Communication Problems in Global Software Development: Spotlight on a New Field of Investigation

Sébastien Cherry, Pierre N. Robillard
Software Engineering Research Laboratory, École Polytechnique de Montréal
{sebastien.cherry, pierre-n.robillard } @ polymtl.ca

Abstract

While it is widely recognized that communication plays a critical role in software development, it has been observed that problems of coordination may be generated when teammates in the field are working at a distance from one another.

This paper presents an ongoing empirical study on ad hoc collaborative activities which occur in an industrial software engineering environment. We believe that a better understanding of these types of activities and their content will pave the way to further solutions designed to enhance communication, and thus improve both collaboration and coordination in virtual software development settings.

We include details of our motivations for the study, followed by some methodological considerations, and, finally, some preliminary results which demonstrate not only the significance of our data, but also the relevance of our approach.

1. Introduction

Communication is undoubtedly an essential element which plays a critical role in a software engineering process in the gathering and crystallization [15] of all relevant information in quality software which fulfills the user needs on time and within budget [2], [3], [13], [14], [16]. Moreover, several studies have stressed the fact that informal communications seem to be fairly important in terms of the time spent on a software project. Perry, Staudenmayer and Votta (1994) found during a case study that informal communications take up an average of 75 minutes per day per software developer [12], and Robillard and Robillard established in another case study that ad hoc collaborative activities can occupy up to 41% of the developer's time [14]. Seaman (1996) [16] also supports the need for this type of communication if developers are to carry out their tasks adequately.

However, in the global software development setting, which has become a more common practice for many business reasons, some researchers have observed that communications, specifically informal ones, face significant challenges by virtue of distance, both geographical and temporal [6], [7], [8], [9]. They note that the consequence of this is a potential for problems of coordination to occur.

This short paper presents an ongoing empirical study being carried out within the framework of a case study in the industry which explores the ad hoc collaborative activities that take place in a software engineering setting. By ad hoc collaboration, we mean all informal and spontaneous activities performed by two or more developers who are working on a particular task of the project. These activities can take many forms, such as in informal peer-to-peer conversations, also referred to in the literature as "water-cooler-talk" [6], as well as electronic mail exchanges and phone calls.

We believe that a better understanding of such activities and their content will pave the way for further solutions with the aim of enhancing communication and thus improving both collaboration and coordination in virtual software development settings. Our general approach is to observe and understand the informal and spontaneous collaboration activities that take place in a classical single-site software development environment where developers have as much freedom and opportunity to communicate as they wish, and to measure their positive and negative impacts on the rest of the software project. Depending on the results of our investigation, two avenues of action can be envisaged. The first might be to infer and formalize from our observations some state-of-the-art rules or practices applicable in global software development contexts which will be better adapted to the empirical reality and make collaboration between teammates working apart more efficient. The second might be to use this comprehension to give us some insight into the tools needed to support communications in a distributed software development environment with the intention of creating what some call the "virtual 30 meters" [9].

In this paper, we explain our research methodology in broad strokes and present some preliminary results which demonstrate not only the significance of our data, but also the relevance of our approach.

2. Research Methodology

Empirical studies have been becoming more and more popular in the past few years in software engineering, in parallel with the growing popularity of people-centered researches. Indeed, researchers must innovate in order to study this new topic of interest, namely people, by borrowing certain techniques used in the social sciences, such as psychology and sociology. Our research methodology has been inspired by several papers [4], [11], [18], [19] as well as aforementioned studies which have examined the human aspects of software engineering, but has been to some extent adapted to the field investigated in this study. Below, we give a general description of our research methodology.

2.1. Research Objectives

As stated above, good communication is the *sine qua non* of software development processes in obtaining quality products which meet user needs on time and within budget [2], [3], [13], [14], [16]. Research has shown the non-negligible importance of informal communications [12], [14], [16], while some has specifically highlighted the fact that distance in global software development is a challenge to informal communications which can generate problems of coordination [6], [7], [8], [9]. However, no research has described the content of this type of communication. These elements led us to define the following research objectives:

- To design a model of the ad hoc collaborative activities found in an industrial software engineering setting and characterize them, as well as to identify and describe the content of the communications that ensue.
- To generate a series of hypotheses emerging from the results of this research which could later be validated by confirmatory research.

2.2. General Approach

These objectives will be achieved by means of participant observation. This technique is suitable in exploratory research like ours where the goal is to inductively generate theories from direct observations, also called grounded theory, rather than to empirically verify a hypothesis formulated in advance [1], [10], [17].

2.3. Target-setting

The target is a team of eight individuals which develops software for commercial purposes in a large international company which has been in operation for several years and which has a well-established software development process. It should be noted that, even though the observations are made in a large company, this last contains attributes of small or medium-sized organizations since the work is divided among small teams like the one that is participating in our research. Also, the members of this team are highly representative of developers in the industry, in terms of the wide variety of their ages, software development experience, schooling and length of service in the company.

2.4. Data Collection

The data collection phase, which lasted 8 weeks, is done. The methods used during this phase were selected following an earlier ethnography period of several months. The data collected during this period includes:

- 185 hours of audio-video recordings of working sessions
- 2496 electronic mails exchanged by the 8 teammates
- A daily backup of the source code and other artifacts found in the field

Audio-video recordings were preferred over field notes because they offer the enormous advantage that they can be consulted many times over. This is very important in exploratory research like ours where we do not necessarily know in advance what to observe. E-mails were automatically captured by triggers defined in the messaging software used in the company, and include both those received and those sent to allow cross-validation. Finally, a backup was made of the source code and artifacts found to allow further content analysis.

2.5. Data Analysis

The principal method used to analyze the large amount of data, mainly in the form of the audio-video recordings, was the Exploratory Sequential Data Analysis (ESDA) [5]. This method was chosen because it is particularly well-suited to exploratory research like ours, where theories have to be induced from empirical data, and, more importantly, where the sequential information of the data must be preserved.

Briefly, the ESDA process is iterative. It involves the definition, sometimes intuitive, of concepts arising from the ESDA tradition of taking the point of view of the researcher, subsequently providing a guide as to what to observe in the raw data and how to manipulate it to derive data on which theories will be founded. This process is done iteratively because it is often necessary to step back in order to add, remove or revise some concept definitions [5].

Of the eight different ways to manipulate data proposed by ESDA, encoding is certainly the most

important. This consists of labeling each data sequence with a code formed from an exhaustive, exclusive and limited list of categories. This allows qualitative data to be transformed into quantitative data, which in turn makes it possible to further manipulate the data, by performing statistical calculations, for example [5], [17].

3. Preliminary Results

The following results are based on observations made on the activities of four of the eight developers on the team over a period of 8 hours each. It does not take into account e-mail exchanges. Also, the four individuals observed were chosen because they have been seen to collaborate more with their teammates than the others. This choice is justified because our purpose is to study the content of the ad hoc collaborative activities that occurred, so that these particular individuals were simply more likely to give us more data to analyze.

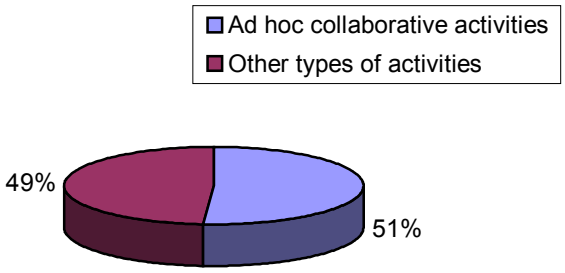

Figure 1. Distribution of time spent in ad hoc collaborative activities in comparison with other types of activities

As Figure 1 illustrates, 51% of the time spent on the project by the observed developers is occupied on ad hoc collaborative activities, in contrast with other types of activities. This is a surprising result which seems to strongly confirm the importance of the phenomenon observed in the target setting, but which needs to be validated by further analysis on a much larger scale.

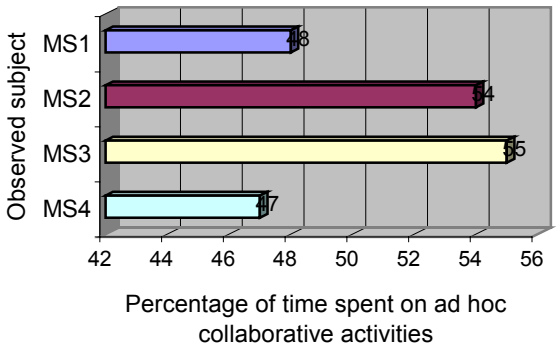

Figure 2. Percentage of time spent on ad hoc collaborative activities by observed subject

Figure 2 shows the time spent on ad hoc collaborative activities per observed subject. It can be noted that the percentages associated with subjects MS2 and MS3 are slightly higher than the others. This difference can possibly be explained by the nature of the work performed by these subjects since MS2 is the team's project manager and MS3 is in charge of the infrastructure for the software built and often the problem-solver on the team. An interesting thing to note is that, in 78% of the interactions in which MS2 is involved, his interlocutors initiated the interactions. However, it has been noted in the field that, most of the time, MS2 shares information by broadcasting a message to his team via e-mail instead of in peer-to-peer conversations. Thus, it will be interesting to analyze these e-mail exchanges. By the way, in each of the 82 interactions observed, an average of 2.3 stakeholders took part, and their average duration was 6:31 minutes.

As for the ad hoc collaboration activities observed, a preliminary outline has emerged from the raw data containing six categories, as follows: "cognitive synchronization" exists when two or more developers exchange information to ensure that they share the same knowledge or the same representation of the object in question; "problem resolution" occurs when two or more developers are aware of the existence of a problem and attempt by various means to solve the problem or to mitigate it; "development" occurs when two or more developers contribute to the development of a new feature or component of the software; "management" is the result of two or more developers coordinating and planning activities such as meetings, common working sessions or scheduling; and "conflict resolution" is the process of two or more developers taking part in discussions to resolve a difference of opinion. Ad hoc collaborative activities in the "not relevant" category, group together all the interactions that do not concern the project or the software built.

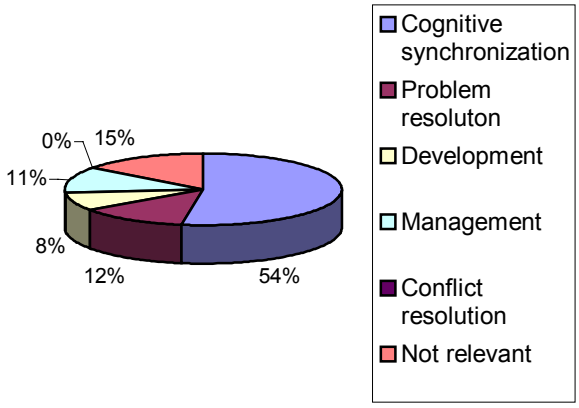

Figure 3. Distribution in number of occurrences of ad hoc collaborative activities identified

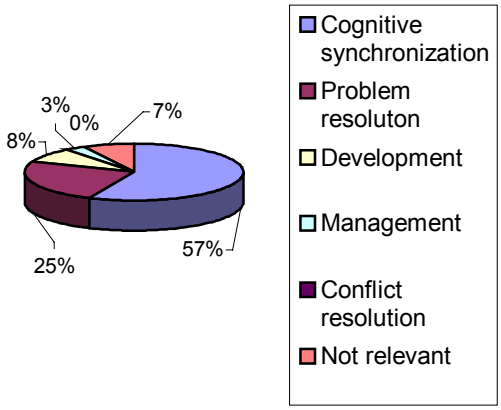

Figure 4. Distribution in terms of time spent on ad hoc collaborative activities identified

Figure 3 shows that, in a little over half the times when these interactions occur, they do so in the form of cognitive synchronization, and this tendency is supported by the data in Figure 4 which show the distribution in time spent. This is not a surprising finding, since it is well established that the sharing of information and knowledge is a crucial element in software development.

Moreover, it can be noted that problem resolution activities represent only 13% of the occurrences, but that, in terms of time spent, the percentage rises to 25%. This suggests that, when they occur, problem resolution activities take much longer than the others. This is supported by the statistics of time spent by sequence as function of ad hoc collaborative activities, which shows that a mean of 9:48 minutes is spent on problem resolution.

Finally, another interesting finding is that management activities, unlike problem resolution, represent 13% of the occurrences, but only 3% in terms of time spent. In other words, they are quite numerous relative to the small percentage of time spent on them. This result probably tends to support the theory of some researchers to the effect that informal communications are important in order that teammates can coordinate their activities efficiently [6], [7], [8], [9].

4. Conclusion

It is widely held that communication is a crucial element in software engineering, but, unfortunately, it is an aspect which seems to be lacking in global software development and one which must be addressed.

This paper presents an ongoing empirical study on ad hoc collaborative activities in an industrial software engineering setting. The general objective of this research is to gain a better understanding of these kinds of activities and their content in order to be able, subsequently, to propose software process enhancements with the aim of rendering collaboration between teammates more effective on the one hand, and, on the other, to obtain some insight into the tools needed to support communications in a distributed software development context.

We think that this kind of research is needed, first of all, because the importance of communication in distributed environments is poorly understood, but also to expose any wrong assumptions there may be that are often mistaken for the truth in software engineering.

Even if further analysis are to be done, the few preliminary results that were partially presented in this contribution tend to demonstrate that a data model and certain patterns are emerging from the vast quantity of data amassed turning the spotlight on a new facet of the empirical reality of software engineering which until today was completely hidden.

5. Acknowledgments

This research would not have been possible without the agreement of the company in which it was conducted, and without the generous participation and patience of the software development team members from whom the data was collected. To all these people, we extend our grateful thanks.

6. References

[1] Babbie, E., *The Practice of Social Research, 9th edition*, International Thomson Publishing Company, 2001.

[2] P. D'Astous, and P.N. Robillard, "Empirical Study of Exchange Patterns during Software Peer Review Meetings", *Information and Software Technology*, 44, 2002, pp. 639-648.

[3] P. D'Astous, P.N. Robillard, F. Détienne and W. Visser, "Quantitative Measurements of the Influence of Participant Roles during Peer Review Meetings", *Empirical Software Engineering*, 6, 2001, pp. 143-159.

[4] Fenton, N.E. and S.L. Pfleeger, *Software Metrics - A Rigorous & Practical Approach*, PWS Publishing Company, 1997.

[5] C. Fisher and P. Sanderson, "Exploratory Sequential Data Analysis: Exploring Continuous Observational Data"*, Interactions*, 3:2, Mar. 1996, pp. 25-34.

[6] R.E. Grinter, J.D. Herbsleb, and D.E. Perry, "The geography of coordination: Dealing with distance in R&D work", *GROUP'99: International Conference on Supporting Group Work*, Coordination and Negotiation, 1999, p. 306-315.

[7] J.D. Herbsleb and R.E. Grinter, "Splitting the Organization and Integrating the Code: Conway's Law Revisited", *Proceedings, International Conference on Software Engineering*, Los Angeles, CA, 1999, pp. 85-95.

[8] J.D. Herbsleb and D. Moitra, "Guest Editors' introduction: Global software development", *IEEE Software*, 18:2, March/April 2001, pp. 16-20.

[9] J.D. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally-distributed software development", *IEEE Transactions on Software Engineering*, 29:6, June 2003, pp. 481-494.

[10] D.L. Jorgensen, *Participant Observation A Methodology for Human Studies*, Applied Social Research Methods Series; v.15, Sage Publications, 1989.

[11] B. Kitchenham and al., "Preliminary guidelines for empirical research in software engineering", *IEEE Transactions on Software Engineering*, 28:8, 2002, pp. 721–734.

[12] D.E. Perry, N. Staudenmayer and L.G. Votta, "People, Organizations, and Process Improvement", *IEEE Software*, July 1994.

[13] P.N. Robillard, "The Role of Knowledge in Software", *Communications of the ACM*, 42:1, 1999, pp. 87-92.

[14] P.N. Robillard, and M.P. Robillard, "Types of Collaborative Work in Software Engineering", *The Journal of System and Software*, 53, 2000, pp. 219-224.

[15] P. N. Robillard, P. Kruchten and P. D'Astous, *Software Engineering Process with the UPEDU*, Addison Wesley, Pearson Education, 2002.

[16] C. Seaman, *Organizational Issues in Software Development: An Empirical Study of Communication*, Ph.D. Thesis, Computer Science Department, University of Maryland, Technical Report CS-TR-3726, UMIACS Technical Report UMIACS-TR-96-94, 1996.

[17] C. Seaman. "Qualitative methods in empirical studies of software engineering", *IEEE Transactions on Software Engineering*, 25:4, 1999, pp. 557–572.

[18] W. Tichy, "Should computer scientists experiment more?", *Computer*, 31:5, 1998, pp. 32–40.

[19] R.J. Walker, L.C. Briand, D. Notkin, C.B. Seaman, W.F. Tichy, Panel: "Empirical Validation-What, Why, When, and How", *Proceedings, International Conference on Software Engineering*, Portland, Oregon, 2003, pp. 721-722.