2012 IEEE Seventh International Conference on Global Software Engineering

From RUP to Scrum in Global Software Development: A Case Study

Ramon Noordeloos
Dept. of Computer Science
VU University Amsterdam
Amsterdam, The Netherlands
Email:ramon.noordeloos@gmail.com

Christina Manteli
Dept. of Computer Science
VU University Amsterdam
Amsterdam, The Netherlands
Email:c.manteli@vu.nl

Hans van Vliet
Dept. of Computer Science
VU University Amsterdam
Amsterdam, The Netherlands
Email:hans@cs.vu.nl

***Abstract*—In this paper we present the results of a case study at two offshore projects that recently adopted the agile way of working. We analyze their multi-site governance activities adopted and adjusted based on the Scrum methodology. Furthermore, we identify those changes that the Scrum adoption brought, in comparison with the previous governance structure of the Rational Unified Process (RUP). We find that a transition from RUP to Scrum brings a positive effect in requirements engineering, communication, cost management and cross-functionality of the distributed teams. We also observe a negative change with regard to the development pace and delivery time. Overall, we add to the body of knowledge in the field of distributed agile, with an additional field study where we describe and compare the migration from RUP to Scrum, and the implications of this transition.**

***Keywords*-agile methodology, Scrum, global software development, governance.**

I. Introduction

In Global Software Development (GSD), agile methods are gaining in popularity [1]. Agile methods can help improve communication and collaboration in offshore development, which often results in better business / IT alignment and responsiveness to business changes [2]. However, when using agile methods in GSD, the main principles of agile become more challenging: direct face-to-face communication occurs less often and the team is not collocated anymore. In offshore development, communication and collaboration is always more challenging [3]. Since agile relies more on good communication and face-to-face contact than other software development methodologies, the implications will be far higher as well.

Whether a global software company decides to adopt an agile way of working or more traditional methodologies, it must be clear how the multi-site activities are governed. In other words, in order to organize and manage their distributed software development activities, organizations must adopt a governance model [4]. The selection of certain governance activities can influence the communication and knowledge management activities between the distributed teams [5].

In this paper, we present the results of a case study at two offshore projects that recently adopted the agile way of working. The projects are developed in a consultancy firm located in the Netherlands, together with an offshore site in India, and the client is a finance firm. We use the multi-site governance model introduced by [5], to analyze their multi-site governance activities adopted and adjusted based on the Scrum methodology. Furthermore, we identify those changes that the Scrum adoption brought, in comparison with the previous governance structure of the Rational Unified Process (RUP). The aim of the study is to add to the body of knowledge in the field of distributed agile, with a complementary field study where we describe and compare the migration from RUP to Scrum, and the implications of this transition.

In section II we elaborate on previous work on multi-site software governance theory and on the distributed agile methodologies and the improvements that these approaches can bring to the global software development challenges. Sections III and IV describe the project overview and the research methodology followed in this case study. In section V, we present our results and analyze the changes that occurred in the transition from RUP to Scrum. Finally, we conclude in section VI with a summary of our main observations and suggestions for further research in the future.

II. Related work

In this section, we describe the background literature related to the way multi-site software activities can be organized and governed. We also discuss current research on the benefits and challenges that agile methodologies brought in the world of the traditional global software development.

A. Multi-site Governance Model

Governance is defined as those arrangements and practices that an organization puts in place to ensure that the activities are appropriately managed [6]. In fact, only recently attention has been paid to define governance in software development: what are the structural attributes of software development governance and what are the coordination mechanisms that this governance embraces. For example, Heeks et. al. [4] investigate different strategies in multi-site cooperation that can lead to either *synching*, that is a successful cooperation, or *sinking*, an unsuccessful cooperation.

978-0-7695-4787-9/12 $26.00 © 2012 IEEE
DOI 10.1109/ICGSE.2012.11

Additionally, previous research suggest a structural perspective of the various multi-site governance activities that a global software company must adopt [5]. According to this perspective, global software activities can be categorized based on three structural aspects; the *business strategy* that binds the relationship of the remote offices, the *team structure and composition* of the remote offices and the *task allocation* that is the way activities and responsibilities are distributed across locations [5].

In this paper, we use the above mentioned multi-site governance model in order to analyze the distributed software activities of the case study company. Such a model can provide us with a structured outline in order to classify the various multi-site activities and how these activities are governed. Later in this paper, we compare the current multi-site governance model, which is based on Scrum, with the previous multi-site governance model applied, which was based on the RUP methodology.

B. Distributed agile: benefits and challenges

The use of agile methodologies in global software development is an emerging trend. Usually, organizations turn either to offshore development or to agile development to deal with software engineering challenges like reducing development costs, improving quality, aligning business and IT. Agile aspects like intensive collaboration and low-weight documentation seem to make the two approaches diametrically opposed. Intensive collaboration is quite challenging when one part of the team is located at an offshore location. Distributed software development often requires more documentation and a strict plan. Yet, it seems organizations who successfully overcome the complexities of merging the two approaches can gain advantages from both.

The transition from the more traditional methodologies towards an agile way of working is a challenge that comes with both benefits as well as risks. Nevertheless, such a change must been seen as an "ally rather than an enemy" [7]. According to [7] some of the challenges that can accompany a transition from the traditional methodologies to an agile approach relate to the requirements engineering, communication, cost estimation and other process-related challenges. Global software development is usually characterized by engagements with different national and organizational cultures in various geographic locations and time zones. Therefore, GSD practitioners need to employ suitable context specific mechanisms to mitigate these problems [8].

An important part of global software activities that can be influenced by agility is the requirements related activities such as requirements elicitation, prioritization, design and communication [9]. Requirements engineering is a difficult task even in a co-located environment, but it becomes even more challenging when requirements must be managed across time, distance, cultural and language differences. A prominent issue for example that prohibits requirements elicitation in GSD is the distance between the client site and the development team, as well as the trust issues that might exist between the two parties [10]. Agility seems to help mitigate challenges in requirements engineering in global environments. For example, agile methodologies such as the XP programming, promote a more frequent interaction between the customer and the developer which ultimately brings the two parties closer to each other and "motivates" the distributed teams to improve communication volume and frequency between them [10]. Furthermore, frequent integration and testing also help to ensure that every team member has understood the requirements correctly [11]. This is especially helpful if the participants in a globally dispersed team are from different cultures and have not worked together before. Through frequent integration and testing, team members will get a lot of feedback and any misunderstandings become visible in an early stage of the project. This prevents problems to grow or accumulate [11]. Short iterations bring also transparency of the work progress to all participants. Offshore developers for example can get instant feedback on their work which helps to motivate them and build trust between the different participants in the project.

Agile methods emphasize communication and provide useful coordination and collaboration practices. Applying those practices in offshore development can help improve many issues related to distance [12]. Increased collaboration and communication between the dispersed team members can also reduce cultural differences between the participants [11]. Frequent and open communication between team members and the frequent releases to the customer builds trust and helps to better understand each other's culture. Within the field of knowledge management, intensive collaboration and communication can increase shared tacit interpersonal knowledge between the distributed team members and reduces the need of sharing explicit knowledge [3]. For example, it can improve the knowledge on the business domain of the customer at the offshore site and it lowers the need to make these requirements clear through extensive documentation.

Applying agile methods in offshore development brings also additional flexibility to the contractual relationship between the involved parties. In the traditional global software projects, due to the limited ability to control the activities of the distributed teams, projects often rely on fixed commitments and pre-defined requirements [1]. On the contrary, agile processes are more flexible and adaptive and they emphasize changes during the development process. The customer can apply changes during the development phase without big consequences related to contract negotiations [11]. The customer does not need to specify all the requirements beforehand since agile processes are adaptive and focus on creating value to the customer. Requirements can change over time and they can be derived from a constant

negotiation between the developer and the customer.

Finally, agile methodologies use short iteration, frequent builds and continuous integration. These practices bring challenges to configuration management and version control in software engineering in general [11]. This practice is even more challenging in global software development where part of the projects must be managed over different, distributed locations. When teams are dispersed, long distances and bad infrastructure can create extra impediments in communication and cooperation.

The status of combining agility and GSD is captured in two systematic reviews [13], [14]. Many of the case studies reported therein concern situations where one company is developing software for its own use, or a product they sell, and outsource part of the development to another party. That is, there is no third party client. Examples hereof are [15] and [16]. If there is an outside client, he is not part of the team, as in [17]. Another typical aspect is that many of the case studies concern situations where the different sites operate relatively independently, and their work is coordinated at regular intervals. In the case of Scrum, this means different Scrum teams at the different sites, and Scrum masters that coordinate the work through a Scrum of Scrums, a practice advocated by the Scrum Alliance. This is different from a fully distributed Scrum team, where members of one Scrum team are distributed across geographies. An example of the latter is [17]. The characteristics of our case study that, together, distinguish ours from others, are:

- An agile project involving an outside client as well as a professional software development organization,
- A fully distributed team that evolved from a traditional (RUP) way of working to agile (Scrum).

III. Project Overview

The case study was conducted in a software consultancy firm, located in the Netherlands. We focus on two development projects that the company was working on, for a financial firm. The names of the case study company and its client were not allowed to be disclosed and for this reason, we will refer to the consultancy firm as Company A and to the financial institute as Company B. The two projects were derived from an initial project where Company A used Scrum for the first time in an offshore context, with team members being located in the Netherlands, and in an offshore office in India.

In both projects two different independent applications are built, but there are also a lot of similarities between the projects. Because of this, team members do often switch from one to the other project if needed. Furthermore, at the onshore site the two projects are also located in the same room. For this reason we will not make a distinction between the two projects in this field study when analyzing the results.

What is interesting in this case study is that both projects used initially traditional software development methodologies. They first used the Rational Unified Process (RUP) [18] as their main development methodology. In 2010 a manager of the client company challenged his employees to come up with an approach to deliver twice as much value for the same amount of money. As a result, they decided to adopt Scrum as their new approach. Most team members who are now working on the two projects have experienced this transition from RUP to Scrum and thus they have experience working with both RUP and Scrum, in a global environment.

Note that the projects started off using RUP. So an architecture-centric process was followed for a while, and the basic architecture was in place when the transition to Scrum was made.

IV. Research Methodology

The research was conducted based on a qualitative data analysis approach. Qualitative research refers mainly to the investigation and analysis of personal experiences and behaviors, as well as organizational functions and social interactions [19]. In order to gather the required data for the analysis, we chose to perform semi-structured interviews. In semi-structured interviews, questions can be open-ended allowing a conversational manner, while at the same time, an interview protocol can still be followed [20].

We interviewed 13 team members from both projects in a period of one month through semi-structured interview sessions. Each session took about one hour. The questions were predefined but open ended. During the period of interviewing the prime investigator worked in the same room with the onshore Scrum team members and also participated in some typical Scrum practices like the daily Scrum meetings and the demo meetings. Through this we could also observe and experience their way of working during the case study. Table I provides an overview of the different interviewees with their role, current project, site and company.

Table I
Interviewee profiles

Role	Project	Company	Site
Software architect & developer	1	Company A	Netherlands
Software architect & developer	2	Company A	Netherlands
Developer	2	Company A	India
Developer	1	Company A	India
Project Leader	1 & 2	Company A	Netherlands
Project Manager	1	Company B	Netherlands
Scrum master & Business analyst	1	Company B	Netherlands
Scrum master & Test manager	2	Company B	Netherlands
Tester	1	Company B	Netherlands
Tester	2	Company A	Netherlands
Tester	1	Company A	India
Tester	2	Company A	India
Tester	1 & 2	Company A	India

In the following paragraphs, we describe the case study in detail, by focusing on how Company A works using Scrum in a global software environment, what kind of activities and

processes are performed and how they are organized among the remote teams that participate in the project. We do that using the multi-site governance model, introduced in section II.

A. Business Strategy

In this case study, there are three parties involved; the client (Company B), the onshore provider in the Netherlands (Company A), and the offshore party in India. According to the multi-site governance model, the multi-site business strategy should consider the contractual and legal relationships between the remote sites. In our case study, Company A has a captive center in India, which means that the two remote offices belong to the same firm. A captive center is a subsidiary that a company creates in a offshore location, either by acquiring an existing company or by building one [21].

There is also a legal barrier between the onshore and offshore location; according to the European Union rules and regulations, a person from outside the European Union is not allowed to access financial systems of a company within the EU. This has several implications on the multi-site activities; integration testing can not be performed by people in India and the two remote locations work in different technical environments which they have to keep synchronized on a daily basis.

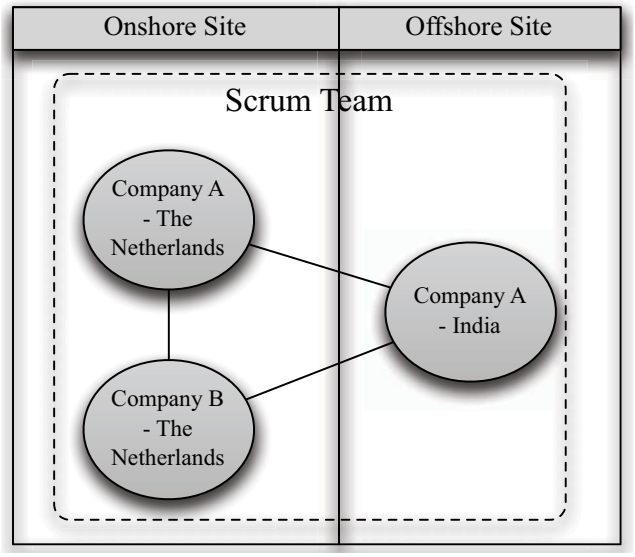
Additionally, the business strategy between Company A and Company B involves a flexible contractual relationship where change requests can be accepted without extra costs from the client part. This relationship is in line with the Scrum methodology which "dictates" shorter iterations and more changes can be considered during development. Finally, the client site is actively participating in the development process, by having members from Company B within the Scrum team.

B. Team Structure and Composition

In software development, team structure and composition is a critical factor for good performance [22]. Team size, role descriptions and role distribution are among those characteristics in distributed teams that can influence team coordination and communication and therefore team performance [5]. In this case study, we observe several aspects that characterize the way remote and co-located teams are organized and structured. More particularly, for each one of the two projects under investigation, there is one, unified Scrum team, with members from all three engaged parties. According to [13], this type of team structure is defined as a fully integrated Scrum team, where the team is cross-functional and the members are distributed across locations. Figure 1 illustrates the members that comprise the Scrum team.

The Scrum team is composed of different roles and responsibilities. In line with the standard Scrum methodology, there is a Scrum Master whose role is coaching the Scrum team and guiding it through the Scrum practices and rules. The Scrum Master is a person from Company A in the onshore site. Other members from this site include architects, system analysts and test coordinators. From the offshore side in India, there are mainly developers, testers and a team leader as part of the Scrum team. Additionally, based on the Scrum framework, there is also the role of the Product Owner, the person responsible for managing the requirements list (the product backlog). Here, the Product Owner is represented by a member from the client site (Company B). Other members from Company B include business analysts and requirements managers.

Figure 1. A Fully Integrated Scrum Team



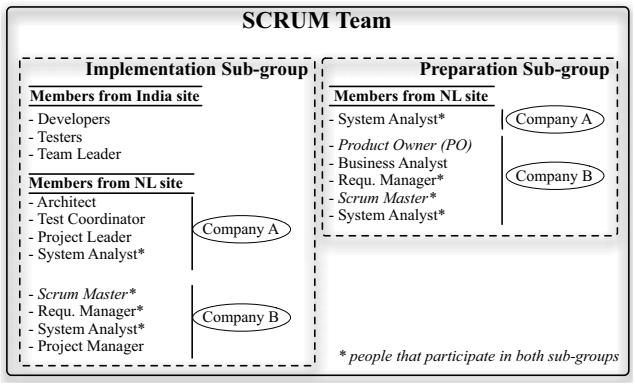
C. Task Allocation

Development is mainly done in the offshore site in India, where the developers and testers are positioned. Architecture and requirements management activities are the responsibility of the Scrum team members located in the Netherlands. Nevertheless, developers and testers from the offshore site actively participate in the requirements decisions, e.g. which requirements will be included in the upcoming Sprint, how to prioritize those requirements etc.

Within the Scrum team tasks and responsibilities are divided into two sub-groups. The first sub-group is responsible for implementing all changes and requirements in the product backlog, and make sure that everything goes according to plan. Part of this sub-group are the developers, the testers and the architects from the onshore and offshore site of Company A, and the system analyst and requirements manager from Company B. The second sub-group of the Scrum team is responsible for preparing the product backlog,

deciding what changes are to be implemented and make sure that everything is arranged and agreed upon, before the next Sprint begins. Members of this sub-group include the Product Owner and the business analyst from the client site, as well as a system analyst from Company A. Figure 2 illustrates the two sub-groups of the Scrum team, based on their preparation and implementation tasks.

Figure 2. Sub-groups of the Scrum Team



Furthermore, according to [13], Scrum practices need to be extended or modified in order to support global software development teams. In this case study, Company A has extended the standard Scrum framework, to fit the company's needs. More particularly, they use an additional phase at the end of the Scrum life cycle, during which all activities are summarized, documentation is completed and all the responsible team members ensure that the backlog is completed and the product is ready to be delivered.

Finally, cross-functionality is highly promoted within the teams. Since Scrum prescribes that every team member should be crossfunctional this is also applied in the two projects. Although all team members still have their primary role, they are now also involved into other tasks like elaborating stakeholder requests.

V. The transition from RUP to Scrum

As mentioned in section III, the team members of the current Scrum projects were also participating in older projects where RUP was applied. Therefore, they have experience working in a software global project with both a more traditional methodology (RUP) as well as with an agile methodology (Scrum). In this section, we present the changes that were observed in the transition from RUP to Scrum and the implications that such a transition brought to the projects' performance. In other words, we compare the current multi-site governance structure of the projects, which is based on the Scrum methodology, with the previous multi-site governance structure based on the Rational Unified Process. Overall, we observe that Scrum improved or eliminated the challenges that the distributed teams were facing when they were working with the more traditional development methodology. One of the onshore interviewees said:

> *"I am still working on this project because we now use Scrum which works a way better and is a lot more fun to do. If we would have continued working in the traditional way I would have already gone to another project."*

Change 1: Requirements engineering & Customer involvement. One of the major changes of the transition from the RUP to Scrum concerns the process of requirements engineering. During the traditional way of working, requirements were discussed in the onshore site, between people from Company A and the participants from Company B. After deciding on which requirements and which changes were to be implemented they were "throwing the requirements over the fence", that is they were communicating them to the team in India. As a consequence, the developers and the testers in the offshore site were not participating in the process of requirements elicitation and prioritization. With the current Scrum approach this aspect has changed. All team members, including the developers and testers from India, participate in requirements management, and they can all share ideas and solutions.

The most prominent consequence of such a change is that the team members in India can now gain more domain knowledge. They have the chance to develop personal skills and not only "act as robots". Additionally, the team feeling has increased, as the people in India feel more close to the their colleagues in the onshore site, as well as to the client site, and their motivation has increased. A developer from the team in the offshore site told us:

> *"In the RUP model, we worked more isolated and I often felt like I was a robot since I was just developing what I got through documentation, without knowing what it was meant for. Today, we are interacting with the client and I can now also increase my knowledge on their domain which makes the work more interesting."*

Furthermore, while following the RUP, there were three separate teams participating in the project; the onshore site of the consultancy firm, their offshore captive center in India, and the client site. As a result, requirements were first discussed between the onshore people of Company A and Company B, and then the members of Company A were communicating the requirements to the team in India. In other words, the people in the onshore site were acting as the intermediate between the client and the developers. That also meant that whenever the development team in India had questions about the implementation of certain requirements or changes,

they first had to ask their onshore colleagues, a process that can cause more delays and misunderstandings. An onshore interviewee from the client site said:

> *"In the past a lot of parties were involved in communicating things to India. This adds a lot of noise and delay in the communication process. Now, we can directly talk to India team members and explain them certain things face-to-face. This really improves progress of the project and the quality of the delivered work."*

In the current governance, with the use of Scrum, there is only one, unified Scrum team with members from all three parties involved in the development. That means that the client site is also actively participating in the development process, and they are closely connected with the offshore site. Consequently, requirements are better communicated and understood by all Scrum members. People in India understand better what needs to be implemented and together with their colleagues from the onshore site, they can prioritize the requirements more efficiently. In addition, the members from India can directly communicate with the client site, and as a result communication lines are shorter and less misunderstandings occur. According to an offshore developer:

> *"Because of agile, we now directly communicate with the client. This helps us understand better what they want and this results in a better product. It is also more interesting to work with the business people (Company B) and hear what they find about the product."*

Change 2: Communication.

Another change that is observed in the transition from RUP methodology to Scrum practices, is the frequency in the meetings held between the team members and consequently the changes in the frequency of communication. Following the Scrum project life-cycle, the team members meet daily in the stand-up meetings. They have adjusted the time frame of those meetings, from 15 minutes that is prescribed in the standard Scrum practice, to 25 minutes, in order to accommodate the proximity between the Scrum members and the involved delays. An onshore interviewee from the client site said:

> *"It is easier to handle business when people are working together. I always said that people should work more together and see each other on a daily basis. Now, we are finally doing this through Scrum practices."*

With the daily meetings and the more frequent feedback that the team members can get and receive, misunderstandings between them have been eliminated and issues are now resolved faster. According to an onshore interviewee:

> *"Today misunderstandings and impediments are identified much earlier than in RUP. Now we can work on something wrong for maximum one day. In the past, we could be working on a particular task for weeks before any misunderstandings or impediments were identified."*

Additionally, it was observed that through the daily interactions of the Scrum team, the members can synchronize better their activities between the onshore and the offshore locations. It becomes easier to keep track of each others' progress and status, since the frequent communication allows them to get a daily insight on which part of the project their colleagues work on.

Finally, another change with regard to the communication is the increase of the face-to-face interactions, through their video-conferencing tool. Company A uses a commercial tool called the Eye-Catcher[1] for video-conferences. The EyeCatcher is located in separate rooms where team members can talk to each other. In RUP, only certain people were communicating with each other using video-conference. The rest were using emails and instant messaging tools for their communication needs. Now with the daily meetings, where all Scrum members participate, they all participate in the video-conference and they all can see each other face-to-face. This increases trust between the remote and local colleagues and as a result people feel more motivated to collaborate with each other.

> *"During the Daily Scrum meeting we can see each other through the video-conference tool, which makes it more easy to detect if someone from the offshore site is holding information back. During the meeting it is also easy to break the ice through little jokes for instance. As a result, the offshore colleagues are more comfortable to speak."*

The client site, which in the past had no or very little direct face-to-face contact with the offshore site, now they also appreciate more their offshore colleagues and they show more respect to them since they are participating in the same Scrum practices and see each other daily. An onshore interviewee of the consultancy company said about this:

> *"In the past there was no face-to-face contact with India. It often happened that someone from the offshore site had some questions about a particular subject and asked this through email to a team member from the client site. As a result, the team member from the client site came to me saying: 'That person from India with the strange*

[1] http://www.qconferencing.eu/product/eye-catcher/

name is asking me strange questions again...'. This reflects how the client site was seeing the India site. Today this has totally changed since India and the client are fully involved and see each other on a daily basis."

Change 3: Pricing model and Internal costs.

A change that came with the introduction of Scrum practices concerns the pricing model between Company A and the client (Company B). Because of the agile way of working and the Scrum project life-cycle, there are smaller and shorter iterations between deliveries. Consequently, the members of Company A are more flexible in accepting and incorporating changes that members from Company B request. This arrangement benefits the client site because they don't need to pay extra for every change they request or for overwork. According to an interviewee from the client site:

"Since we have different agreements with the consultancy company I experience that the consultancy company colleagues are more open for changes from the client. In the past we had to negotiate about the price for every additional feature which was not part of the initial agreement."

Transition from RUP to Scrum also benefited the internal costs of Company A for communication purposes. In RUP, they were experiencing a lot of communication impediments, which resulted in more internal expenses in order to facilitate communication and collaboration between the remote offices. As a result, the internal costs of Company A were more than the benefits gained from outsourcing to a lower cost country. With the adoption of the Scrum methodology, and the improvement in communication between remote team members, Company A experiences less internal costs and now they can fully profit from the outsourcing benefits.

Change 4: Cross-functionality.

Another improvement that was introduced with the change from RUP to Scrum is the cross-functionality of the team members. Team members can be involved in more than one task and the moment they finish working on one task, they can immediately start working on another.

"During RUP it was quite common that the India site had nothing to do since they had to wait for tasks from the Netherlands, but it also happened that they worked the whole weekend to get a certain task done. Since the introduction of Scrum the offshore site can easily pick up other tasks from the Scrum board. Also, no overworking hours are any longer reported."

With the traditional way of working, the offshore members did not have the "freedom" to get involved in more tasks, and they had to wait for instructions from their colleagues at the onshore site. This process was causing delays. Now with Scrum, people from the India site can always work on something and therefore their feeling of responsibility and motivation is improved. An offshore interviewee said about this:

"In the RUP model we worked more isolated and I often felt I was like a robot since I was just developing what I got through documentation without knowing where it was meant for. Today we are interacting with the business and I can now also increase my knowledge on their domain which makes the work more interesting to do. The Daily Scrum meetings also helps to keep you motivated"

It should be noted that some people want to stick to their own role, for various reasons. For some, it has to do with building up a resume. For others, it has to do with difficulties handing over work. One has to pay explicit attention to this aspect.

Change 5: Delivery time and Development pace

During the analysis of the case study, we have observed certain changes that were introduced with the migration from RUP to Scrum which had a negative impact for the projects' performance. The first change relates to the delivery time. In RUP, delivery was every 6 months while now with Scrum delivery is every 2 months. Because things go faster with the Scrum way of working, there is not enough time for proper documentation. This has a negative effect later in the development phase, because the team members "miss the tracing lines". In other words, if something goes wrong people find it hard to go back and trace the issue because decisions are not well documented.

Finally, another disadvantage that was introduced with the transition from RUP to Scrum relates to the development pace. Because processes and activities are running now faster than before, the client site finds it hard to follow the pace with which changes are implemented and deliveries are ready. This is because within Company B there are certain departments involved in various aspects of the product such as risk management or quality assurance. Before the software can be released, or before a new feature is introduced, the Scrum team needs approval from these departments. According to their SLA's this process can take up to 10 days for instance, but during Scrum, people need approval the same day in order to implement requirements and changes instantly and create more business value. Consequently the client site creates delays and bottlenecks.

The changes we observed map well onto the challenges of distributed agile projects, as observed in [1]:

- **Communication need versus communication impedance.** Communication is an obvious challenge

Figure 3. The transition from RUP to SCRUM

	RUP (Before)	SCRUM (After)	Multi-site Governance Model
Change 1: Customer Involvement & Requirements Engineering	The offshore site is involved in the development and testing phases only. They were receiving the requirements and they had to implement them.	The offshore team is involved from the beginning in requirements management. • They have the chance to develop personal skills and not only "act as robots". • Team feeling has increased.	***Task Allocation***
	The onshore team was acting as an intermediate betweem the offshore team and the client site, for requirements communication.	There is one unified Scrum team with members from all three engaged parties. There is a direct contact between the client and the offshore team. • Offshore team understands better the requirements • Shorter lines of communication, less delays.	***Team Structure & Composition***
Change 2: Communication	Feedback was not so often. Someone might have been working on a wrong task for weeks or months before they realize.	With the daily meetings they can get daily feeback: • Less misunderstandings. Worst case is that someone is working on a wrong task for a day. Issues are identified immediately. • They can sychronize the work between the remote sites better. • They can keep track of each others progress and get an insight on which part of the project their remote colleagues are working on.	***Task Allocation***
	Only certain people were communicating with each other using the video-conference tool. All other members we using only emails and/or the chat tool.	Now everyone communicates with each other daily through the eye-catcher. Therefore, all Scrum members know each other, trust is increased, and they are more motivated to collaborate as a team.	***Team Structure & Composition***
Change 3: Pricing Model & Internal Costs	There was a fixed-price contract between Company A and Company B: • It was harder to accept changes • Company B had to pay extra for changes.	In Scrum there are small and short iterations: • Company A is more flexible in accepting changes. • Company B benefits, because they don't need to pay overwork (more cost efficient).	***Business Strategy***
	Company A was experiencing a lot of communication impediments which resulted in more intenal expenses in order to facilitate communication and knowledge transfer. Therefore the costs were more than the gains from outsourcing in a lower cost country.	With the adoption of Scrum, and the improvement in communication, Company A experiencse less internal costs, which is the initial driver of outsourcing, and that means that now they can "fully profit from the outsourcing benefits"	
Change 4: Cross-Functionality	In RUP, the offshore site should wait until the onshore team assigns them a new task to do. This caused delays.	The Scrum team is cross-functional. Once the offshore members finish their current task, they can immediately start working in another. • The feel of responsibility grows, motivation is improved and consequently performance is improved. (The members in India are more motivated to ask more questions, than before.)	***Task Allocation***
Change 5: Development pace & Delivery time	With RUP delivery was every 6 months.	With Scrum now they deliver every 2 months. Things "go faster", processes and development are running much faster than before. • Disadvantage: The client part (Company B) find it difficult to "keep up with the pace", especially because for them Risk Management and security are important issues. As a result communication bottlenecks occur. • Disadvantage: Less documenation is produced: they miss "trace lines" which means that if something is missing or something goes wrong, it is hard to trace it back because it is not documented.	***Task Allocation***

in distributed development, and a vital part of agile processes. Change 2 above shows that the transition to agile actually improved communication in our case.

- **Fixed requirements versus evolving requirements.** Due to the limited ability to control activities in distributed projects, global projects often resort to fixed requirements, while agile projects emphasize change. In the projects discussed, requirements could change in between sprints, as is normal in Scrum projects. The active participation of the offshore site in the requirements discussion improved their understanding of and commitment to the requirements (change 1).
- **People-oriented versus process-oriented control** and **Formal agreements versus informal agreements.** In global development, control is often achieved through formal processes, while control is achieved informally in agile processes. The lower control that resulted from the transition to agile necessitated a different pricing model, and actuially reduced overall cost (change 3).
- **Short iterations versus distance complexity.** Agile projects uses short iterations, which brings challenges to configuration management, version control and doc-

umentation. This becomes even more difficult to manage over different sites. This increase complexity was indeed observed in the current projects (change 5).

- **Team cohesion versus team dispersion.** Team cohesion is easier to realize when the team is collocated, and agile processes emphasize the importance of this team aspect. As noted in change 1, team feelings actually improved through the move from RUP to Scrum.

VI. Conclusion

The aim of this study was to investigate the transition from a traditional development methodology, to an agile way of working in global software development. We used the multi-site governance model to classify the different global activities based on three aspects; the business strategy, the team structure and the task allocation. Overall, we observed five different changes in the governance structure and the resulted implications. Figure 3 presents the results of our case study.

All changes, except one, had a positive impact on project performance. Scrum methodology introduced a fully integrated Scrum team, with members from all engaged parties instead of separate teams which shortened communication lines among the members and improved requirements communication. Additionally, the daily standup meetings improved performance, because the team members are now able to identify issues sooner and resolve misunderstandings. The cross-functionality of the teams creates a feel of responsibility and improves motivation of the offshore colleagues. Furthermore, cost benefits were observed as a result of the more flexible way of working introduced by Scrum with shorter and smaller iterations. Finally, there seems to be a negative impact from this transition due to the faster development life-cycle of Scrum methodology which also implies limited documentation. Despite the latter negative effect, we can conclude that overall, agile practices can have a positive impact on GSD projects and can help to mitigate many of the well-known challenges of GSD.

References

[1] B. Ramesh, L. Cao, K. Mohan, and P. Xu, "Can distributed software development be agile?" *Communications of the ACM*, vol. 49, no. 10, pp. 41–46, 2006.

[2] S. Moore and L. Barnett, "Offshore Outsourcing and Agile Development," Forrester, Tech. Rep., 2004.

[3] P. Ågerfalk and B. Fitzgerald, "Flexible and Distributed Software Processes: Old Petunias in new Bowls?" *Communications of the ACM*, vol. 49, no. 10, pp. 26–34, 2006.

[4] R. Heeks, S. Krishna, B. Nicholson, and S. Sahay, "Synching or sinking: Global software outsourcing relationships," *IEEE Software*, vol. 18, pp. 54–60, 2001.

[5] C. Manteli, B. van den Hooff, A. Tang, and H. van Vliet, "The impact of multi-site software governance on knowledge management," in *Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on*, 2011, pp. 40 –49.

[6] P. L. Bannerman, "Software development governance: A meta-management perspective," in *Proceedings of the 2009 ICSE Workshop on Software Development Governance*, ser. SDG '09. IEEE Computer Society, 2009, pp. 3–8.

[7] B. Boehm and R. Turner, "Management challenges to implementing agile processes in traditional development organizations," *IEEE Softw.*, vol. 22, pp. 30–39, September 2005.

[8] E. Hossain, P. Bannerman, and D. Jeffery, "Scrum practices in global software development: A research framework," in *Product-Focused Software Process Improvement*, ser. Lecture Notes in Computer Science, D. Caivano, M. Oivo, M. Baldassarre, and G. Visaggio, Eds. Springer Berlin / Heidelberg, 2011, vol. 6759, pp. 88–102.

[9] L. Layman, L. Williams, D. Damian, and H. Bures, "Essential communication practices for extreme programming in a global software development team," *Information and Software Technology*, vol. 48, no. 9, pp. 781 – 794, 2006.

[10] D. E. Damian and D. Zowghi, "Re challenges in multi-site software development organisations," *Requirements Engineering*, vol. 8, pp. 149–160, 2003.

[11] M. Paasivaara and C. Lassenius, "Could Global Software Development Benefit from Agile Methods?" in *Proceedings International Conference on Global Software Engineering (ICGSE'06)*. IEEE Computer Society, 2006, pp. 109–113.

[12] H. Holmström, B. Fitzgerald, P. Ågerfalk, and E. Conchúir, "Agile Practices Reduce Distance in Global Software Development," *Information Systems Management*, vol. 23, no. 3, pp. 7–18, 2006.

[13] E. Hossain, M. Ali Babar, and H.-Y. Paik, "Using Scrum in Global Software Development: A Systematic Literature Review," in *Proceedings 4th International Conference on Global Software Engineering*. IEEE Computer Society, 2009, pp. 175–184.

[14] S. Jalali and C. Wohlin, "Global software engineering and agile processes: a systematic review," *Journal of Software Maintenance and Evolution: Research and Practice, in press*, 2012.

[15] M. Vax and S. Michaud, " Distributed Agile: Growing a Practice Together," in *Proceedings Agile 2008 Conference*. IEEE, 2008, pp. 310–314.

[16] T. Niinimʹaki, " Face-to-face, Email and Instant Messaging in Distributed Agile Software Development Project," in *2011 Sixth IEEE International Conference on Global Software Engineering Workshops*. IEEE, 2011, pp. 78–84.

[17] M. Paasivaara, S. Durasiewicz, and C. Lassenius, "Distributed Agile Development: Using Scrum in a Large Project," in *Proceedings 2008 IEEE International Conference on Global Software Engineering*. IEEE, 2008, pp. 87–95.

[18] P. Kruchten, *The Rational Unified Process: An Introduction, Third Edition*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[19] A. L. Strauss and J. M. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Sage Publications, Inc., 1998.

[20] R. K. Yin, *Case Study Research: Design and Methods*. Sage Publications, Inc, 2003.

[21] R. Prikladnicki, J. L. N. Audy, D. Damian, and T. C. de Oliveira, "Distributed software development: Practices and challenges in different business strategies of offshoring and onshoring," *Global Software Engineering, International Conference on*, vol. 0, pp. 262–274, 2007.

[22] H.-L. Yang and J.-H. Tang, "Team structure and team performance in is development: a social network perspective," *Information & Management*, vol. 41, no. 3, pp. 335 – 349, 2004.