denmark. The *project manager* who works from Oslo, Norway, participates in the daily telephone meetings held by the project team and tries to visit the Malmö office once a month. The daily telephone meetings (called stand up-meetings) are managed by the *project coordinator*, who has a professional background as a nurse and also functions as the team's product quality tester and domain expert.

Phase 0: Very early in the life-cycle of the project a plan-driven [32] waterfall-like software development method was used. However, this method in combination with the distributed work environment having three Scandinavian sites generated a lot of severe problems for the project such as missed deadlines and a product with unacceptable (low) quality. Thus the project was in need of a dramatic change in both management and development method and a decision of trying out agile method was taken. This turned out to be a successful move.

Phase 1: The project team switched from a plan-driven software development process to communicating and coordinating their daily work and tasks according to agile processes, thus turning a failing course of action into a successful project. They were now delivering software on schedule and with acceptable quality.

Phase 2: Top management in the company of course took notice of the now successful project that was working across three different sites (Malmö, Oslo and Copenhagen). Based on the team's success the director of development decided to try and integrate offshore outsourcing by adding two Indian developers to the team employed by an Indian subsidiary company working from India. The rationale behind this decision was partly the success in doing work across different sites and partly a company strategy aiming for cost savings through offshore outsourcing (to India).

Thus when going from phase 1 to phase 2 the team started to experience problems. Our data collection and analysis will focus here.

## 4 Case analysis

### Story Phase 0: From plan-driven chaos to agile success

The project team's original work method of plan-driven development [32] in combination with distributed work environment posed a number of severe problems for the project team, which ultimately created an urgent need for change. The developers from the project team felt lost in the overall process. The work climate was stressful work and deadlines were chaotic and often overdue. The result was a system with a design and quality which wasn't acceptable to either the team or the customer. The project manager worked from another location than the other team members and had only met the whole team once. Project management concentrated on creating work breakdown structures, estimates and deadlines, which the project team then time after time failed to meet.

*"So we needed to deliver, she (the PM) throttled on and wanted estimates from everybody, estimates, estimates, estimates. And she promised to deliver before Christmas 2007."*
The result being a severe breakdown in trust towards the project manager's methods, competence and ability; something had to be done. Thus a new project manager was hired and suggested trying out agile processes.

**Story Phase 1: The use of agile processes in distributed collaboration is a success.**

The IT company in general as well as the specific project team was used to working virtually, collaborating over geographical distances, linking several worksites together. In spite of this, the project team decided to try to work with an agile development method in their daily work. The way of working chosen was inspired by eXtreme Programming - XP [33, 34] and SCRUM [35]. Even though this type of software development approaches usually prescribes close collaboration and face-to-face-communication as an essential part of the procedure, the project team succeeded in finding alternative ways of communicating and coordinating daily work.

*PM: "We have pretty good communication, meetings, mails and such. […] I am supported and collaborating closely with Britt. (the project coordinator)."*

One of the alternative ways of making sure that all team members take part in the development process is to engage in stand up-meetings. Due to the team's distributed work environment where many of the coworkers sometimes are elsewhere (at the customer's supporting or training the it-staff or end-users or simply working from home), they have to transform physical boundary objects [36, 37] used as help tools into digital form and also engage in virtual communication on the stand up-meetings using telephone conference calls with shared desktop. The boundary objects now used are a spreadsheet called a "Cookbook" that contains stories (short specification requirements), a spreadsheet called a "Bank" with finished stories, and the stand up-meetings which play the role of verbalizing and constituting specifications and task assignments.

In fact, the team experienced an increase of productivity and handling of changes after they introduced function points to their stories and as their use of agile processes increased. Overall the team releases better post-release quality shown by customer satisfaction and quicker handling of change requests, even though they often work distributed and have limited possibilities for meeting and working together at the same worksite. This higher productivity in the shift to agile methods coincides with findings from Layman et al. [6] and Fitzgerald et al. [38].

**Story Phase 2: Offshore outsourcing leads to breakdown – the agile method fails**

Being so used to working distributed, one would think that a single site more or less would not have any major impact on the project team's successful use of an agile method. However, this was not the case. After introducing developers into the team that were working for the Indian subsidiary company from their Indian office, the agile working process gradually broke down. Right after the decision was made to include offshore outsourcing it was known from company experience that socializing

into the team was important. Thus two Indian developers are sent on a 2-3 month introduction at the Swedish development site where the majority of the developers work in order to get to know the work environment, the work domain, the work processes, the various aspects of the system, and last but not least their Swedish colleagues in person.

The stand up-meetings works well while the Indian developers are at the Swedish site but when they return to the site in India after their introduction, the team experiences a significant change in productivity, participation and communication from their Indian counterparts.

*"Here we have a decline in our function point production. I don't know why, perhaps it's the introduction of the site in Pune?"*

The project coordinator tells us about some work tasks which are given to the Indian developers that turn out completely wrong according to what the Swedish site intended. However, she is aware of the difficulties that communication over great geographical distances may cause:

*"There are probably a thousand ways of interpreting these kinds of things when you're at the other end of the line. You don't get this ... ping-pong as we call it, when you ask questions and get an answer and then you ask new questions, and then you ask away until you understand how things are. You just don't get that."*

This is one of the major challenges in global software development in general and especially for teams working with agile processes, where the close collaboration is dependent on the frequent communication and face-to-face communication is recommended to achieve this. It appears in examining this case study's project team that the step from working distributed across worksites in Scandinavia to working across worksites with greater geographical (and cultural) distances is too large. The concept of communication in this regard is however of great importance. It is peculiar how the type of communication which the project coordinator describes coincides with that which a traditional plan-driven requirements specification method would prescribe.

The Indian developers also remain mostly silent at the daily meetings and are instead engaged in forced communication by the Swedish developers and the stand up-coordinator. This tips power and control in the Scandinavian team's favor and goes against the vision of collaborating closely as one team. We observed stand-up meetings and saw examples of the Indian co-workers getting work tasks appointed to them by the Swedish team members rather than themselves suggesting what they could work on during the workday. This more and more resembles a plan-driven way of communicating and goes against an agile vision of close collaboration as well as the idea that everybody in the team are peers at the same level [35]. Instead it makes it difficult for the Indian developers to understand their work tasks because the team creates a discourse of authority and leadership where questions asked result in more time spent. In continuation of this it becomes a barrier which prohibits the Indian developers in gaining further knowledge of the system and its context because the typical discourse is that the flow of communication and information moves from the Swedish sites to the Indian site. One of the Swedish developers is already aware of this, because it can not possibly be a fulfilling work situation for the Indian developers:

*"To sit that far away and not getting everything explained properly, that can't be fun. They are really nice and well-educated boys, so... I think they could get something else, with better working terms."*

There are also technical difficulties to overcome as the Indian developers do not use the boundary objects given at their own initiative and sometimes they are not granted access to test environments, databases and even shared screen at the meetings which the rest of the team members always can and do use. There may be a number of reasons for this happening. A Scandinavian developer, who recently had been posted at the Indian worksite for a couple of months, during this time also participating in the stand up-meeting from India, told us that it was difficult to hear the Scandinavian colleagues over the phone, and some of the Swedish co-workers were downright impossible to understand. However, these technical difficulties were not at any time brought up and to attention by the Indian co-workers: Most likely because of the discourse of authority which was being enacted at the meetings. The team members working from India might also have misunderstood or missed out on the information that the content of a certain stand up-meeting would require a shared screen in order to execute the meeting in an effective way. It might also be a combination of misunderstanding or underestimating the work task itself which leads to the Indian developers not being able to evaluate what type of boundary objects they need in order to be able to discuss work tasks at a stand-up meeting in a sufficient and comprehensive way. As a result the gap between Indian and Scandinavian co-workers' knowledge and experience with the systems and its domain grows even wider. This also contributes to the lack of knowledge from the Swedish team about the context of the Indian work site which the Swedish developers realize far too late in the development process.

The project coordinator thinks of the interaction and socialization which naturally takes place at a worksite as one of the factors which might put strains on the communication and coordinating across globally distributed worksites:

*"Just think of the amount of information we swop with each other here every day, which isn't... it's just sort of ad hoc. A comment, a question. They (Indian developers) get nothing of that."*

As a result of all this there is an unevenly distributed amount of information available for the two work-sites and has a negative, and sometimes even destructive, impact on communication and coordination within the team as a whole. The co-workers in Scandinavia are caught in a difficult work situation where they end up using a lot of time explaining and elaborating work tasks for their Indian co-workers, thus limiting the time they spend on their own work. In some situations they even end up choosing whether they should execute certain work tasks themselves rather than letting the Indian developers try to do it, knowing that the latter option will cost a lot more in terms of time and effort from both sides:

*"Interviewer: So it takes the same amount of time for Peter to explain how to solves this task?*

*Project coordinator: It takes longer time.Interviewer: Maybe even longer time?*

*Project coordinator: He (Peter) says, this would take me a quarter of an hour to solve, and he (Indian developer) had estimated 14 hours to do this task."*

There are several obvious problems in this: Skilled programmers using their time and effort working as communicators and teachers, trying to get co-workers to do

some work that they easily could have done themselves. This becomes problematic when resources are not distributed for this purpose. Further there may be potential conflict lying in the fact that it was top management that decided – with a business strategic rationale in mind - to add Indian offshore outsourcing resources to the project (going from phase 1 to phase 2); and maybe the developers themselves never accepted this kind of (changed) work situation and the asymmetry in power between the two groups was facilitated even before they began their collaboration.

There is also the problem of the Indian co-workers never getting a fair chance of learning and understanding the system they work on, as well as its context and domain. If they only get uncritical and sometimes boring tasks to carry out, they will never overcome the skills and knowledge gap between themselves and their Scandinavian colleagues. This renders them in an unfortunate dependency situation where they are dependent on the good will and help from the Scandinavian team members all the time to be able to perform their work, ultimately wasting a lot of time and resources which could have been used instead of sitting idle and wait. It is important to note that the dependency relationship between the work sites in the project team does not go both ways. The Scandinavian team members do not need their Indian colleagues to function in the same way as the Indian team members do. Needless to say, this is not a good platform to build an agile global software developing project team on.

## 5 Discussion

When we phrased our research question "why is it difficult to get agile processes to work in a distributed environment?", many readers may have thought: That is just because they are not sitting face-to-face! However, as we have explained above agile processes worked very well in a distributed environment with three Scandinavian sites and we aimed at expanding this theory to the antecedents of the distributed environment. Developers at these three sites were obviously not sitting face-to-face. Thus we carried on our analysis digging behind the face-to-face answers.

Based on the story of our analysis, three primary issues emerge that indicate the challenges agile teams must face: (1) the cultural distance after splitting the team is reinforced by the physical distance, in spite of the team meeting virtually through conference calls, (2) different levels of expertise, skills and experience become more obvious due to the limited communication channels available, (3) knowledge asymmetries between both teams where the Indian worksite depends on the communication, information and knowledge of the Scandinavian sites.

The three issues we found were all constituted and reinforced by the way the team communicated at the stand-up-meetings and also from the emerging deadline which drew nearer as time passed.

The list is by far exhaustive but merely represents the empirical findings of our actual case.

One of the reasons this happened was due to an uneven level of domain knowledge which was not properly constituted during the introduction period and as a result of uneven socialization throughout the process which was a result of the geographical

distance between the two major work sites. Drawing on power theory from Fairclough, we can characterize the underlying power that the Scandinavian team exerted over the Indian site as a result of this uneven socialization and domain knowledge which was reinforced through their daily interaction [39].

The findings regarding distance and salience of information leading to different interpretations are very similar to the findings of Cramton [17]. Although Cramton studied chats used in study groups with no significant 'developing process' we saw that many of the same problems arose in our study. After some time in the development process, the Scandinavian developers simply had to put their trust in the Indian developers' understanding of the information shared on the stand-up-meetings as the primary means of feedback was the actual software product that the Indian coworkers had to develop.

Our case is an example of distance vs. agile tools that really benefit the most from physical presence. The situation is that we now have available the means of communicating and coordinating technology. However, we have yet to provide an easy way (and even more important, a cheap way) of interacting physically over geographical distances. As a result of this, we saw a reinforcement of the specific work processes and enactments towards one another as time went by that became more or less destructive towards the agile vision of working together as one team.

So far we have mostly seen empirical findings from teams that have implemented agile processes on-the-fly pragmatically and with good reason. The problem of standardizing agile processes is of great importance in this paper, as the theory of agile methodologies often does not resemble the actual work in which they are practiced because of the need of implementing changes [7, 40].

Problems maintaining strict guidelines of agile processes also correlate to the conclusions of Agerfalk [41] where no true answer concerning the performance of agile processes is given. It is even suggested that the hitherto 'pragmatic' method of implementing the agile principles should be discarded completely in favor of adapting the agile processes in their entirety in order to benefit properly from them [41].

David L. Parnas (cited in Agerfalk [41]) simplifies the problem of software development at its core: the problems in software development are not new and the solutions are not agile processes. In our case we saw a classic scapegoat: a belief that it was the communication as a whole that was at fault, since this was obviously the problem and the solution of the development team in the first fiasco project. However, as we dug deeper, we saw that this was not only so. Communication was definitely a problem but not the final and foremost reason for the problems. Instead, communication can be seen as symptoms of other, more physically and deeply grounded problems; cultural distance, difference in experience and expertise of the developers and in the end resulting in an unevenly distributed amount of information resulting in dependency issues from and a change in power leading to a more plan-driven offshore outsourcing strategy.

Indeed, we believe that our study supports the point towards the *potential* benefits of agile offshore outsourcing. Just like the potential benefits of offshore outsourcing were financially grounded in the beginning of the 90's and still not realized to its full extent, the potential benefits of working globally agile are flexibility and speed. The problem is that uneven socialization combined with distance between work sites pose a major difficulty for agile processes because socialization is a prerequisite for

successful close collaboration. It is inevitable that each worksite develops and enacts their own social behavior and patterns, resulting in each worksite socializing more with each other internally than with their distributed colleagues.

We argue that this vision of close collaboration poses difficulties so strong that it is not economically feasible. Having two software teams with dedicated project management each would in our case have been more feasible and would most likely prohibit a lot of the 'communication scapegoating' we saw during our study. It could also have prohibited the power asymmetry which was present from the beginning.

Customer location is also a significant issue that must be taken care of through close collaboration and communication in agile methodologies but are very difficult to establish in a distributed environment. One of the primary problems of integrating agile processes in a globally distributed environment is the role and location of the actual customer and end-users and this is a factor which one must not underestimate although a distributed work environment a priori contains challenges of close customer collaboration in the first place.

This finally brings up another issue of the discussion regarding what kind of product is being developed concerning requirements regarding customer's needs and demands. Would an off-the-shelf product be more suited for agile distributed software development? "What are we developing" and "who is the customer" should therefore be trivial yet important questions to ask before, during and after engaging in global software development using agile processes.

## 6 Conclusion

Using the three categories of observations from the discussion we used grounded theory selective coding (as we explained in the research method section) to derive a theory. It says:

*The use of agile processes is made difficult by cultural distance, differences in competencies and knowledge asymmetry. Furthermore cultural distance and differences in competencies will in itself influence the development of knowledge asymmetry. If cultural distance, differences in competencies and knowledge asymmetry becomes too great it will make the use of agile processes impossible.*

Our theory is also shown in figure 1. We have *saturated* the theory [26] using all the data available; this means making sure that nothing in our data contradicts the theory (core story) and that no major observation is not explained in the theory.

Cultural distance was derived through the team participants' stories of different interpretations of meaning in their communication or in their lack there of. The cultural difficulties in this instance are also an instance of subcultures between the worksites in the team where both teams create different meanings on their own.

Differences in competencies were a given due to the original Scandinavian team's background of close collaborating and due to the introduction of new coworkers (the Indian worksite) in general. However, the differences became apparent when close physical collaboration was not possible.

Knowledge asymmetry and dependency of information and communication from one site to another, was the not-very-surprising result of cultural distance and

differences in competencies. Primarily the Indian site became dependent on the three other sites.

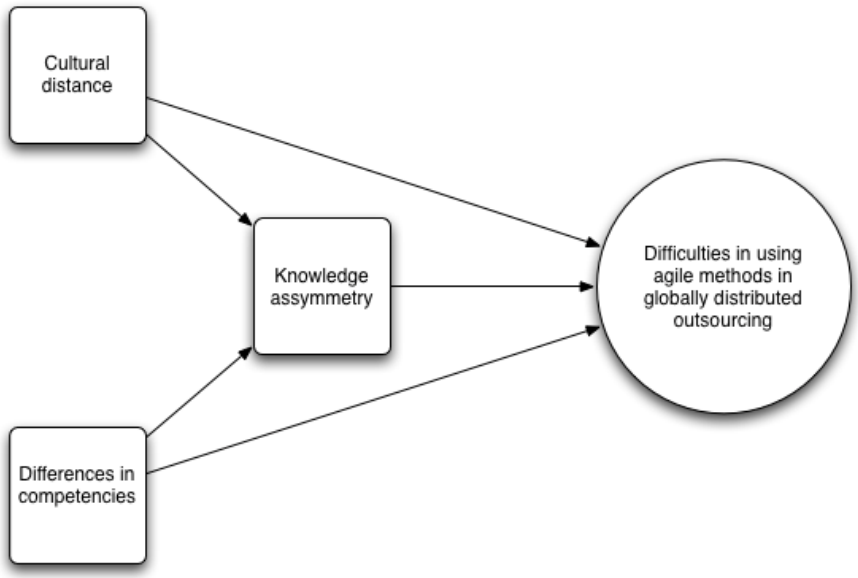


**Figure 1: The resulting theory on what makes use of agile processes difficult**

Of course our validation of the theory is based on a single case as well as the concept of saturation from GT. According to Yin [42] and Baskerville & Lee [43], generalizing to statements or theories is more valid through a multiple case study, because a single case study often will have difficulties in generalizing outside its specific, contextual setting. Nevertheless, we deviate from this normative standard of assuming that a quantifiable study result is the best way of creating new knowledge. After all, knowledge and meaning creation in qualitative, interpretative studies is very specific depending on the researcher and the individual as well as the group making up the surroundings of the research.

This means that given enough time and analysis, any case would theoretically deviate from one another and the question remains as to what extra value several cases actually bring? Instead, the theory we produce from this setting is a result of an analytical generalizability process derived not only from the single case but also coded using GT and afterwards compared to current literature through a literature study. The value we add is not that of purely quantifiable results but instead that of a transparent research process and also a proposal of a theory, which other research can benefit from using or testing.

Finally, one of the primary limitations as also discussed in the Discussion is that of practice versus theory. It is difficult to interpret actions and meanings of the use of agile processes because hardly any team proclaiming to use agile processes is actually using it in a rigid (prescribed) way. Thereby leaving behind doubts to whether a more ‘strict’ use of certain principles and tools - i.e. retrospectives - would have resulted in a different effect than what we found?

## References


1. Harter, D., M. Krishnan, and S. Slaughter, *Effects of process maturity on quality, cycle time, and effort in software product development.* Management Science, 2000. **46**(4): p. 451-466.
2. Cusumano, M.A. and D.B. Yoffie, *What Netscape learned from cross-platform software development.* Communications of the ACM, 1999. **42**(10): p. 72-78.
3. Cusumano, M.A. and D.B. Yoffie, *Software development on Internet time.* IEEE Computer, 1999. **32**(10): p. 60-69.
4. Iansiti, M. and A. MacCormack, *Developing Products on Internet Time.* Harvard Business Review, 1997(September-October): p. 108-117.
5. Pries-Heje, J., et al., *Advances in Information Systems Development : From Discipline and Predictability to Agility and Improvisation*, in *Advances in Information Systems research, Education and Practice: IFIP 20th World Computer Congress, TC 8, Information Systems, September 7-10, 2008, Milano, Italy*, D. Avison, et al., Editors. 2008, Springer: New York, NY, USA. p. 53-75.
6. Layman, L., et al., *Essential communication practices for Extreme Programming in a global software development team.* Information and software technology, 2006. **48**(9): p. 781-794.
7. Ramesh, B., et al., *Can distributed software development be agile?* Communications of the ACM, 2006. **49**(10): p. 6.
8. Lacity, M.C. and R.A. Hirschheim, *Information Systems Outsourcing; Myths, Metaphors, and Realities*. 1993: John Wiley & Sons, Inc. New York, NY, USA.
9. McFarlan, F.W. and B. DeLacey (2004) *Outsourcing IT: The Global Landscape in 2004, Harvard Business School*. **Volume**, DOI: 9-304-104
10. Dibbern, J., et al., *Information systems outsourcing: a survey and analysis of the literature* ACM SIGMIS Database, 2004. **35**(4): p. 6-102.
11. Sauer, J. *Agile practices in offshore outsourcing–an analysis of published experiences*. in *Proceedings of the 29th Information Systems Research Seminar in Scandinavia, Helsingborg, Denmark, dated: Aug-12-15-2006*. 2006.
12. Olson, J.S., et al., *The (unique) advantages of collocated work*, in *Distributed work*, P. Hinds and S. Kiesler, Editors. 2002, The MIT Press. p. 113-136.
13. Kussmaul, C., R. Jack, and B. Sponsler, *Outsourcing and offshoring with agility: A case study.* Extreme Programming and Agile Methods-XP/Agile Universe 2004, 2004: p. 147-154.
14. Paasivaara, M. and C. Lassenius, *Collaboration practices in global interorganizational software development projects.* Software Process Improvement and Practice, 2003. **8**(4): p. 183-199.
15. Moore, S. and L. Barnett, *Offshore Outsourcing And Agile Development*. 2004, Forrester Research.
16. Bloch, M. and S. Spang, *Reaping the benefits of business-process outsourcing*. 2003, McKinsey
17. Cramton, C.D., *The mutual knowledge problem and its consequences for dispersed collaboration.* Organization Science, 2001. **12**(3): p. 25.
18. Cockburn, A., *Agile software development: The cooperative game*. 2nd ed. 2002: Addison-Wesley.
19. Kiesler, S. and J. Cummings, *What do we know about proximity and distance in work groups? A legacy of research*, in *Distributed work*, P. Hinds and S. Kiesler, Editors. 2002, The MIT Press. p. 57-82.
20. Cockburn, A., *Learning from agile software development - part one.* Crosstalk - journal of Defence Software Engineering, 2002. **October**: p. 10-14.
21. Conboy, K. and B. Fitzgerald, *Toward a conceptual framework of agile methods.* Extreme Programming and Agile Methods - XP/Agile Universe 2004, 2004: p. 105-116.

22. Atkinson, P. and A. Coffey, *Making sense of qualitative data: Complementary research strategies*. pbk ed. 1996: Sage Publications, Inc.
23. Fairclough, N., *Peripheral vision: Discourse analysis in organization studies: The case for critical realism.* Organization Studies, 2005. **26**(6): p. 24.
24. Glaser, B. and A. Strauss, *Awareness of Dying.* 1965, Chicago, IL, USA: Aldine.
25. Glaser, B. and A. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. 1967, Chicago, IL, USA: T. Aldine.
26. Glaser, B., *Basics of grounded theory analysis*. 1992, Mill Valley, CA, USA: Sociology Press.
27. Orlikowski, W., *CASE Tools as Organizational Change: Investigating incremental and Radical Changes in Systems Development.* MIS Quarterly, 1993. **17**(3): p. 309-340.
28. Urquhart, C., *Exploring Analyst-Client Communication: Using Grounded Theory Techniques to Investigate Interaction in Informal Requirements Gathering*, in *Information Systems and Qualitative Research*, A.S. Lee, J. Liebenau, and J.I. DeGross, Editors. 1997, Chapman and Hall: London. p. 149-181.
29. Urquhart, C., *Strategies for conversation and systems analysis in requirements gathering: A qualitative view of analyst-client communication.* The Qualitative Report (On-line serial), 2000. **4**(1).
30. Baskerville, R. and J. Pries-Heje. *Racing the e-bomb: How the Internet is redefining information systems development methodology*. in *IFIP Working 8.2 Working Conference*. 2001. Boise, Idaho, USA.
31. Strauss, A. and J. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. 2nd ed. 1998, Beverly Hills, CA, USA: Sage Publications.
32. Boehm, B. and R. Turner, *Using Risk to Balance Agile and Plan-Driven Methods.* IEEE Computer, 2003. **36**(6): p. 57-66.
33. Beck, K. and M. Fowler, *Planning Extreme Programming*. 2000, New York, NY: Addison Wesley Longman.
34. Beck, K., et al., *Embracing change with extreme programming.* IEEE Computer, 1999. **32**(10): p. 70-77.
35. Rising, L. and N.S. Janoff, *The Scrum software development process for small teams.* IEEE Software, 2000. **17**(4): p. 26 -32.
36. Gasson, S., *A genealogical study of boundary-spanning IS design.* European Journal of Information Systems, 2006. **15**(1): p. 26.
37. Levina, N. and E. Vaast, *The emergence of boundary spanning competence in practice. Implications for the implementation and use of Information Systems.* MIS Quarterly, 2005. **29**(2): p. 335-363.
38. Fitzgerald, B., G. Hartnett, and K. Conboy, *Customising agile methods to software practices at Intel Shannon.* European Journal of Information Systems, 2006. **15**(2): p. 200-213.
39. Fairclough, N., *Language and Power*. 2.ed ed. 2001, Edinburgh Gate: Pearson Education Limited.
40. Vidgen, R., S. Madsen, and K. Kautz, *Mapping the Information System Development Process* in *IT Innovation for Adaptability and Competitiveness*. 2009, Springer Boston. p. 157-171.
41. Agerfalk, P. and B. Fitzgerald, *Flexible and distributed software processes: old petunias in new bowls?* Communications of the ACM, 2006. **49**(10): p. 7.
42. Yin, R.K., *Case study research: Design and methods*. 3rd ed. 2003, CA, USA: Sage Publications.
43. Baskerville, R. and A.S. Lee, *Generalizing Generalizability in Information Systems Research.* Information Systems Research, 2003. **14**(3): p. 221-243.