Distributed Software Development, Issues and Benefits, in a Norwegian Context

Hans Kristian Flaatten, *BS Informatics, NTNU*

Abstract

This paper discuss problems with distributed software development that are not present when the all of the development happens on the same site. It brings up different aspect of the issues with awareness as well as tools and methods for communication and coordination in a distributed environment. The paper then discuss these issues in a Norwegian context related to a distributed software development project between two medium sized companies.

Index Terms

Software Development, Collaboration, Geographicly Distributed Teams, Online Collaboration, Communication Tools, Awareness, Cordination.

1 MOTIVATION

Software development projects are fundamentally cooperative. Most commercial projects today are fairly complex and requires a huge collaborative effort in order to succeed. While these projects have gotten more complex over the years; the importance of good communication within the team, as well as a continuos flow of up-to-date, and relevant, information, becomes more important [1, p. 1107]!

There is a shift in the industry towards more globalization of software development teams. This brings a geographical distance and creates additional challenges to development process as direct communication through physical appearance is no longer present [2, p. 174].

With employees are working more distributed, and a greater access to affordable software products of the shelf, video conferencing is on the rise. A study conducted by

the technology company Brother concluded that among small and medium-sized UK businesses 50% of their workers are working remotely and this trend has been increasing by 36.5% over the last few years [3].

A project manager for an European Union pilot project called SPOCS [1] recently made an interview regarding their use of collaborative tools in a distributed environment. With a team scattered over 200 km^2, communication has been a constant challenge. Furthermore, the project includes many components with several other European companies working with them.

The team had no common vision on the project before the introduction of a collaborative management system named Trello[2]. As a result their conference calls are more efficient in the sense that the duration of meetings went from 40 to 15 minutes because they can follow lists and labels that were available to everyone in Trello. They experienced a huge increase in the overall efficiency of the project [4].

1.1 A Day at the Office

I work as a consultant system engineer for a joint project between two medium sized companies. One head quartered from Oslo, the other in Bergen, however, half of the team in the second company work form their major branch in Oslo. And than it is me, working from Trondheim.

A normal day starts at 8 in the morning, as most members of the project team do. I start by checking in on Skype[3], reveling my presence to the other team members as well as checking for any unread messages since last time I logged off. The time before our daily Scrum[4] meeting is spent while hanging out in the dedicated group chat room; doing small talk. At exactly 8:30 the video conference starts.

The Srcum meetings are usually short and effective; where the Project Manager goes through everyone's task in the Issue & Bug Tracker JIRA[5], prioritizing them if necessary.

1. www.eu-spocs.eu
2. www.trello.com
3. www.skype.com
4. An Agile software development methodology
5. www.atlassian.com/JIRA

This way everyone in the team knows what each member will be doing for the rest of the day, and it also allows for group discussions and comments regarding different tasks and priorities.

During the day our shared chat room serves as a bulletin board where important information and questions are posted. In order to keep everyone in the loop an "@" followed by a team members first name is placed in the beginning of messages directed to a certain person in stead of using a private chat room. If the question or conversation is of a more complex character, a private video conference is usually preferred to prevent cluttering up the shared chat room.

1.2 Scope of the Paper

The definition of distributed software development is extremely broad. It covers everything from Open-Source Software (OSS) project which are almost always a collaborative effort. Developers from all parts of the world, who never meet one another face-to-face, are working together [5]. Many OSS projects have managed to create highly complex systems that are widely used, and are often industry standard, such as "The Apache HTTP Server Project" [6] with a 57% marked share across all domains on the public Internet [6].

Another extreme end of the scale is "Crowd-sourcing" which is a "collaborative input from a volunteer web-based user community" [7, p. 52]. In these distributed environments barriers like time-zones (temporal distance), and cultural and linguistically (Socio-cultural) differences are commonly found [2, p. 177].

Same-country distributed collaboration software projects are mostly concerned with lack of *awareness*, and inadequate *communication* and *coordination* [1, p. 1108]. Herbsleb and Mocus [8] suggests that issues with coordination and communication is a huge source of loss in development speed.

2 Literature Review

An absolutely essential part of the software development process is the ability for developers to maintain a commonly shared understanding of the project; it's tasks and

6. httpd.apache.org

artifacts. [1]. Software development project are constantly subject to unforeseen changes throughout the development cycle either through increased competition, products, standards and regulations as well as customer requirements [8].

2.1 Awareness

There are many definitions of awareness in different domains but in collaboration it is the ability to perceive information from the working environment and understanding the activities [1, p. 1110].

According to Omoronyia et al. awareness, or in many cases the lack of awareness, is one of the key aspects of collaboration. In it's pure essence; awareness is mission critical in order for any software development team to succeed [1, p. 1108]. Maintaining a satisfactory level of awareness has also been proven more difficult in distributed environments compared with those who are co-located – where all members of the software development team are in a physical proximity [9].

2.1.1 Awareness Types

Several types of awareness exists, and they all overlap and relate to one another as shown in Figure 1. A few of them are particularly important when talking about distributed collaborative work and are concerned with the group dynamics [1]. These are as follows:

- Workspace awareness: *Knowledge on recent collaborative activities and how it all fits together in a larger perspective*
- Informal awareness: *A general, often almost subconscious, knowledge of coworkers, what they are doing and will be doing [10]*
- Group-structural awareness: *Knowledge on responsibilities, positions and roles of people within the group [10]*
- Social awareness: *An extension to informal awareness and covers attention, interests, connections and availability of individuals*
- Context awareness: *Knowledge across different awareness types and is concerned with them changing states.*

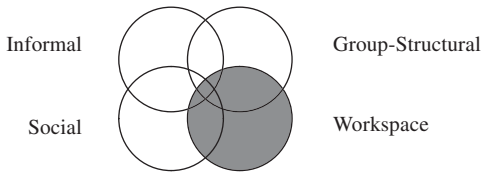

Fig. 1. How awareness types overlap and interact [10, p. 287]

The more subtle category of awareness, *context awareness*, arrises when, over time, individual collaborators each work on a different set of tasks, and artifacts. Because of this each person form their own perception of their co-workers; what they work with, which roles they play and relationships. This is also known as ”meta-awareness”; awareness about awareness. “The evolving internal and external state information that fully characterizes the situation of each entity in a shared environment” [1].

2.1.2 Maintaining Awareness

In order to maintain awareness in a distributed environment–the following set of ”objects” and artifacts must be correctly addressed; *identity*, *changes*, *actions*, *activity level* and *objects*. This is done by maintaining focus on awareness during general collaboration across the project [1].

Gutwin et al. propose in their paper on ”Group Awareness in Distributed Software Development” that there are three main mechanisms for “maintaining awareness in co-located situations” [5, p. 73]: 1) *explicit communication* between team members, 2) *cosequential communication* by watching a person’s activities, and 3) *feedthrough* by observing project changes on artifacts.

2.2 Communication

In a study by Herbsleb et al. if was found that there was a lack of informal communication between developers collaborating remotely which resulted in major coordination problems and ultimately an overall decrease in collaboration [8]. Awareness is something that is often taken for granted in co-located environments, and is hence

very difficult to maintain in a distributed setting without doing additional efforts in maintaining good communication [5].

2.2.1 Onsite Communication

Perry et al. presents in their empirical study on co-located developers, and how they utilize their time, that "one of the most salient impressions conveyed by observation was the sheer amount of time each developer spent in informal communication." [11, p. 41] On average a developer could spend up-to 75 minutes each day on "unplanned interpersonal interactions", which are informal meetings between two or more developers.

In a study of a telecommunications software project, conducted by Herbsleb et al. [12], an analysis on use of time was conducted. The authors looked closer on how developers' use of time evolved during the development of the project. They observed that during the first month, when the project was in it's design phase, almost 50% of the time was spent doing "group activities". This included group meetings and other work-related discussions. The study continued for 7 more months where they saw a steep decline in time spent on group activities. At the end this portion only accounted for 10% of the total time used last month. A possible explanation was that design activities accounted for the largest part of all intra communications [8, p. 491].

2.2.2 Cross-site Communication

Herbsleb and Mockus performed an experiment where they separated developers' offices in a medium sized software company with as little as 30 meters. They saw was that communication between coworkers dropped substantially, almost to the same level as previous studies had found for offices separated by hundred of kilometers [8]. "Frequency of communication generally drops off sharply with physical separation among coworkers' offices and that the sphere of frequent communication is surprisingly small." [8, p. 481]

Problems within the development team quickly arrises with inadequate communication and the most common ones are; decreased collaboration, lack of awareness, poor visibility and diminished trust [1]. The most noticeable reported effect of these kinds

of problems with cross-site communication is an increase in delay. Everything start to take more time as communication continues to drop [8].

2.2.3 Communication Tools

A variety of technologies have been developed in order to stimulate causal conversations among co-workers such as video, audio, and instant messaging [8, p. 482]. However ”a critical mass of users is essential for the success of any communication system“ [13]

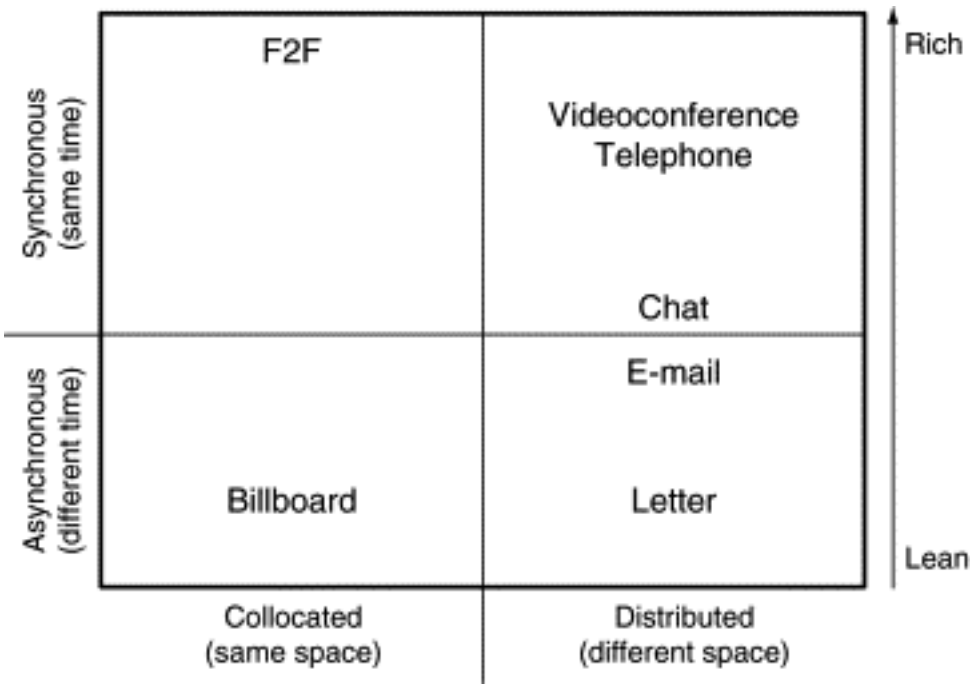

Fig. 2. Time/Space Classification Matrix for Communication Tools [2, p. 184]

Unlike many groupware applications the obvious beneficiaries of emails are not the managers or discussion-makers, but the users. This is the exception to the rule on who benefits from groupware applications [14, p. 258]. There is a relatively equal division between who does the work and who gets the benefit, provided that everyone takes equal turns sending and receiving. Gutwin at al. suggest that designers should be careful when designing new tools for distributed software development [5, p. 73].

2.2.4 Communication in Open Source Project

Gutwin et. al did an extensive study on communication and collaboration in three major Open Source Software groups [5]. To their big surprise the main mechanisms for maintaining awareness was plain and simple text based communication tools such as mailing lists and text chats. These were the project's main source of information. Both of these two mechanisms were publicly available, so that anyone could become a peripheral participant at any time.

The mailing lists provided people with a tool to find the experts with ease. All messages went to the entire group and experts would simply identify themselves and join the discussion. Text chats was primarily used for informal conversations, but if the right people were present they could just as well be used for technical discussions.

The study concluded that the most likely reason that these methods for communication worked so well was a result of the strong culture in OSS communities of "making it public". They also stressed the importance of promoting this mindset to other groups of distributed software development in order to make this level of group awareness possible there as well.

2.3 Coordination

Successful project coordination requires knowledge about all the different activities within the project; thus a good level of group awareness is crucial. The level of awareness needed, in a certain project, depends on the degree developers need to coordinate their activities with one another [5, p. 74].

Ineffective communication was found to lead to serious coordination problems. Among these was "unrecognized conflicts" and "incorrect interpretation of communications" the two major ones [8].

A way to reduce the need for coordination and awareness is through partitioning, where developers work in tightly constrained areas. This restricts each person's awareness needs to only their own code and its immediate dependencies. [5, p. 74]

2.3.1 Tools

Grudin et al. suggests that the best demonstration of the potential of a computer-supported operative work is a project management application which is running on a distributed system. Such an application covers scheduling, and chronologically ordering, of activities, as well as management of milestones, and monitoring of responsibilities and resources [14].

The individual use of collaborative tools creates an certain overhead, and for some individuals it may increase their workload. A good example is a shared calendar system that is responsible for automatic meeting scheduling. Such a tool is excellent for management and makes their work easier, but for the individual employee it ends up creating more work without any inherently benefit for them. Gudin et al. suggests that if the collective benefit increases the implementation of collaborative tools will be more successful. Usually this means that organizations must recognize how much the company may be losing on ineffective collaboration [14, p. 248].

There are a whole range of different group-ware applications. The general definition are any application that can be used to interact with two or more persons at, at the same time [14, p. 246].

- Communication tools: *Send messages, files, and other data. These tools facilitate the sharing of information between team members.*
- Conferencing tools: *Sharing information between team members in an interactive way.*
- Collaborative management tools: *Management tools that facilitate and manage group activities.*

2.3.2 Who Benefits?

Groupware applications can create very quickly create additional work for it's users by failing to account for wide range of errors, exceptions and improvisations that are common characteristic of group activity and collaboration [14].

As for the example with the automatic meeting scheduler; the managers who are the ones making final design decisions may only recognize their own benefits of such an automatic meeting scheduler, without any thought if the tool may end up forcing other

users to do extra work in order to support this feature. For them this would be no benefit at all!

Grudin et al. discussed the importance of asking three simple questions 1) Who is the immediate beneficiary of the software, 2) Who will be forced to extra work in order to make the software a success?, and 3) What is the motivation for users to do this extra work?.

"The best solution is to insure that everyone benefits directly from using the application."[14, p. 250]

3 Discussion

In theory distributed software development provides a lot of benefits; but it is not always that glamourous. As a part time consultant and full time student I am rarely working "9 to 5". I work whenever I find the time; which sometimes are evenings, nights, weekends and holidays. Not very different from how a coworker in another time zone would work actually. At these times there are not many other team members online to answer my questions so I often have to rely on inaccurate, outdated and sometimes completely wrong information when I am working. Some times I am unable to proceed because important information is not available online at that time and I have to wait for a reply when requesting information. Project managers, in my experience, are not working later then what they are payed to do.

Maintaining and enhancing awareness has been proven more difficult in a distributed environment than a co-located one. There there are several of reasons but the most inherent is the fact that the awareness throughout distributed software collaboration is very dynamic, tacit, and highly contextual. Hence, this awareness is extremely challenging to distribute automatically and i a good fashion [1]. The dynamic nature of awareness stems from the never ending change of state in software development projects, and the relevance of information varies through the differing stages in the software development process.

Studies on distributed software development conclude that poor visibility within the development team as well as the lack of control of remote resources; inadequate communication, and coordination across the members of the distributed team, and a lack

of shared context are apparent problems in distributed environments. As a result of this there may be diminished trust within the development team, difficulty to initiate contact and misunderstandings will occurs more frequently then in co-located workplaces.

My experience with distributed software development is over all fairly good. At most times I am able to do my work by adapting to the working environment and proactively keeping myself up to date on what the rest of the team is doing; especially on those tasks that are related to what I am doing at the time. Because we have a dedicated project manager who's only responsibility is managing the team, making sure that everyone knows what to do and that critical information is distributed. He also serves as a proxy; any question can be directed to him and he will find the best suited person to answer it. He also manages our Issue & Bug Tracker by creating and closing tickets, shuffling them around according to the latest developments. This enables everyone on the team to get a correct sense of all the moving parts, all activity and the general state of the project. The team is also completely open, there are no secrets and everyone are constantly brought up to date on the latest developments. These two factors are what I think are the main reason for our success as a distributed project.

If I should have studied these problems further in my organization I would have focused more on what makes our project work well. What are the things we have done right and how can they be generalized into theories on distributed development best practices? These would then again need to be tested further in other case studies. There are already a lot of literature on the various problems related to distributed software development but few of them seams to provide any good solutions on how distributed development is done right, only what they are doing wrong. Since distributed collaboration, in many aspects, varies so much from onsite collaboration it might be reasonable to think that there exists completely new ”facts” for distributed development and that these can not simply be adopted from those already existing for co located collaboration. One might need to have a look at distributed collaboration in a whole new light!

REFERENCES

[1] M. R. M. W. I. Omoronyia, J. Ferguson, “A review of awareness in distributed collaborative software engineering,” *Software - Practicie and Expertice*, vol. 40, pp. 1107–1133, 2010.

[2] F. Lanubile, "Collaboration in distributed development," in *Software Engineering*, F. F. A. De Lucia, Ed.

[3] E. Nation. (2012, aug) Info-graphic: The growth of video conferencing in the uk. [Online]. Available: http://www.enterprisenation.com/blog/infographic-the-growth-of-video-conferencing-in-the-uk/

[4] F. C. Software. (2012, jul) "how do you use trello?" contest winners! [Online]. Available: http://blog.trello.com/how-do-you-use-trello-contest-winners/

[5] K. S. C. Gutwin, R. Penner, "Group awareness in distributed software development."

[6] Netcraft. (2012, nov) November 2012 web server survey.

[7] K. F. Z. E. J. K. D. Lewis, S. Curran, "Web service integration for next generation localisation," Morristown, New Jersy: Association for Computational Linguistics, 2009, pp. 47–55.

[8] A. M. J.D. Herbsleb, "An empirical study of speed and communication in globally distributed software development," *IEEE Translations on Software Engineering*, vol. 29, no. 9, pp. 481–494, 2003.

[9] C. Crampton, "The mutual knowledge problem and its consequences in geographically dispersed teams," *Organisational Science*, vol. 12, no. 3, pp. 346–371, 2001.

[10] M. R. C. Gutwin, S. Greenberg, "Workspace awareness in real-time distrubted groupware: Framework, widgets, and evaluation."

[11] L. V. D.E. Perry, N.A. Staudenmayer, "People, organizations, and process improvement," *IEEE Software*, vol. 11, no. 4, pp. 36–45, 1994.

[12] G. O. H. B. J. O. J. H. J.D. Herbsleb, H. Kein, "Object-oriented analysis and design in software project team," *Human-Computer Interactions*, vol. 10, no. 2, pp. 249–292, 1995.

[13] S. Ehrlich, "Strategies for encurraging sucessful adoption of office communication system," *ACM Transactions on Office Information Systems*, vol. 5, pp. 340–357, 1987.

[14] J. Grudin, "Why groupware application fail: Problems in design and evaluation," *Office: Technology and People*, vol. 4, no. 3, pp. 245–264, 1989.