Communication in Distributed Agile Development: A Case Study

Mikko Korkala and Pekka Abrahamsson
VTT Technical Research Centre of Finland
P.O.Box 1100, FIN-90571, Oulu, Finland
Mikko.Korkala@vtt.fi; Pekka.Abrahamsson@vtt.fi

Abstract

Distributed software development is an increasingly important development approach for software companies as it brings tempting opportunities. Distributed development is already burdened with several problems and agile methods bring further challenges in the form of their reliance on verbal communication and volatile requirements. There is little empirical knowledge on distributed agile software development. We conducted two distributed agile software development case studies and compared our findings against existing recommendations about communication in distributed agile development. Our findings along with existing literature conclude that presented recommendations are worthwhile considering in distributed agile development, but with some caution. Our empirically based findings indicate that the role of a well-defined customer is the key recommendation. The lack of a well-defined customer able to meet responsibilities, as well as volatile requirements and inefficient communication, can cause severe problems even in small-scale distributed agile software development projects. Discussed recommendations are complemented with an additional recommendation.

1. Introduction

Distributed software development (DSD)\(^1\) is becoming a common practice in modern software industry [e.g. 1], where the level of distribution can range from team members being located in the same city to those on different continents [2]. The significance of DSD has accelerated because of factors such as improving time-to-market through constant development across different time-zones, quick formation of virtual teams and the benefits of business market advantages. These needs have driven the software development efforts further towards a multi-site globally distributed environment. [1] Simultaneously, several studies have concluded that distributed enterprises are risky [e.g. 3-5]. For example, communication and coordination, software quality, schedule overruns and exceeded costs are some of the problems troubling both single-site and distributed software projects. However, the extent of the problem in the case of DSD seems to be so complex that a thorough understanding of it has not yet been defined. [3, 4] Several studies agree that communication is a particularly important issue in distributed agile development, [e.g. 5-7]. Agile methods rely on volatile requirements that are managed through efficient verbal communication [8] and thus agile software development methods pose their own challenges to the field of DSD.

In order to tackle the problems of DSD, several different techniques have been proposed. These techniques range from using different tools, such as instant messaging [9], videoconferencing [10] and whiteboard software [5] to a set of more general recommendations [5]. We conducted two different case studies with different levels of distribution ranging from the customer being in the same city, to one with a geographical distribution of 600 kilometers within the same country. Therefore, cultural differences were not an issue in these cases. We compared our findings against the recommendations of Layman et al. [5] and provide more insight on their application based on our empirical findings and the existing literature. Even though we were able to evaluate only three recommendations out of the existing four, our contribution provides valuable insight into conducting distributed agile projects. Our results further emphasize the critical role of effective communication, indicating that inefficient and irregular communication in conjunction with volatile requirements can cause severe problems even in very small-scale agile projects. However, it seems that effective communication is not the key. Our cases suggest that having a well-defined customer\(^2\) is the key recommendation affecting to recommendations about having a Development Manager [5] and using asynchronous communication channels. As ineffective customer collaboration may render the other recommendations redundant, effective customer collaboration seems to be a key factor for successful distributed agile development. In addition, we complement the existing recommendations by introducing

---

1 The literature discusses both Distributed Software Development (DSD) and Global Software Development (GSD). In this research, DSD is used to cover both terms if not explicitly stated otherwise.

2 In this study the term means those persons responsible for product requirements and prioritization, i.e. the case projects' customers.
an additional recommendation: i.e. enable and support direct communication between the developers. Unexpectedly, the teams in the second case were not allowed to communicate directly with each other. To compensate, a management-led communication channel was established to balance the communication flow, which gave poor results. This factor severely blocked the progress of the project. As a limitation in this study, the results are drawn only from the viewpoint of the development teams.

The content of this paper is as follows: the next section reviews the existing literature on DSD and agile DSD. This is continued by setting out the research settings. Section four presents the results, and a discussion of the results, and the paper concludes with final remarks and indications for future research.

2. Related literature

This section discusses DSD and distributed agile development.

DSD allows geographically independent software development for companies of all sizes and it is becoming a common practice in the modern software development industry [1, 3-5, 11]. Several factors have been contributing to the growing interest in distributed development, such as time-zone independent 24 hour development, reduced costs, access to well-educated labour and maturation of the technical infrastructure, just to name a few [4, 12-14]. Even though these factors inevitably provide tempting opportunities for software companies, DSD is troubled by the same problems as single-site efforts. Communication, requirements engineering, cost related problems, problems with quality and schedule are common issues in single-site development alone, and distribution makes these problems even more complex. [3, 4] Additionally, language and cultural differences create problems in DSD [12]. It has been also found that distributed development tasks take about 2.5 times longer to complete compared to co-located tasks due to communication and coordination related issues [9]. Komi-Sirviö and Tihinen [4] conducted an extensive study into charting the main problems encountered in DSD. Their results were categorized into nine specific problem areas, and the top three issues were 1) Development tools and environment, 2) Communications and contacts and 3) Design knowledge. Problems with tools and environment were the number one problem source within the large companies. The problems related to incompatible development tools and to the functionality of network connections. Medium sized and small organizations suffered mostly from design knowledge related issues, such as situations in which the software design and implementation take place in separate locations. While not being the main source of problems, 74% of the respondents identified communication as a problem in DSD. In general, the problem aspect of DSD seems to be so complex that a thorough understanding of it has not been reached. [4]

The research on distributed agile development has identified communication as one of the main issues to be taken into account [e.g. 5, 7, 10]. Agile software development involves highly volatile requirements which are managed through efficient verbal communication [8]. Effective communication is a crucial element in software production regardless of the development approach [8, 15, 16], and it can be considered even more important in agile software development due to its paramount role. In addition, the lack of informal communication results in lower awareness [5, 7] and poor coordination [17]. Awareness is defined as [18]: “an understanding of the activities of others”. Coordination helps individuals to view, plan and execute their actions in relation to their colleagues’ actions towards a common goal. Awareness is a key concept affecting coordination in a distributed environment. [7]

Requirements engineering has also proven difficult in DSD [3, 4]. According to [4] requirements related issues were the major source of software errors, and communication makes this critical subject even more challenging [3-5]. Since in agile development the software requirements are documented on a very general level and communication can be considered cumbersome in distributed agile development, it is quite safe to argue that the volatility of agile requirements in conjunction with troublesome communication can create significant risks for distributed agile development efforts.

Several solutions for tackling the problems in distributed agile development have been proposed. These techniques include e.g. different support tools, such as instant messaging [9], videoconferencing [10] and whiteboard software [5]. Also more general level recommendations have been made. Layman et al. [5] have proposed guidelines presented in Table 1 for distributed agile development. They conducted a distributed case study using Extreme Programming [8]. The developers were located in the Czech Republic, while the management was situated in the USA.

Table 1. Recommendations by Layman et al. [5]

<table>
<thead>
<tr>
<th>Proposed recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a)</strong> Define a person to play the role of the customer up front. This individual must be able to make conclusive decisions on project functionality and scope, must be readily accessible, and must have a vested interest in the project.</td>
</tr>
<tr>
<td><strong>b)</strong> When the project management and development teams are separated, create a role within the XP team whose purpose is to work closely with both development and project management teams on a daily basis, preferably someone who speaks all the languages involved.</td>
</tr>
<tr>
<td><strong>c)</strong> When face-to-face, synchronous communication is</td>
</tr>
</tbody>
</table>
These recommendations aim to create a communication-rich environment for projects unable to achieve it due to their distributed nature. Even though these recommendations have been developed as a result of agile GSD effort spreading over different time-zones, they could be applied also to efforts with a less distributed context.

To define the level of distribution we applied the model proposed by Prikladnicki et al. [2]. The model can be used to analyze the distribution through different scenarios in which the model’s actors interact. The suggested model is set out in Figure 1.

![Figure 1. DSD Distribution Level](image)

The model consists of the following actors: **Project Team, Customers and Users.** A **Project Team** includes everyone involved in the development including developers, business-oriented personnel, testers etc. **Users** represent people responsible for providing e.g. requirements, and **Customers** are the person(s) or organization that requested the project. The term **Intra-Group** defines the distance inside the different stakeholder groups, while **Inter-Group** refers to the distance between the different actors [2]. The stakeholder groups interact within different **scenarios** involving both intra- and inter-group distances [2]. According to the model, the level of distribution can range from relatively small, e.g. the **Project Team** and the **Customers** are located in adjacent buildings, to larger scenarios of global distribution [2].

This section presents the research method, research settings and data collection techniques. Also the levels of distribution in the case projects are described.

### 3.1 Research method, research settings and data collection

Our research covered two separate semi-industrial case studies [19] providing real software with real business value for real customers. The project team in **Case A** consisted of one experienced professional developer from the customer organization, and three experienced 5-6th year information processing science students working in a single shared environment. The target was to implement a mobile version of already existing software as an extension of that product. The customer was located in the same city and participated personally in the process during iteration planning and release phases.

In **Case B**, there were two development teams working on the same product and sharing a single customer. The project implemented a mobile front-end application that was integrated into the customer’s main product. The front-end team was located in Helsinki, Finland and the main product team in Oulu, Finland was located approximately 600 kilometers away. The customer spent the first two weeks of the project with the Oulu team and for the rest of the effort he was located with the Helsinki team. Proxy customers were not utilized. The customer organization was located in Helsinki and **Case B** was not affected by time zone differences.

The Oulu team in **Case B** consisted of four to seven developers for the different iterations. This composition included an experienced developer and a Test Driven Development specialist.

The case projects were implemented following Mobile-D™, which is an incremental, iterative, agile inspired development methodology specially designed for mobile application development [20, 21]. Because of the limitations of this paper, Mobile-D™ is only briefly discussed. The data was collected using the following techniques: **onsite observations** in **Case A**, **final interviews** with the developers in **Case A** and the developers of the Oulu team in **Case B**. The interviews were semi-structured. A personal research diary and email correspondence between the team and the customer were utilized in both cases. In **Case B** the email correspondence between the customer and the Oulu team was examined. One team member from **Case B** was separately interviewed after the project using a semi-structured interview to cover issues not discussed in the final interview. This developer was selected to be interviewed since he was one of the most experienced developers and the one most easily available for the interview.
The customers were not interviewed which can be considered a limitation of the study.

3.2 Levels of distribution in the case projects

The case projects applied different levels of distribution. Following the distribution level model [2], the distribution in Case A appears in Figure 2. In both cases the Users and Customers were the same, which is often the case [2].

![Figure 2. Distribution in Case A based on [2].](image)

Since the developers were sharing the same workspace, the Project Team was centralized, while the Inter-Group distribution between the team and the other stakeholders followed a Cross Town Scenario. There was only one customer, so the distribution within the Customers/Users was the Same Physical Localization Scenario. A similar analysis was conducted in Case B, and appears in Figure 3.

![Figure 3. Distribution in Case B based on [2].](image)

In Case B the Project Team was distributed. The Customers/Users spent time with both of the teams, one team at a time. Therefore the Inter-Group distribution falls into a No Time Shift Scenario -category since the stakeholders were located in the same country without constant collaboration between all parties involved.

4. Results

In this section the results of the study are presented. We discuss our findings against the recommendations by Layman et al. [5].

**Recommendation 1:** Define a person to play the role of the customer upfront. This individual must be able to make conclusive decisions on project functionality and scope, must be readily accessible, and must have a vested interest in the project.

The study conducted by Layman et al. [5] indicates that the customer role is essential for effective requirements management in a distributed agile project, and active, consistent customer involvement cannot be overlooked.

The customer in Case A participated personally during the iteration planning and release phases. Our findings on this recommendation are drawn from Case B. During the Case B the customer was at first participating actively in the development with the Oulu team. He was available for continuous feedback for the first two weeks, afterwards he stayed with the Helsinki team. After his departure the communication was managed by daily emails and during the planning and release phases by telephone. The customer communication left a lot to be desired from the perspective of the developers. When he [the customer] was present he was active, but when otherwise he should...
have been more active (On the customer communication. A Developer, Case B).

In addition to the lack of efficient customer communication, the communication of technical issues between the teams was not allowed. All communication between the teams was passed through the customer. Since the customer communication was not effective, it is not surprising that the Oulu team encountered significant problems during the development. The Oulu team, for example, did not receive any information about the communication protocol between the mobile front-end and the main application. This may indicate a lack of trust between the customer organization and the Oulu team. Prikladnicki et al. [22] identified trust as one of the difficulties in distributed development. Whether or not the lack of trust was the reason why the teams were not allowed to communicate directly, communication between separate teams should be allowed and maintained. This assumption can be supported by the finding made by Layman et al. [5]: “The customer noted that cooperation and teamwork were the biggest success factors in this project”. Thus active customer communication in conjunction with an environment supporting direct communication between the teams is essential for a successful result. The importance of peer-to-peer links is also mentioned in [23].

Recommendation 2: When the project management and development teams are separated, create a role within the XP team whose purpose is to work closely with both development and project management teams on a daily basis, preferably someone who speaks all the languages involved.

The project team in Case A was complemented with an experienced developer from the customer organization. This developer had a profound understanding of the software domain and he had been participating in the development of the desktop version of the same product. He had a strong working relationship with the customer so it seemed natural that he assumed the role of the Development Manager [5] – the role is described in recommendation two. However, he did not communicate with the customer on a daily basis. During the project the requirements analysis process was conducted almost invariably by the Development Manager and the customer. When the other team members were asked why they were not participating in the process, the answer was short: “He [the Development Manager] knows what to do” (Developer, Case A). The team also mistakenly referred to the Development Manager as the “Onsite customer” even though he did not possess any power over the requirements definition or project scope. The passiveness of the team during the requirements analysis, along with the misunderstood role of the Development Manager, may indicate that the he was the main source of information for the rest of the team. The following comment from the final interview supports our view: “if he [the Development Manager] hadn’t participated in this project, we would have had to ask the customer a lot more questions and the requirements analysis would have been a lot more difficult” (A Developer, Case A). We argue that these comments indicate that having a member assuming the role of the Development Manager can create a false sense of confidence and security. The false sense of security is not a new issue in the field of agile development [24].

Another factor that needs to be taken into account if having a Development Manager is to emphasize that the requirements analysis process (for example) is not the responsibility of the Development Manager alone. We contend that allocating this task to a single person can reduce the possibilities of understanding the functionalities correctly. More people involving in requirements analysis process can reveal issues that can go unnoticed if the process is conducted only by the Development Manager and the customer. In addition, having a Development Manager may increase the chance of information distortion as, according to [25], information mutates and some of it gets lost if it passes through different people. In addition, similar viewpoints have been presented e.g. in [26]. In addition, it is also quite safe to argue that using intermediaries between the development and management sides is pointless if the customer is not able or willing to work actively with the Development Manager.

Recommendation 3: When face-to-face, synchronous communication is infeasible, use an email listserv to increase the chance of a response and encourage prompt, useful, and conclusive responses to emails.

During our research the daily iteration time communication between the developers and the customers was managed through email (Case A) and email and telephone (Case B). These channels were specifically requested by the customers during both cases, despite the wide array of different existing communication channels (see section 2 for examples and [27]). Layman et al. [5] had positive experiences using asynchronous communication by utilizing mailing lists to share information. Their findings indicate that the customer was actively and constantly participating in the development, i.e. the customer was well-defined.

On the other hand, empirical evidence indicates that increasing reliance on so called lean, asynchronous communication channels can result in higher software defect rates [28]. During Case A, a total of 18 daily emails about the project were sent to the customer. All of these messages were brief status reports describing what was done during the day. Questions were not asked in these
reports. Thus it seems that the team was confident about what they were doing. During Case B, a total of 46 reports were sent. Three of these reports included questions for the customer, and the customer responded only to one issue. The average defect rate was 41.8%, while 62.6% of all defects were caused by inefficient customer communication [28]. These findings are in contradiction with the results by Layman et al. [5], indicating that this recommendation is strongly linked to the recommendation for a well-defined customer. Thus we conclude that if the customer is not able or willing to participate actively in the communication when needed, any communication mechanism becomes redundant.

**Recommendation 4: Use globally-available project management tools to record and monitor the project status on a daily basis.**

Shared project management tools were not utilized in either of the cases.

In summary it seems that a well-defined customer (recommendation 1) is the key recommendation. If the customer is not able or willing to participate actively in the development, this makes the use of intermediaries between the development and customer sides, and use of mailing lists or any medium used in customer-developer communication, fruitless. In addition, we conclude that enabling and fostering direct inter-team communication deserves considerable attention. We propose this as an additional recommendation to the existing recommendations.

5. Conclusions

Distributed software development (DSD) is becoming a common practice in modern software development [5]. It both provides exciting opportunities [4, 12-14] and poses significant risks to the success of distributed projects [3-5]. Agile software development methodologies bring challenges to the field of DSD in the form of volatile requirements managed by informal communication. Several tools proposed in [e.g. 9, 10] have been used to tackle the problems of distributed development. Recent research has provided more general communication related recommendations focusing on distributed agile development [5]. The aim of this paper was to review these recommendations, and increase the knowledge about their usage based on the existing literature and empirical findings from two different small-scale distributed agile development efforts with different levels of distribution. Table 3 summarizes the recommendations made by Layman et al. [5] and illustrates how they were implemented in our research and are complemented with our empirical findings. We conclude by presenting our empirically based additional recommendation: Enable and support direct communication between the developers.

Table 3. Recommendations by Layman et al. [5] their utilization in this study and corresponding findings, including a new recommendation.

<table>
<thead>
<tr>
<th>Recommendations [5]</th>
<th>Utilized in this study</th>
<th>Findings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Define a person to play the role of the customer up front. This individual must be able to make conclusive decisions on project functionality and scope, must be readily accessible, and must have a vested interest in the project.</td>
<td>During Case A the team had a single customer located in the same city. The customer was accessible during iteration planning and release. During Case B the customer spent the first two weeks constantly with the developers located in Oulu, Finland, afterwards he was not accessible. Both customers were in a situation to make decisions on functionality and scope.</td>
<td>Our findings from this area come primarily from Case B, concerning the team located in Oulu, Finland. The customer participated actively for the first two weeks; afterwards the communication was diminished radically. The customer replied only once to the email clarification requests sent by the team. This ineffective communication backfired with an average of 41.8% defect rate [28].</td>
</tr>
<tr>
<td>2: When the project management and development teams are separated, create a role within the XP team whose purpose is to work closely with both development and project management teams on a daily basis, preferably someone who speaks all the languages involved.</td>
<td>The Development Manager role [5] was utilized in Case A. The Development Manager was a member of the customer organization. The Development Manager was not communicating with the customer in daily basis. Case B did not have a Development Manager.</td>
<td>During our study the Development Manager seemed to be the main source of information for the developers in Case A. Personal customer communication e.g. requirements analysis was almost entirely managed by the Development Manager alone. It was admitted that without him the requirements analysis would have been a lot more difficult. We argue that having a Development Manager can create a false sense of confidence within</td>
</tr>
</tbody>
</table>
the team. There is also a chance of information distortion e.g. [25, 26]. The whole team should be encouraged to participate e.g. in planning actions. However if the customer is not well-defined, the Development Manager becomes redundant. Thus this recommendation is linked to recommendation 1.

3: When face-to-face, synchronous communication is infeasible, use an email listserv to increase the chance of a response and encourage prompt, useful, and conclusive responses to emails.

While the customer was not personally available, the communication was managed by email (Case A). Email, like a mailing list, is an asynchronous communication channel. In addition to email, also the telephone was used in customer communication (Case B).

Asynchronous communication can be used if all the parties involved in the development are committed to communicating actively. Empirical evidence has indicated that communication increasingly relying on asynchronous communication media can increase software defect rates accordingly [28]. This recommendation is highly linked to recommendation 1. If the customer is not able or willing to participate actively in the communication, the utilization of any communication mechanism becomes redundant.

4: Use globally-available project management tools to record and monitor the project status on a daily basis.

Not utilized in this study.

We have no findings from this area.

**New recommendation:** Enable and support direct communication between the developers.

The teams (Case B) were not allowed to communicate directly, but only through the customer who did not communicate actively with the Oulu team after the initial two weeks.

The teams should be able to communicate directly in order to achieve successful results. The lack of direct peer-to-peer communication can result in significant problems.

Even though the cases were not globally distributed as in [5], the presented recommendations can be viable also in a less distributed context, if they are fulfilled properly.

Based on the empirical data it seems that having a well-defined customer (recommendation 1) is the key to successful distributed agile development. Without proper customer collaboration, the other two investigated recommendations become redundant. Thus the customer relationship should be given extra effort in planning, managing and executing distributed agile projects. This relationship should also be maintained throughout the project.

As a limitation in this study, the results are drawn from the developers’ points of view alone. Therefore, further research is required on the viewpoints of all the stakeholders for additional validation of all the existing recommendations, and possible discovery of new recommendations.

**References**


[6] C. Poole, "Distributed product development using


[27] E. Gottesdiener, *Requirements by Collaboration*. Addison-Wesley, 2002,