Agility Across Time and Space

Implementing Agile Methods in Global Software Projects

Bearbeitet von
Darja #mite, Nils Brede Moe, Pär J. Ågerfalk

schnell und portofrei erhältlich bei

beck-shop.de
DIE FACHBUCHHANDLUNG

Chapter 16
Considering Subcontractors in Distributed Scrum Teams

**Jakub Rudzki, Imed Hammouda,
Tuomas Mikkola, Karri Mustonen,
and Tarja Systä**

Abstract In this chapter we present our experiences with working with subcontractors in distributed Scrum teams. The context of our experiences is a medium size software service provider company. We present the way the subcontractors are selected and how Scrum practices can be used in real-life projects. We discuss team arrangements and tools used in distributed development teams highlighting aspects that are important when working with subcontractors. We also present an illustrative example where different phases of a project working with subcontractors are described. The example also provides practical tips on work in such projects. Finally, we present a summary of our data that was collected from Scrum and non-Scrum projects implemented over a few years. This chapter should provide a practical point of view on working with subcontractors in Scrum teams for those who are considering such cooperation.

16.1 Introduction

In this chapter we discuss industrial experiences in organising distributed Scrum teams that include members from subcontracting organisations. We present specific

J. Rudzki (✉) · T. Mikkola · K. Mustonen
Solita Oy, Satakunnankatu 18 A, 33210 Tampere, Finland
e-mail: jakub.rudzki@solita.fi

T. Mikkola
e-mail: tuomas.mikkola@solita.fi

K. Mustonen
e-mail: karri.mustonen@solita.fi

I. Hammouda · T. Systä
Tampere University of Technology, Korkeakoulunkatu 1, 33101 Tampere, Finland

I. Hammouda
e-mail: imed.hammouda@tut.fi

T. Systä
e-mail: tarja.systa@tut.fi

context of working with subcontractors in such teams. We discuss details of our experiences in subcontracting [1] and in Scrum projects [2] that have been reported in our previous publications. However, in this chapter we focus on the practical aspects of cooperating with subcontractors in distributed Scrum teams.

The practices that we discuss should be particularly helpful for companies that are planning to use subcontractors in their distributed Scrum teams. However, those who already have such teams, can use all or selected recommendations in the later parts of this chapter. First we present, in Sect. 16.2, the process of selecting subcontracting partners we used. Next in Sect. 16.3, we discuss agile practices and tools that have been used in our distributed and subcontracted project teams. Those practices and tools are rather generic agile practices, but their specific aspects, for example quick feedback, are particularly important in the context of distributed teams with subcontractors. In Sect. 16.4 we go through a life-cycle of an example distributed project phases where we present how the agile practices and tools are used when working with subcontractors. Finally, in Sect. 16.5, we conclude this chapter with a summary of findings and possible future work. However, before going into details we should present the company (Sect. 16.1.1), which is the context for this work, and present our methodology (Sect. 16.1.2) and main results (Sect. 16.1.3) as an executive summary for the readers who are not interested in the actual details.

16.1.1 Company Context

As the experiences that we discuss have been gained in a context of a specific company, we will first present this context. Our experiences have been obtained through a few years of cooperation with subcontractors at Solita.[1] Solita is a Finnish software service provider (SSP) that specialises in providing high quality software services in various domains. Since 1996 Solita has offered its services to customers in different domains, ranging from media, telecommunication, to public sector institutions and others. The diversity of customers and provided solutions demand very close cooperation with end customers, which requires a special approach to team organisation and choice of processes used. Additionally, the company has been changing internally over the years and has grown into a medium size company of 150+ specialists with two offices in two cities in Finland. These details give some perspective to the experiences we report on.

16.1.2 Methodology

Our findings on subcontractors in Scrum teams are based on our experience and the study we have conducted to find out the differences between Scrum and non-Scrum

---

[1] www.solita.fi.

projects. The summary of that study has been already published [2]. To obtain the data we prepared five questions following the Goal Question Metric (GQM) [3] methodology. We found this method to be most suitable for finding out how Scrum teams performed comparing to other teams. In this chapter we focus only on the study parts that are most relevant for the subcontractors involvement in Scrum projects.

Our GQM questions were as follows:

- *Q1: Do the current agile practices benefit projects?* We used three metrics Customer Satisfaction, Profitability, and Team Performance to answer this question.
- *Q2: Does the customer's direct involvement in the project benefit project success?* We distinguished three levels of customer involvement (none, partial and full involvement) and grouped project results based on this metric.
- *Q3: Is the communication different in agile projects?* We used two metrics the Communication Factor (time spend by the whole team for communication tasks) and Project Manager time (time spent for management tasks).
- *Q4: How to adapt agile practices to suit commercial needs?* This question was answered based on interviews with project leaders of all the investigated projects.
- *Q5: What kinds of projects suit agile practices?* This question was based on the project results and interviews with project leaders.

The data was collected based on interviews with Solita's project leaders and data was also obtained from IT systems that track project work time and financial data. We would like to clearly indicate that the data we collected was gathered with as much care as it was possible at the time, but it is not a perfect collection. Our study is limited to only one specific company. Therefore, our study is placed in the context of a medium size software service provider. Cases dealing with larger distributed teams can be found in literature, for example in a report on distributed teams developing Windows Vista [4].

Additionally, the collected data has also limitations regarding its quality, which might have affected metrics used for measuring customer satisfaction and team performance in particular. Both of the metrics were based on the subjective judgement of the project managers. Despite those limitations the data used as indicative reference point may be useful for other researches or practitioners who seek some references from similar contexts.

16.1.3 Main Results

All of the analysed projects amounted to 18 projects realised between 2006 and 2008. From those projects 8 projects were Scrum projects, which are most relevant for this study. We present more detailed data in the Appendix, while in this section results mostly relevant to subcontractors in Scrum teams are briefly discussed.

Findings to the question *Q1: Do the current agile practices benefit projects?* showed that Scrum projects performed in general better than traditional projects.

The number of subcontractors did not seem to be a factors impacting projects success in the analysed cases.

The next question that was relevant to subcontractors was *Q3: Is the communication different in agile projects?* An answer to this question showed that both Communication Factor and PM time were usually higher in projects with one subcontractor in a team. That can be explained by a need of additional formal communication with a person who is on a remote side. In general we found that communication factor and PM time increased in Scrum projects, but the increase was not very high (within a few percent points) comparing to traditional projects.

The question *Q4: How to adapt agile practices to suit commercial needs?* is the most relevant to work with subcontractors. The answers included observations and suggestions about the practises used in projects. Those suggestions and observations are presented in details in Sect. 16.3 and illustrated in the example presented in Sect. 16.4.

Finally, in the question *Q5: What kinds of projects suit agile practices?*, which was based on data analysis and interviews, we did not find any absolute limits for usage of Scrum and subcontractors in various project types. Scrum projects performed well even in cases where the teams included subcontractors and were distributed across more than two locations. Most of the Scrum projects were implementation projects where Scrum and subcontractors were used during development phase.

Generally, our findings presented in this chapter indicate better performance in Scrum-driven projects comparing to more traditional methodologies. In the context of work with subcontractors the quick feedback and organised communication seem to be the success factors that help project performance. We also discuss a subcontractor selection process where among other aspects the compatibility in terms of methodologies used and culture are criteria taken into account during the selection. Finally, we provide examples of tools that can be used in distributed projects to provide easy access and communication channels between team members from different organisations.

16.2 Subcontractors in an SSP Company

Having defined the company profile in which we operated we can discuss the processes we used for selecting subcontractors. The selection processes are important from the cooperation with subcontractors point of view that we describe as an example project walkthrough in Sect. 16.4.

The selection of a suitable subcontracting partner is especially important in the case of close cooperation between the development team that includes (or consists of) subcontractors and the end customer. In Solita we used a subcontractor selection process that has been built to suit our specific needs. The process was based on our experiences and existing best practices, which for example include a study on the quality in virtual teams and culture aspects of such teams [5], which was based on sourcing model eSCM-SP [6, 7]. We have already reported details of this process [1], and therefore in this section we focus on the elements that are most relevant to the creation of distributed teams with subcontractors.

16.2.1 Why Subcontractors?

There are many reasons for using subcontractors, the main ones include access to specialists, flexibility in team creation without the need to temporarily extend own staff, and potential differences in costs. Usage of specialists from other software organisations, namely subcontractors, requires careful selection of those partners as well as good team organisation. Subcontracting in a software service provider (SSP) company differs from that of product-orientated companies. The main difference is the direct interaction of subcontractors with the end customer, which usually does not take place in the case of product development.

In the case of SSP, the company serves its customers by delivering customised solutions specific to customer's needs. These solutions may differ in technology and nature of the solution, which may be a software implementation, but it can also be specialised consulting service provided to the customer in the area of the expertise of the SSP. However, in any case a team developing a solution for a customer works very closely with the customer. In that setting the team members, including the subcontractors, must be suitable for such a work environment. This is one of the reasons why Solita carefully selects subcontracting partners.

16.2.2 Distributed Development Stakeholders

Teams that work in the context of SSP companies generally involve a few stakeholders whose roles we should discuss in order to understand their roles in the distributed development teams. We can list three main stakeholders involved directly in development of specific software solutions:

- *Software Service Provider* is the party that orchestrates the whole development process of a solution. SSP is responsible for contacts with other parties (i.e. customers and possible subcontractors). SSP is responsible for delivering a solution that the *Customer* expects. SSP plays the central role in the case of customised solution development.
- *Subcontractor* is the other specialised software organisation who provides their experts to SSP's teams. The subcontractor takes the responsibility for their staff but does not have to be directly involved in specific solution development as a whole. Only selected experts from *Subcontractor*'s team are directly involved in development.
- *Customer* is the ordering party who expects a specific solution to help its business. The *Customer* may be involved in the solution development to a different degree, depending on the needs and expectations. In the case of teams consisting of subcontractors the customer is usually not interested in the details of the team organisation, but the customer is very interested in the results of the team work.

These three stakeholders interact during the solution development process. Frequently, all of these stakeholders are distributed, which contributes to the complexity

of cooperation. Furthermore, in some cases the stakeholders' roles may be mixed, for example, the original SSP organisation may be hired by another SSP. In that case the original SSP is a subcontractor to the hiring SSP, while the original SSP still has its own subcontractors. This may make the organisational complexity difficult to grasp, but the basic roles can still be recognised and the processes and tools specific to given role are applicable.
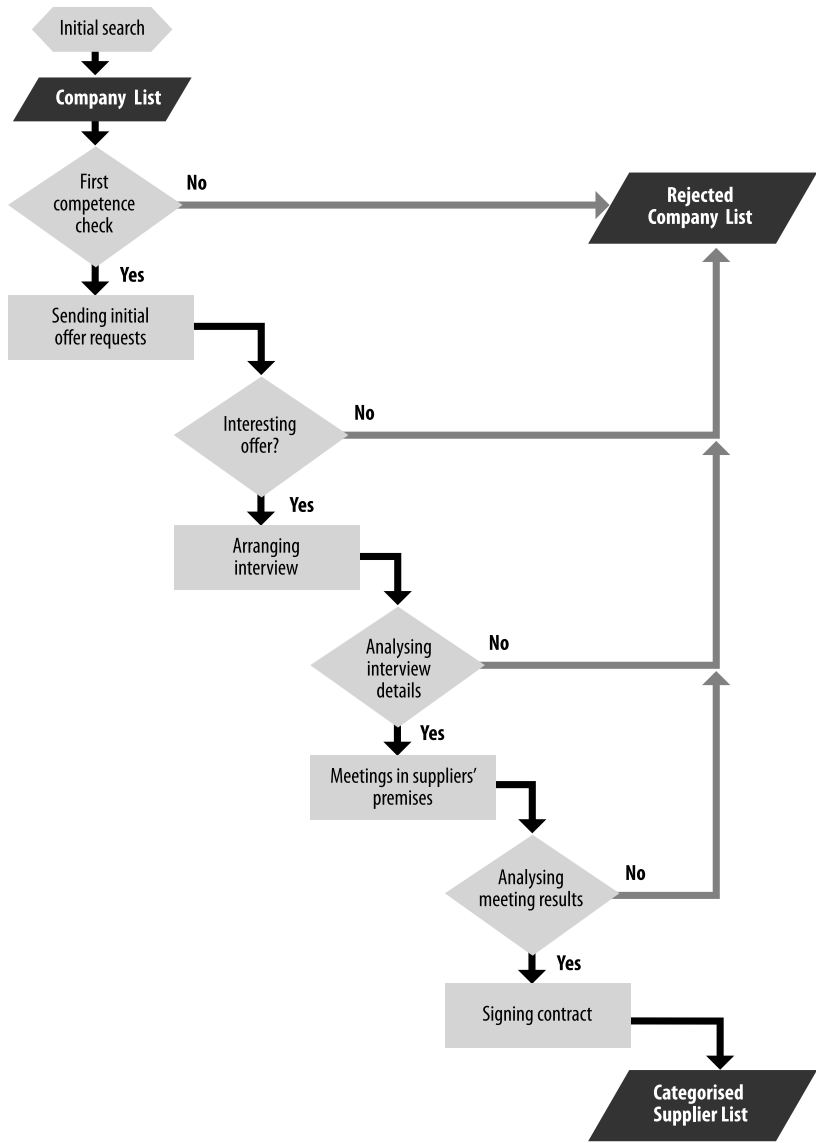
16.2.3 Subcontractor Selection Process

The subcontractor selection process used at Solita consists of a few stages. The process' steps are depicted in Fig. 16.1. First, we performed an *initial search* of all possible candidate companies. Already at this stage the search was limited by location and technology. In our case the location was limited to a few European countries that from the Finnish perspective can be regarded as nearshore locations. This location limitation was imposed in order to have a possibility of arranging face-to-face meetings quickly, and additionally to reduce timezone problems in the case of every day communication.

The next step was *sending initial offer requests* to companies that were selected from the results of the initial search. Then the replies were analysed for basic company details and possible pricing provided by the potential subcontracting partner. Then *arranging interview* could start. As preparation for the interview, the potential subcontractor companies were sent a questionnaire with questions about the candidate's organisation. If such a company has experience with agile techniques, or at least some of them, it is likely that cooperation with the company at that level will be more straightforward than in the case of companies unfamiliar with such techniques. Additionally, our questions were related to typical projects done at the candidate company. We asked, for example, about project size, typical length of projects, roles in the projects and customer types. Answers to those questions provided information about the candidate company and its suitability to work in SSP teams. We conducted the actual interviews with candidate companies as teleconferences. We also discussed in more details the answers to our questionnaire at that stage giving us clarifications to answers that might not be clear after reading the questionnaire answers.

After conducting the interviews we selected a few companies that seemed to be the most suitable candidates and we arranged an on-site *meeting in the candidate's premises*. One of the reasons for meeting in the candidate office was to see the environment in which the people work there. For example, the Internet connection quality, open or closed office space, and equipment can provide some indication of company infrastructure and the culture, which both are relevant in the case of cooperation with SSP's projects. The visits also gave us a chance to talk directly to specialists from potential subcontracting partners.

Finally, after conducting the interviews and analysing the data the subcontracting partners could be chosen. The partners selected over a certain period of time are

Fig. 16.1 Subcontractor selection phase [1, p. 227]

likely to differ in terms of their technological expertise, but they are likely to have similar cultures. Further cooperation will reveal possible differences and areas for improvements. We present an example of cooperation with subcontracting partners in a project in Sect. 16.4.

16.3 Subcontractors in Scrum Teams

Before going into details of software development in distributed subcontracted
Scrum teams, we discuss different agile practices and tools used in projects. We
focus on Scrum methodology, but we also discuss other agile supporting practices
used in Scrum projects. These practices and tools can be used together or selec-
tively depending on needs. However, certain practices should always be followed
in the case of agile projects. There are many practices and tools that could be dis-
cussed, however, we are focusing on the ones that have been used at Solita. The
practices and tools we discuss have been gathered based on the literature reports,
experiences of our customers, and our own observations over years [2]. A good
overview of agile practices in a distributed team provides, for example, an article by
Martin Fowler [8].

16.3.1 Scrum

The most prominent agile practice used by us is *Scrum* methodology [9]. Scrum has
been researched rather extensively in recent years, including the initial methodol-
ogy description by Schwaber [9] as well as later applications in distributed teams
reported, for example, by Sutherland et al. [10] or by Paasivaara et al. [11]. This
methodology consists of a few elements that practically help to organise projects.
One of the most important aspects of Scrum is a quick feedback loop, which bene-
fits projects in cases where corrective actions must be taken. A team performs work
according to a feature list defined in a *product backlog*. The features reflect cer-
tain functionalities that the end customer wants to have in the final product. Please
note that in this case a product does not have to refer to a real marketable product,
but rather a tailored software solution, which helps the customer to achieve certain
business goals.

   The features are implemented in an order determined by the priorities set by a
product owner, who is responsible for setting the priorities according to the business
value of particular features. The *product owner* selects features that should be im-
plemented in a coming *sprint*. A *sprint* is a period of time dedicated to development
of selected features.

   An overview of Scrum activities in a project is presented in Fig. 16.2. For each
sprint there is always a planning session when the implementation scope for the
sprint is decided. Then status meetings are held on daily basis. At the end of a
sprint the team demonstrates the implemented features. Also then the past *sprint* is
analysed as a retrospective and all team members have an opportunity to comment
on the work done. Retrospectives are important in distributed teams as they encour-
age open communication. Finally, either the development continues in following
sprints, or it is finalised with a delivery.
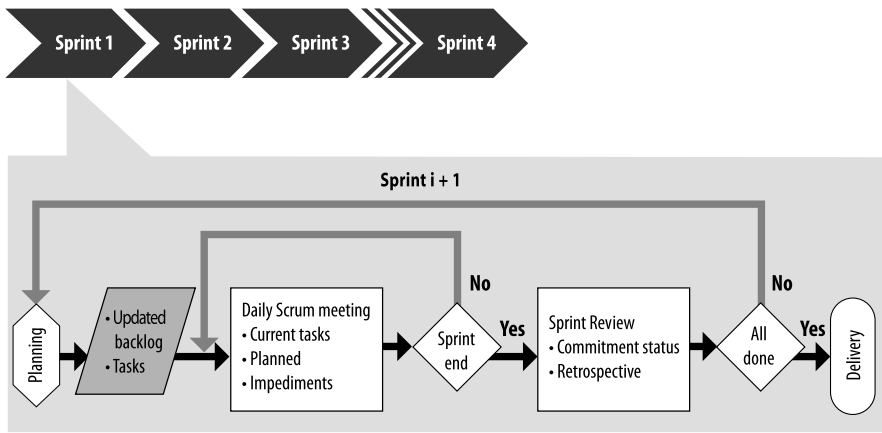
Fig. 16.2 Scrum activities overview

Practical Tip: Our experiences show the usefulness of Scrum in distributed teams with subcontractors. Scrum practices are especially useful in such a context as they provide important feedback loops. One is a daily feedback provided in the whole team on the progress of the development, so that the state of a project should be well known to the whole team regardless of their location. The other feedback is provided at the end of the sprint when the sprint has been analysed. This quick feedback gives a Scrum team the opportunity to take corrective actions before the project goes too far off the planned track. Finally, Scrum renders subcontractors as equally involved in the team activities as the team members from the SSP organisation.

16.3.2 Communication

Communication is extremely important especially in the case of distributed teams, which cannot benefit from direct informal communication as occurs in the case of co-located teams. Scrum defines a few meeting types (i.e. daily meetings, planning and demonstration sessions). Naturally, in addition to these meetings other meetings are often arranged, e.g., special workshop sessions on specific topics, discussions with customers, etc. In all theses cases, the practical arrangements of the meetings can be done in different ways.

Face-to-face meetings are most optimal for the efficiency and quality of communication, but such meetings frequently held may not be feasible in the case of distributed teams or customers located in remote locations. In these cases telecommunication tools can be used as a replacement for face-to-face meetings. Of course, a phone teleconference is a natural choice, but it can be replaced by Voice-Over-IP

solutions, which are typically more cost efficient. In addition to voice communication, videoconferencing tools can be used. There are many such solutions, which often offer very good quality. Videoconferencing can be a really good replacement for face-to-face meetings, but its use is limited by the infrastructure of the participating parties. Therefore, we find videoconferencing to be working best between selected locations (e.g., company offices), where the necessary equipment has already been installed. In addition to high-end hardware-based videoconferencing solutions, different software clients can be used. Such clients usually also support other functions, in addition to audio and video. A very useful feature during teleconferences is a shared screen where presentation, or other material can be shared.

Formal meetings that are the official channel of communication, even in the case of frequent Scrum meetings, may not be sufficient. That is why we use additional tools that at least try to enable less formal communication within distributed teams. Instant messaging tools, which often are the same tools as the ones used for voice or video communication, can be a good substitute of less formal communication. Usage of IM in projects is not a new concept and usage of IM in distributed teams has been reported in literature, for example by Herbsleb et al. [12]. In Solita we use Skype[2] as the primarily teleconferencing and instant messaging tool and LifeSize[3] for videoconferencing.

> *Practical Tip:* A development team can use a common chat where all the team members can easily exchange options, notify about changes, ask questions, and see availability of other team members. A chat can be used more willingly by subcontractors from other locations than phone, as it is less disruptive and more cost-effective. Team chat rooms have proved to be quite useful in real life projects.

16.3.3 Planning and Progress Tracking

The planning of a *sprint* is done in a meeting of the whole team. The *product backlog* features are moved to a *sprint backlog*, divided into tasks and re-estimated. The team can do the estimation using planning poker or other techniques, which for example have been described by Mike Cohn [13]. From our experience planning poker works well in distributed teams as it involves the participation of the whole team, which encourages also active participation of subcontractors. Additionally, in this way of estimating it is easy to notice any differences in opinions concerning the difficulty of a task. When estimates from individual team members differ significantly additional

---

[2]http://www.skype.com.

[3]http://www.lifesize.com.

analysis and discussion is needed, as the difference indicates a lack of common understanding.

When the planning is done the tasks are recorded in an issue/task tracking system. In our case JIRA[4] is used. Issue tracker, like JIRA, allows for following statuses of individual tasks. Finally, JIRA is a web application, which makes easy for all team members to use it regardless of their location.

> *Practical Tip:* Planning poker ensures that also team members from subcontracting organisation take part in the estimates. Additionally, the formula of planning poker, which resembles a card game, relaxes the atmosphere in the team.
>
> In the case of issue status tracking, if the team members do not update regularly the task statuses, it can be done in the daily scrum meetings.

16.3.4 Code Sharing and Development Feedback

A development team must be able to share its work and see feedback on individual work in the context of the whole team. Team members developing a software solution must be able to share their work artifacts in a way supporting versioning. Version control systems allow team members to share code and specify possible access restrictions to it. As the team is spread across multiple sites, all the sites and team members must have an access to the version control. In the case of Solita projects, Subversion (SVN)[5] is typically used. It is an easy tool to use and can be integrated with many IDEs or file management applications.

In the context of agile distributed teams, techniques of organising version control in a way that supports efficient development can also be implemented. One arrangement of SVN that we use is to have two branches for ongoing development and releasable code. That style of configuring, but in the context of multiple teams, has been described by Henrik Kniberg [14]. This SVN arrangement allows for having a stable code base at any time of the development.

In addition to collaboration on the shared code base, the team must be able to get quick feedback about status of the whole software they develop as a team. Continuous Integration (CI), which was described by Martin Fowler [15], is a technique that ensures that the code submitted to the version control is verified in a build process. The CI tool that we use for Java projects is Bamboo,[6] but there are also many open source alternatives.

---

[4]http://www.atlassian.com/software/jira.

[5]http://subversion.tigris.org.

[6]http://www.atlassian.com/software/bamboo/.

Practical Tip: We typically configure the CI tool in a way that all team members get notifications by email if build was unsuccessful. The successfulness of a build can be determined based on compiled code or test results. Usually, also other static analysis are done on the code so that, for example, coding conventions are verified.

16.3.5 Knowledge Sharing

Knowledge sharing among the members of a distributed team is essential for their success. Moreover in this case a tool that can be easily accessed by all team members regardless of their original organisation is most useful. Wiki-like web applications give much freedom in creating and managing documentation on-line. At Solita we primarily use a commercial tool Confluence.[7] There are many other commercial or free tool alternatives. The main advantage of wiki-like systems is the support for easy page creation, editing, sharing, so that team members can add and edit information themselves. Furthermore, page versioning reveals what kinds of changes have been done, and by whom.

16.3.6 Team Spirit

As a final note about subcontractors in teams, we would like to mention an important aspect of any cooperation, which is a good team spirit. Trust and good relations between all team members must be built up over time, but in distributed teams that aspect of team existence should not be neglected. One channel of providing feedback of the team spirit are the retrospective sessions where all team members can have a say about the project. However, in some cases this may not be enough, particularly if a project is a long lasting one, the team atmosphere can change over time. Therefore, it may be a good idea to create a survey about the team atmosphere. Such a survey can be easily created on-line in free services hosted externally. When the survey is organised this way, the team members can feel comfortable that their responses are anonymous. A survey should have questions covering different aspects of working in a distributed team, starting from technical challenges but also including motivation, identification with the team, workload (too much/too little work) and general feeling about the project work. The results of such a survey should be analysed and if needed corrective actions should be taken.

---

[7]http://www.atlassian.com/software/confluence/.

16.4 Subcontractors and Project Phases

In order to explain all the typical activities in a Scrum project that is distributed and uses subcontractors, we now present an illustrative example project walkthrough. The example project is not any particular project we have done, but rather a representation of a typical scenario, which should be more useful than any particular case. In this walkthrough we point out any issues that in real cases we found important and which may help the readers in their own projects.

This example project walkthrough starts from team assembling, and does not include any activities related to offering the project to the customer. The only selling point which is really relevant to the project organisation, is the project contract model. The most suitable for Scrum projects is a time-material type of contract as it is most natural for this kind of project organisation. On one hand the customer has the freedom to make changes and decide what has to be done, on the other hand the team is not internally limited by certain scope. Additionally, such contract models in a way force more active participation from the customer side, which provides better feedback from the customer. If the project is organised under a fixed scope and price contract, the scope is known, but the implementation order and internal communication of the team still can follow Scrum principles.

We can distinguish the following high level project phases: preparation, development, and release.

16.4.1 Preparation

In the preparation phase the project team is assembled. Knowing the exact or estimated project scope we can predict the resource needs for the project, and also the kind of expertise the project needs. At that phase the subcontractors are selected. Naturally, the initial decision that subcontracting partners can be used is done at the earlier stage and agreed with the customer. Depending on the required expertise a specific partner's resources can be allocated. Our Scrum project typically consists of a team of three to six members. There are no real limits to the size of a team but practically smaller teams may not be feasible, while larger ones may be considered for splitting into smaller groups. If the subcontractors work in a different location than other team members, the sites should be balanced, so that there is not a site with only one developer. In such arrangement the isolation of one developer might impact the team integration and communication needs. If there are at least two or more team members per site, they still can communicate locally and be equally involved, as a site, in the development.

Once the team is known the tools and access to them can be arranged. All the team members, regardless of their location, should have access to the needed tools. The tools depend on the technology of a project, but the communication, knowledge sharing, and tools providing feedback should be accessible across the whole team.

The first activity that the whole team does together is to explain the project scope to the team. It is so called project kick-off meeting. At that point the project practicalities must be explained, too. These include the processes in the project. In the case of subcontractors, especially if they work for the first time with us, it is essential to explain the methodology and ways of working together. If the team is experienced, only the new elements specific to the project must be explained. Additionally, the vision of the project and what is expected from it, must be clear to everybody.

> *Practical Tip:* It is good practice to have the processes and ways of work documented in a shared place, e.g., wiki. So that all the team members can access the documentation when they need to. Moreover as the practices are generally common, they can be reused in major parts across multiple projects.
>
> A project should also collect the basic contact information about all the team members, so that the people could easily find the correct person. Such an internal contact directory may also contain photographs of the people, so that the communication can be more personal even when it is done remotely.

Finally, after assembling the project team, including the subcontractors and internal personnel, the project development phase can start.

16.4.2 Development

The development phase follows the Scrum principles. Therefore the planning session is the first development activity. Such a meeting requires the whole team as well as preferably a customer representative and/or *product owner*. In practice often the *product owner* role is not performed by a person from outside of the team. So the role is taken by the technical lead who knows the background of the project. In some cases it is possible to have a customer representative who is able to present the customer point of view, but may not necessarily be technically capable of recognising dependencies and impact of different features. In that case the *product owner* role is shared between the customer and the technical leader in the project.

During the planning meeting the planned features should be discussed and a few first ones taken into *sprint backlog*, divided into tasks and estimated. This estimation is needed as the team commits to completing specific features in a *sprint*, and the whole team should decide how much time they need. At the project offering stage an initial rough estimation of the features is also done in order to predict possible effort needed for completion of all features. However, the initial estimate is not done by the whole team, it is done by experts planning the project. When the team estimates enough tasks for the following *sprint*, then the commitment for the *sprint* can be done. Additionally, as the efficiency of the team may not be known very thoroughly at the beginning, it is possible to have new features additionally estimated at the planning meeting. These features are not in the scope of a *sprint*, but they can be

taken in, if all other features for which the team commits are done before the end of the *sprint*.

Practical Tip: The distributed team, especially if working together for the first time, should have a chance to meet face-to-face. So the introduction meeting and the first planning meeting can be combined into one session that takes place at one of the development sites. When the team members meet for the first time it also may be worth organising an informal event that brings the team together. A common dinner may be one of the options to start building up the team spirit.

Furthermore in cases when important matters must be discussed a face-to-face meeting including at least the key people, may be more productive, than teleconferences or similar forms of communication.

For the backlogs we use just a spreadsheet which is updated during the planning session. The reason for using a simple spreadsheet is that it is quicker to edit than tasks in an issue tracker, and additionally it can be easily sent to a customer with indicated progress if needed. After the planning the project leader transfers all the tasks into the issue tracking tool.

Then on a daily basis the team discusses the current development status. It is important for the whole team to know beforehand when a feature is considered as complete and can be marked as such in the issue tracking tool. The definition of completion depends on the nature of the project, but for example, for a Java software implementation the definition could include: working code, unit tests, automated acceptance test cases, and documentation. If all these elements that are considered to be mandatory to regard a feature as complete are in place, only then a feature is really complete.

Practical Tip: If the team has problems with following the definition of completeness, individual tasks can be explicitly defined for each element of the definition, e.g., unit tests, documentation, etc. So that the tasks are at a low granularity level. Usually after some time the team members remember that each implemented feature must also include working unit tests and documentation.

For the meetings the teleconferencing tools can be used. If the team notices, and especially the *scrum-master*, that there are issues that must be clarified in order to proceed with the development, then additional workshops on a specific subject can be organised. Such an issue may be a technical impediment, but it can also be a need for finding out customer's preferences for the solution, or some technical decisions that need additional expertise. Furthermore the workshop in practice can be just a short phone call where the issue is discussed.

Practical Tip: If a team is not used to participating in daily Scrum meetings, it may occur that the status reporting becomes chaotic, and that can happen especially when the meeting is organised as a teleconference. A simple solution that we used in such cases, is to have the status records kept in a wiki page that is displayed and edited on a shared screen during the meeting. The status record contains the names of all team members, which determines the order of providing the status, and also the previous status, which allows for observing the changes in status every day. Finally, the use of written status reports, can help to avoid misunderstandings if the voice quality is not very good, as the reported status is visible to everybody.

During the development the team also receives constant feedback about the implementation status from Continuous Integration tool. The builds are done each time the changes are made to the code base and the notification of build failures are sent by email. Moreover all team members can check build statues in a web UI of the application, which again can be easily accessed by all team members regardless of their location.

A *sprint* ends with a demonstration and retrospective sessions. Practically that can be a single meeting. At the demonstration the completed features are presented. Features can be presented even if the development did not concern any user interface. A running process, web service, file transformations, or database content can be presented. If some features have not been completed (according to the completeness definition), the team should discuss why the features were not completed. Then those features should be re-estimated according to their statuses, and included in the following *sprint*, providing that the features are still within the scope of the project. After the demonstration, the team retrospective can be performed. If the customer actively participates in the development, they can also provide feedback during the retrospective. The team's feedback can be gathered as a *sprint* summary in a wiki page.

Finally, if there are any other features left for the development, then a new planning session is carried out. After that the whole Scrum *sprint* cycle is repeated. If there are any actions based on the retrospective feedback received that should be addressed at the time of new planning, they should be included in the planning. One important aspect of planning is the length of a *sprint*. If the length of the *sprint* seems to be a problem, then it should be adjusted. Typically our *sprint's* length is between two and four weeks.

Practical Tip: In the case of long-lasting projects before a retrospective session a survey about the team spirit can be carried out. Then the results can be discussed at the retrospective session and possible corrective actions can be agreed.

Furthermore at least at the end of each *sprint* the version control system is updated with the latest version of the solution, so that the changes are also trackable at the interim release level in addition to constant version control system updates done during the development.

If all the planned features are done, the development activities may be completed and the project transforms to the release phase.

> *Practical Tip:* We found it useful for team building to have a small celebration after a successful *sprint* completion. As the team is distributed common activities may be difficult to organise, but one way to have a common activity is to have a short team game session on-line, which was actually suggested by one of our customers. The game itself is not as important as a relaxed team activity. It also does not have to take much time, but rather encourage people to share their hobbies and interests.

16.4.3 Release

When the implementation is ready there can still be some final activities for the team. The software to be released should be tested well by the point of the release, however, the customer may require additional tests, or there may be some manual tests that should be executed once more when the software solution is fully completed. If the automated tests cover all the functionality, then naturally manual testing is not needed.

The final activities are the software packaging and possibly deployment. The packaging is always required as the customer must receive a full package containing all ordered deliverables, including code, documentation, test reports. If the customer also requested, the deployment may need assistance from the team side. After the deployment and acceptance tests on the customer side, the project ends, or if any problems are found, then they are fixed and software is delivered again.

The project ends when the software is accepted by the customer. Also then the team gathers for the last meeting to summarise the project. The overall feedback from the customer, team, and the project leader is analysed. Internally the feedback received can be used for the future projects. The team ends the project and the final solution is versioned as the final release. The team members are free to start work in new projects.

16.5 Conclusions

In this chapter we have presented our experiences with subcontracted Scrum teams in a software service company Solita. We discussed different agile practices and

tools highlighting their particular applicability in distributed and subcontracted Scrum teams. We also presented a process of selecting potential subcontracting partners suitable for specific companies, which in our case was a software service company. We presented the process steps and certain selection criteria relevant to our context. We also walked through an example Scrum project that used subcontractors. The example showed when particular agile practices can be used in different project phases. The combination of the selection and later working with subcontractors should provide the reader with a good overview of practices used successfully in Solita.

Additionally, we have shared our research results that can be regarded as indicative data for further comparison with other cases. The data collection and metrics we used, can be developed and adjusted to other organisations' needs. We also noted problems with our current data collection approach, which should be refined over time.

16.5.1 Practical Implications

We have discussed a real-life subcontractor selection process used in a software service company. We also indicated the process aspects (e.g., compatible methodology and culture) that are important during the subcontractor selection in addition to technology and business aspects. We have noted benefits of Scrum feedback loops in the case of distributed projects with subcontractors. We have also gathered a few practical tips that have been used in our projects with subcontractors. The tips included:

- Documentation practices in a way that allows all team members easy access and contribution,
- importance of face-to-face meetings in the process of building team trust and cooperation,
- usage of wiki pages for Scrum meeting minutes to facilitate daily meetings organised as teleconferences, and
- social team activities as an additional way of building team spirit.

These practical tips can be used directly in other Scrum teams, and they can be modified to fit the needs of other environments as well.

16.5.2 Research Implications

Our research findings and detailed data collected from 18 projects of different types, which included 8 Scrum projects, can be used as a reference point in the case of other studies. The presented data contains quantitative data obtained from various sources (e.g., company IT systems and interviews). The data covers a few years between 2006 and 2008, therefore it is relatively extensive for one organisation case.

The subcontracting process as well as the agile tools and practices will be further mastered and developed in our future projects. Therefore, a more detailed and broader study in terms of time and number of analysed projects can be presented in the future. Also we encourage other practitioners to present comparison of findings from theirs organisations, which would broaden the scope of the research in this area.

16.5.3 Summary

We hope that the presented practices and tools, as well as practical pieces of advice for each phase of a project life-cycle, will be a good source of information for other organisations. Our recommendations can be used by organisations that are about to start cooperation with subcontractors or by those who have already started their cooperation. Naturally, we are not providing a silver bullet recipe for cooperation with subcontractors, but our experiences applied and adjusted to the contexts of other organisation, should at least provide a good reference point for further improvements.

Appendix

Table 16.1 presents detailed findings for 8 Scrum projects and average values for all 18 investigated projects. The other project types included Iterative projects (marked Iter.) and traditional/waterfall projects (marked WF). We have used scale from 1 to 5, where 5 is the best result, for the following metrics: Customer satisfaction, Profitability, and Team performance. In subtotals for different project types, namely 'Scrum Result', 'Iter. Result', 'WF Result', and 'Grand Total', the project id (column Proj.) was replaced by the count of projects in given category. Additionally, other values in those last four rows represent average values for the corresponding project type.

Table 16.1 Quantitative results (Abbreviations: Proj. is Project id; Size (mm) is size in man months; Comm. factor is Communication factor; Sub. is Number of subcontractors; Meth. is Methodology used; Cust. satisf. is Customer satisfaction; Profit. is Profitability; Team perf. is Team performance; Cust. invol. is Customer involvement; PM time is Project Manager time)

Proj.	Size (mm)	Comm. factor	Team size	Sub.	Meth.	Cust. satisf.	Profit.	Team perf.	Sites	Cust. invol.	PM time
P03	27	24.00%	6	0	Scrum	4	3	4	2	Full	8.00%
P04	56	20.00%	6	2	Scrum	4	3	4	3	Full	9.00%
P07	19	30.00%	11	2	Scrum	3	4	4	2	Full	7.00%
P09	16	33.00%	4	2	Scrum	1	4	4	2	None	18.00%
P10	29	19.00%	11	2	Scrum	5	5	5	2	Partial	13.00%
P11	7	24.00%	3	1	Scrum	4	4	4	2	Partial	15.00%
P12	10	30.00%	7	5	Scrum	4	5	4	2	Full	14.00%
P18	5	45.00%	3	1	Scrum	5	5	4	2	Partial	16.00%
8	20.96	28.12%	6.38	1.88	Scrum Result	3.75	4.13	4.13	2.13		12.42%
4	29.87	22.76%	4.75	0.75	Iter. Result	4.5	2.5	3.5	1.75		10.22%
6	35.84	26.51%	5.83	2.33	WF Result	3.5	2.83	3.83	2		9.40%
18	27.9	26.39%	5.83	1.78	Grand Total	3.83	3.33	3.89	2		10.92%

References

1. Rudzki, J., Systä, T., & Mustonen, K. (2009). Subcontracting processes in software service organisations—an experience report. In Q. Wang, V. Garousi, R. J. Madachy, & D. Pfahl (Eds.), *Lecture notes in computer science: Vol. 5543. ICSP* (pp. 224–235). Berlin: Springer.
2. Rudzki, J., Hammouda, I., & Mikkola, T. (2009). Agile experiences in a software service company. In *SEAA '09. 35th Euromicro conference* (pp. 224–228). Washington: IEEE Computer Society.
3. Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). Goal question metric paradigm. In J.J. Marciniak (Ed.), *Encyclopaedia of software engineering* (Vol. 1, pp. 528–532).
4. Bird, C., Nagappan, N., Devanbu, P., Gall, H., & Murphy, B. (2009). Does distributed development affect software quality? An empirical case study of windows vista. In *ICSE '09: Proceedings of the 2009 IEEE 31st international conference on software engineering* (pp. 518–528). Washington: IEEE Computer Society.
5. Siakas, K. V., & Balstrup, B. (2006). Software outsourcing quality achieved by global virtual collaboration. *Software Process: Improvement and Practice*, *11*(3), 319–328.
6. Hyder, E. B., Heston, K. M., & Paulk, M. C. (2006). *The esourcing capability model for service providers (escm-sp) v2.01, part 1—the escm-sp-v2: Model overview*. CMU-ITSQC-06-006, Pittsburgh, PA: IT Services Qualification Center, Carnegie Mellon University.
7. Hyder, E. B., Heston, K. M., & Paulk, M. C. (2006). *The esourcing capability model for service providers (escm-sp) v2.01, part 2—the escm-sp-v2: Practice details*. CMU-ITSQC-06-007. Pittsburgh, PA: IT Services Qualification Center, Carnegie Mellon University.
8. Fowler, M. (2006). Using an agile software process with offshore development. Available online. http://martinfowler.com/articles/agileOffshore.html. Cited July 2006.
9. Schwaber, K. (1997). Scrum development process. In J. Sutherland et al. (Eds.), *OOPSLA business object design and implementation workshop*. London: Springer.
10. Sutherland, J., Viktorov, A., Blount, J., & Puntikov, N. (2007). Distributed scrum: Agile project management with outsourced development teams. In *HICSS '07: Proceedings of the 40th annual Hawaii international conference on system sciences* (p. 274a). Washington: IEEE Computer Society.
11. Paasivaara, M., Durasiewicz, S., & Lassenius, C. (2008). Using scrum in a globally distributed project: A case study. *Software Process: Improvement and Practice*, *13*(6), 527–544.
12. Herbsleb, J. D., Atkins, D. L., Boyer, D. G., Handel, M., & Finholt, T. A. (2002). Introducing instant messaging and chat in the workplace. In *CHI '02: Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 171–178). New York: ACM.
13. Cohn, M. (2005). *Agile estimating and planning*. New York: Prentice Hall.
14. Kniberg, H. (2008). Version control for multiple agile teams. Available online. http://www.infoq.com/articles/agile-version-control.
15. Fowler, M. (2006). Continuous integration. Available online. http://martinfowler.com/articles/continuousIntegration.html.

Further Reading

16. Eckstein, J. (2004). *Agile software development in the large: Diving into the deep*. Cambridge: Dorset House Publishing Company.
17. Upadrista, V. (2008). *Managing offshore development projects: An agile approach*. Oshawa: Multi-Media Publications.