Journal of Universal Computer Science, vol. 17, no. 14 (2011), 1913-1930
submitted: 29/3/11, accepted: 28/9/11, appeared: 1/10/11 © *J.UCS*

Collaboration, Information Seeking and Communication: An Observational Study of Software Developers' Work Practices

Márcio Kuroki Gonçalves
(Federal University of Pará, Pará, Brazil
marciokuroki@gmail.com)

Cleidson R. B. de Souza
(IBM Research Brazil, São Paulo, Brazil
cleidson.desouza@acm.org)

Víctor M. González
(Instituto Tecnológico Autónomo de México, México City, México
victor.gonzalez@itam.mx)

Abstract: Different aspects defining the nature of software engineering work have been analyzed by empirical studies conducted in the last 30 years. However, in recent years, many changes have occurred in the context of software development that impact the way people collaborate, communicate with each other, manage the development process and search for information to create solutions and solve problems. For instance, the generalized adoption of asynchronous and synchronous communication technologies as well as the adoption of quality models to evaluate the work being conducted are some aspects that define modern software development scenarios. Despite this new context, much of the research in the collaborative aspects of software design is based on research that does not reflect these new work environments. Thus, a more up-to-date understanding of the nature of software engineering work with regards to collaboration, information seeking and communication is necessary. The goal of this paper is to present findings of an observational study to understand those aspects. We found that our informants spend 45% of their time collaborating with their colleagues; information seeking consumes 31,90% of developers' time; and low usage of software process tools is observed (9,35%). Our results also indicate a low usage of e-mail as a communication tool (~1% of the total time spent on collaborative activities), and software developers, of their total time on communication efforts, spending 15% of it looking for information, that helps them to be aware of their colleagues' work, share knowledge, and manage dependencies between their activities. Our results can be used to inform the design of collaborative software development tools as well as to improve team management practices.

Keywords: Observational Study, Software Engineers, CSCW, Multi-tasking, Collaboration, Awareness
Categories: D.2.2, H.5.2, H.5.3

1 Introduction

More than 30 years ago, Brooks [Brooks, 74] defined software development as "a complex interpersonal exercise". This complexity is evident in what is known as

Brooks' Law: "adding manpower to a late software project makes it late." The rationale for such law is that the increase in the communication and coordination activities required to integrate the new team members. Since this seminal work, several other studies have been conducted to investigate the nuances of software development as a cooperative activity and the influence of human and cooperative aspects in the productivity of software development activities. For instance, seminal work by Curtis and colleagues suggests that communication and coordination issues constitute one of the three main problems in software development [Curtis, 88]. Additional studies have helped us to understand other cooperative aspects of software development. It has been found that formal and informal communications take more than 50% of software engineers' time [Perry, 94] and collaborative activities demand up to 70% of their time [Vessey, 95]. Perlow has reported that software engineers spend 30% of their time on interactive activities and 60% working alone [Perlow, 99]. More recently, others have investigated how the communication and coordination might impact software development productivity [Cataldo, 06; Cataldo, 08].

It is interesting to observe that most, if not all, empirical studies are performed in North American and European organizations. Given the current trend towards outsourcing software development efforts [Herbsleb, 01][Damian, 06], it is necessary to study how work is performed in other countries. Therefore, this paper describes the results of an empirical study conducted in a large software development Brazilian organization. The study aims to characterize how software engineers spend their time in a typical workday, and it also explores whether the usage of information technologies and the adoption of software process models have influenced their individual and collaborative activities. By understanding the day-to-day software development activities we can, for example, inform the design of tools to properly support developers' activities.

Our study was conducted in a large software development organization and data was collected using non-participant observation [Jorgensen, 89] and semi-structured and unstructured interviews [McCracken, 88]. More specifically, we conducted interviews and performed direct observation of software developers' work as they performed it, from the beginning to the end of a workday. Our observation data was analyzed and generated tentative explanations about their work that were later tested in follow-up interviews with the software developers themselves. We discuss our results in the context of previous research on the area.

The rest of this paper is organized as follows. Section 2 presents some related work, while Section 3 describes the site studied and the research methods adopted. In Section 4, our initial findings are described, and they are compared with previous studies in Section 5. Finally, Section 6 presents our conclusions and future work.

2 Related Work

The cooperative and human aspects of software development, hereafter called CHASE [de Souza, 09] have been discussed for at least 30 years and includes seminal work from Brooks [Brooks, 74], Weinberg [Weinberg, 71], and De Marco [De Marco, 99]. Brooks presented anecdotal evidence that CHASE aspects played a very important role in software development. Later on, empirical evidence confirmed the collaborative nature of the process of designing and building software systems [de

Souza, 09]. For instance, Curtis and colleagues [Curtis, 88], identified communication and coordination issues as one of the main problems in software development. Later on, Perry and colleagues [Perry, 94] investigated how software development activities were influenced by technology. To achieve this goal, they used direct observation and a time-diary to document what software developers did in their daily work. Their results indicate that (i) formal and informal communication encompass over 50% of a software developer's time, and (ii) much of this interaction among developers aims to allow them to be aware of their colleagues' work. Software developers also communicate when searching for all kinds of information including understanding the code, reasoning about the design, maintaining awareness of each others' activities, and others [Hertzum, 02] [Perlow, 99]. Ko, DeLine and Venolia [Ko, 07] sought to understand which kind of information is searched by developers and why they looked for this type of information. Their results suggest that the most sought after information includes whether any mistakes were made in the code and what the other developers were doing. This is consistent with previous studies about software developers' need to be aware of their colleagues' activities [Whittaker, 99][de Souza, 04].

A consequence of all this communication and information seeking during software developers' workday is *work fragmentation*. González and Mark [González, 04] observed that developers spend on average about 3 minutes in one particular task and 12 minutes in a project[1] before changing to a different task or project. They also observed that software developers spend longer periods of uninterrupted work on their personal computers when compared to other roles such as managers, which is indicative of the need for sustained concentration when doing certain work activities (e.g., programming). Ko, DeLine and Venolia [Ko, 07] reported a similar average of 5 minutes per task.

In a more recent study, Cherry and Robillard [Cherry, 09] studied communication and collaboration in a small and collocated team. They concluded that face-to-face communication, in this context, is far more used than other available media due to the collocation of developers. In fact, they suggest that an expert should be located in the most accessible position so that all other developers can have easy access to him, i.e., based on their observation of communication patterns in the team, they make suggestions about the physical layout of the development environment.

In general, all these studies indicate the importance of communication among software developers in order to understand the code and the design of the software, share knowledge, keep them aware of their colleagues, and so on. At the same time, they point to the fragmentation of software developers' work because they interrupt their colleagues and/or are interrupted by them when seeking information for their work.

While clearly relevant and insightful, these studies have some limitations worth noticing. First, they did not study the communication infrastructure available for software developers in detail, nor the software process models followed by these developers. These aspects are relevant because of the many changes that have happened recently in organizations: notably, collaborative applications such as instant messaging were introduced into the work environment [Handel, 02] and are fairly

[1] González and Mark call it a *working sphere*.

pervasive today, and software process models and tools were adopted motivated by the importance of quality models such as CMMI [Ahern, 03]. More importantly, these empirical studies were performed in North American and European organizations. Given the current trend towards outsourcing software development efforts [Herbsleb, 01][Damian, 06], it is necessary to study how work is performed in other countries. Given Brazil's growing importance in the outsourcing business due to its geographical location [Carmel, 10], understanding the work of *Brazilian* software developers is crucial.

This paper describes our research aiming to characterize how software engineers collaborate, interact and communicate with others i.e., how they divide their time in a work setting, what kind of information they usually seek, and which tools they use to support their work. This study considers the context of a midsize team and has been performed *in situ* in a large Brazilian software development organization. Our research methods are described in the following section.

3 Research Methods

3.1 Introduction

In order to precisely characterize software developers' workday, we adopted a mixed quantitative and qualitative research approach [McGrath, 95]. A quantitative component was necessary to collect and analyze data (e.g., time) about each software developer's activity, while a qualitative strategy was adopted to allow us to focus on the actual activities and the reasons behind why work was conducted the way it was. Our approach, as it is the case of many previous observational studies of software developers, does not aim at investigating a large number of instances of the phenomenon, but instead to focus on a particular instance that provides results that are richer and more informative than quantitative methods [Seaman, 99], and that serve to better highlight the situated nature of software development practices. Based on these results, we identify findings that are particular to the group studied, but nevertheless constitute grounded evidence that defines the phenomenon in general. Findings from empirical studies like the one presented here make progress in the understanding of the phenomenon and serve to define hypotheses that can be tested in large populations (for instance, different organizations) or other contexts. This clearly, is the typical cycle of research contribution, which is usually defined by a constant alternative effort between deductive and inductive research.

3.2 Organizational Context

This study was performed in an organization called Sigma (not the real name). This organization is certified at CMMI level 2 and is located in Belém, Brazil. Sigma has its own software development process, which is specified through macro-activities. This software process is based on RUP (Rational Unified Process) [IBM, 09] and PMBoK (Project Management Body of Knowledge) [PMI, 04], and is adapted to match CMMI's best practices.

The team studied was part of one the software development branches of Sigma and was composed of 41 software engineers divided in three sub-teams namely team

A, B and C. Software engineers were divided as follows: (a) one project manager, responsible for allocating projects to development sub-teams; (b) three project leaders, each one responsible for managing one development sub-team and interacting with customers; and (c) 37 developers, responsible for requirements, development, testing and software deployment. Each sub-team was composed of 12 or 13 members and worked on a single project at a time for the same customer, another Brazilian government agency. Besides the project manager, who worked in a separate office, all other software developers worked in an open area, divided by tables and low stalls. This environment allowed developers who were physically close to each other to easily see and talk to each other while working on their computers.

Members of the team needed to communicate with 3 sites located in different places of Brazil: customers were located in Brasília, the team responsible for maintaining the database was located in Curitiba, and, finally, the database administrator was in Rio de Janeiro. All these cities are located in the same time zone and are about 1-hour apart by plane from each other, and at least, 2-hour apart from Belém, where the developers were located. When necessary, there were conference calls to discuss problems, redefinitions of scope, updates and adaptations of the project schedule.

We observed software developers from two different sub-teams [Jorgensen, 89]. The first sub-team observed was Sub-Team A. It was composed of 13 people: one project leader, two technicians (administrative staff), and ten software developers. In this case, we collected data from 8 different software developers. The second sub-team observed was Sub-Team B, it was composed of 12 people: one project leader, two technicians and nine software developers. Six software developers provided data from sub-team B. So, in general we collected data from 14 different software developers. We also collected data in different work shifts to make sure we covered different patterns of work.

Both sub-teams used many software tools to accomplish their work. They had a version control repository in which were included both: process documentation (meetings reports, manuals and use case descriptions, database models, etc) and versions of the software under development (source code, database schema, etc). These teams also used a bug-tracking system for testing and change approval (or delivery).

In general, despite the subdivision in three different sub-teams, the team we observed had the same customer, employed the same tools, followed the same software process, and shared the same physical environment. Therefore, for the purposes of this study we do not make any distinction and do not compare the results from the two sub-teams. From now on, we refer to both groups as “the team”.

3.3 Data Collection and Data Analysis

Initially, general observations were conducted to learn about how people behaved in their work setting. This period was used to get developers and the observer familiarized with each other’s presence. In the first observations, the observer attended team weekly meetings. After that, the observer started to perform individual observations: the first observed person was the team leader. This choice was motivated by two reasons: (1) she was one of our first contacts in the company, therefore, she knew our research goals; and (2) as a team leader, she could be taken as

an example to motivate other developers. After that, the team leader and other software developers were observed.

Interviews [McCracken, 88] and observations [Jorgensen, 89] were performed by the first author. He performed an interview before every individual observation in order to understand which tasks the developer would perform during the observation period. In addition, the observer explained how the observation would be carried out, reinforcing aspects about data confidentiality and non-intrusion. These meetings lasted about 15 minutes. Having a watch and a notepad, the observer took notes of every activity, tool and person that interacted with the observed developer. For instance, collected information included how long (in minutes and seconds) the developer's activity lasted, who the observer interacted with, which tools (s)he used, etc. When the observed developer changed tasks, the new activity performed was recorded as well as the tools used and people with whom the developer interacted.

More formally, we recorded the following data for each activity: duration, type of activity and who eventually participated in the activity. When the observation session was finished for each developer, we conducted another interview to ask clarifying questions to the observed developer; for instance, the reason for a particular dialog, or the use of an artifact or tool. An example of an observation log is presented on Table 1 below.

10:23:05: Talks to "Maria" about his tasks. 10:24:00: Back to his PC and starts to work in the IDE. So, calls "Maria" to show his progress in the activity (Implementation of a prototype). 10:25:00: Maria left. And he stops the implementation to look at his email. 10:26:35: Returns to the IDE. He continues the implementation. 10:35:00: "João" interrupts him to assess the progress of his activities. 10:38:00: He uses the Control Version System to checkout some documentation about the project, to help him in the task. 10:44:15: He takes a break. (Drinks water, goes to toilet or gets a cup of coffee). 10:49:10: Returns to his table. Implementation phase. 10:56:07: "Maria" interrupts him to ask questions about the implementation.

Table 1: Example of Observation Log

Finally, observations as well as interview notes were transferred to a computer where we performed our analysis (see Section 4). We obtained about 51 hours of observation data, not including the time spent on weekly meetings. As mentioned before, we were able to observe a total of 14 software developers ranging from 1 to 3 observed work shifts by person.

We aimed at identifying typical practices in software engineers' usual working day at Sigma. Thus, we performed our analysis in phases, described as follows. First of all, the actions performed by software engineers were classified and quantified in electronic spreadsheets. Then, all the data were organized in such a way that we could calculate the time spent in each action performed during the observation period. After that, actions were qualitatively analyzed, i.e., in this phase, we grouped the data in

such a way that we were able to quantify the time spent on actions with the same purpose, for instance, two developers talking about implementation issues (personal conversation to solve codification issues). Lastly, all data gathered from the observations of all engineers were aggregated and resulted in the statistics presented in the next section. We conducted two follow-up interviews with members of the team to validate some of the results we report here.

4 Results

4.1 Collaborative activities vs. individual activities

Following previous studies, our first step was to characterize software engineers' workday by classifying their activities as either individual or collaborative. Individual activities do not require other developers' participation (e.g., coding) whereas collaborative activities require more than one person to be involved (e.g., a meeting). A collaborative activity is any activity that requires interaction with another person, for instance, informal meetings, hallway conversations, phone calls, and e-mail messages. In general, the aggregate time *all* observed software engineers spent in collaborative activities was about 45% of their time, while for individual activities it consisted of 40% of their time. We decided to label e-mail as a collaborative activity because activities related to email are aimed at another person, in contrast to coding, or writing requirements specifications that are individual activities, but without a straightforward "target". Other activities include breaks, bathroom usage and so on and account for about 15% of software developers' total time per observation.

Table 2 presents information about the time spent, in minutes, in individual and collaborative activities: we include the average, standard deviation, maximum and minimum values per 4-hour shift for all observed software developers.

	Individual Activities	Collaborative Activities
Average	00:49:26	00:54:32
Standard Deviation	2:04	3:25
Maximum	01:58:06	02:40:00
Minimum	00:00:00	00:00:00

Table 2: Descriptive Statistics for Individual and Collaborative Activities per 4-hour shift

According to Table 2, it is possible to observe a great difference between maximum and minimum values in both collaborative and individual activities. Maximum collaborative values were observed with software developers in three different situations:

- The developer had very good skills with a particular technology which was being used by the team; or

- He had knowledge about the customer, i.e., the business rules which were being implemented; or
- Finally, the developer was a newcomer in the team and was learning to do the work.

In the first two cases, developers will engage in collaborative activities because they will be helping their teammates. In contrast, in the third situation, a newcomer will be engaged in collaborative activities because he will be being helped by his teammates.

Minimum values also indicate important aspects of work. Zero minutes for individual activities basically meant that the developer spent an entire work shift in meetings and/or gathering (providing) information from (for) others so that he (the others) could perform his (their) own work. Meanwhile, zero minutes for collaborative activities, and consequently, high values for individual activities occurred when a developer had to implement a particular feature and had no difficulty doing so. In summary, these figures help us to understand with more precision the different scenarios of situated activity as software developers, depending on their schedules, phase of the project, and particular activity being performed, might require more or less collaboration. In some days Sigma software developers would work independently, while in other days they will be fully engaged with their colleagues.

Individual and collaborative activities were further classified according to the tool or media used by software developers. This can be observed on Figures 1 and 2. Figure 1, describes that much of the collaborative activities, 75% of them, takes place as face-to-face meetings, followed by much less time spent on the phone or communicating by Instant Messaging. Individual activities (Figure 2), on the other hand, were spread across a myriad of tools, but mainly digital ones, including word processors (34%), the development environment (IDE, the database systems, etc - 33%), the version control system (13%), and the application under development itself (11%).

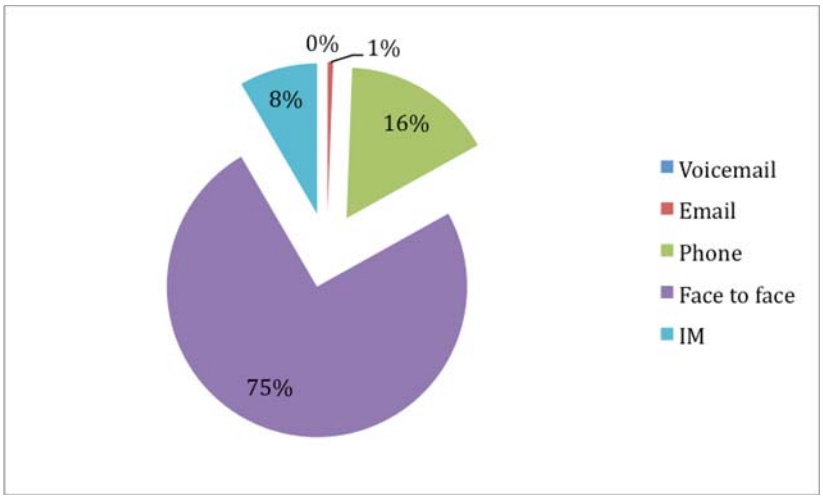

Figure 1: Collaborative Activities detailed

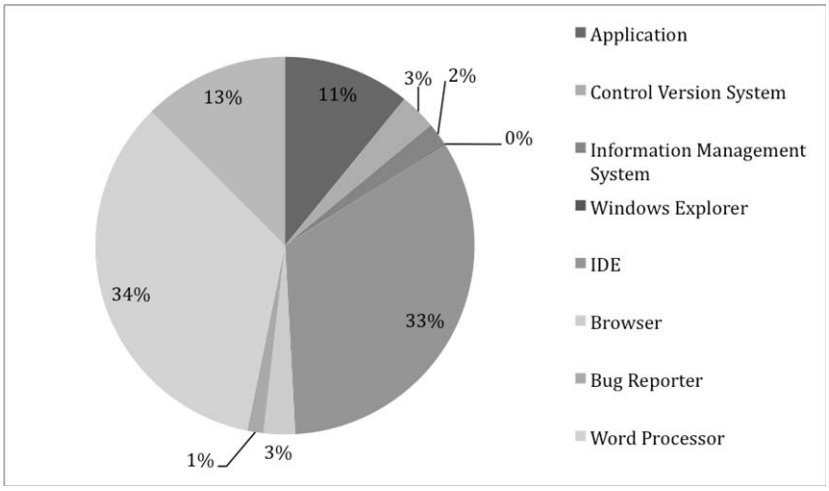

Figure 2: Individual Activities detailed

4.2 The interwoven of collaborative and individual activities

One interesting aspect that we identified in our data is that software engineers' individual and collaborative activities are interwoven during their daily work. This is illustrated in Figure 3 that presents the results obtained from a software engineer's observation. Positive values indicate individual activities while negative values correspond to collaborative activities. Positive or negative values indicate how much time (in seconds) a developer spent in a particular (individual or collaborative) activity, but have no semantic meaning.

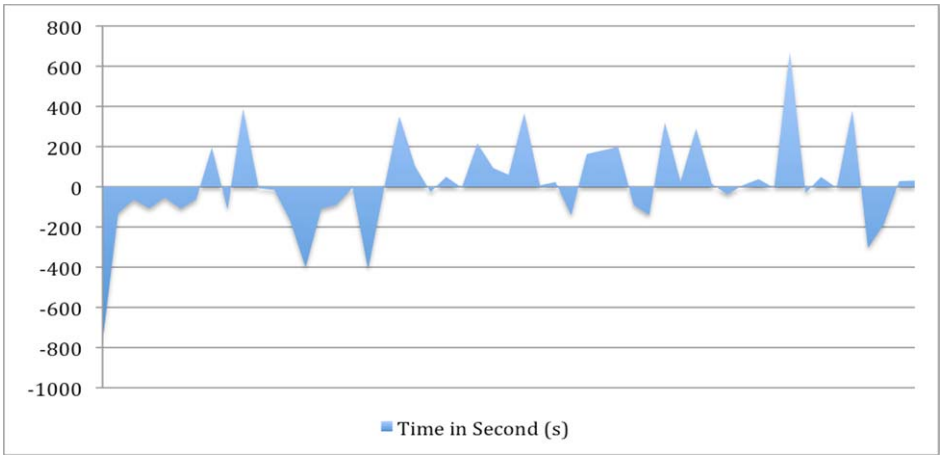

Figure 3: The interwoven of collaborative and individual activities

The data presented in Figure 3 belongs to a developer who was responsible for implementing a particular customer feature. During the data collection, he was implementing the database queries required by his task, while other engineers interrupted him often to ask for help with their own implementation tasks. This happened because this software engineer was an expert in the business rules and technologies used in this project. After helping their colleagues (a collaborative activity), he returned to his (individual) activity. A similar analysis was performed for all observed software developers and the results suggest a similar pattern, i.e., individual and collaborative activities were interwoven.

In addition, we differentiated self-interruptions and external interruptions [González, 04]. A self-interruption happens when a software developer decides to stop his work to engage in a collaborative activity (email, IM, etc), while an external interruption occurs when a software developer stops his current activity because of a collaborative activity initiated by someone else. By doing that, we were able to calculate the average time in which a software developer works without being interrupted: 3,07 minutes, with a standard deviation of 3,10 minutes. Interruptions themselves lasted on average 1,09 minutes with a standard deviation of 0,96 minutes or 57,73 seconds.

4.3 On the usage of software process tools and communication

In the organization we studied, software engineers use tools to help them define and execute their activities, which are based on the software development process adopted by the organization. Examples of these tools include: e-mail; bug tracking; the internal information management system used by all team members; and a version control system. We observed that software developers used tools to:

- Understand activities executed by other Sigma software engineers;
- Identify the next activity they needed to perform; and
- Identify who they needed to interact with in order to find the solution for a certain problem.

In addition to software tools, Sigma developers used additional means to be informed about their next activities in the software development process, or about the current state of their colleagues' activities. For instance, face-to-face communication, phone calls, and instant messages were the three most frequent ways noticed during our observation. More specifically as described in Figure 1, they accounted respectively for 75%, 16% and 8% of software developers' time spent in *collaborative* activities. Now, if we take into account the *total* observation time of a software developer, these values will be 33,75%[2] for face-to-face conversations, 7,2% for phone conversations, and 3,6% for instant messages.

Software engineers used these means to look for information, pass it on, notify colleagues about important changes, disclose the status of their own activities, and attend meetings (either face-to-face or through conference calls). All these activities account for around 32.5% of a software engineer's time: 9.35% due to software process tools and 23.15% due to communication about the software process. This

[2] In this case, it is 75% of face-to-face meetings out of the 45% of collaborative activities.

value, 32.5%, is divided as described in Figure 4 below. Note that some of the categories described here are the same as described in Figures 1 and 2. However, in this case, we are focusing on how these tools or communication media were used regarding the software process used.

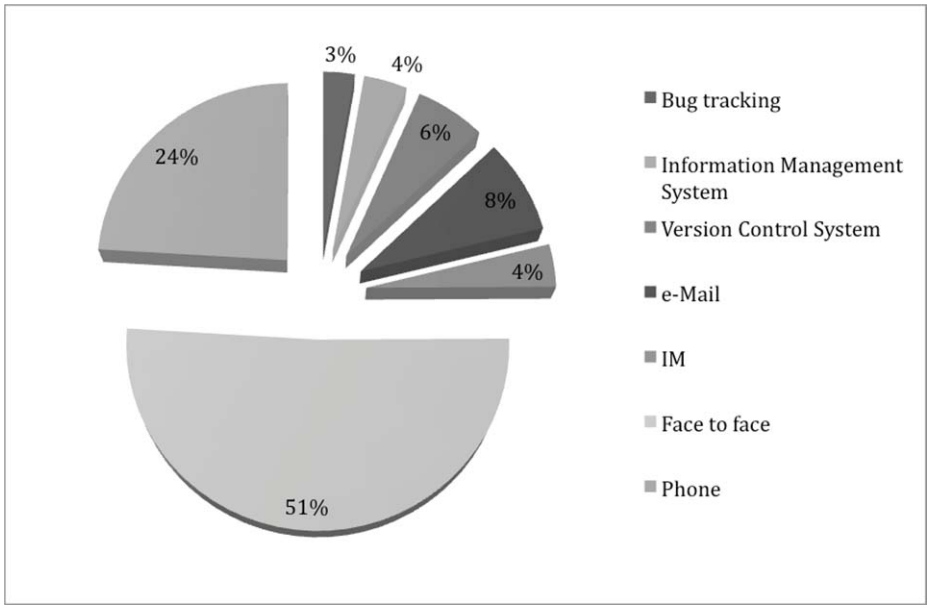

Figure 4: Tools and communication media used by software engineers regarding the software development process

4.4 Information Seeking

Information seeking is another important aspect that has arisen out of the data. According to our analysis, Sigma software developers spend 31,90% of their time seeking some type of information. Figure 5 details this percentage according to the type of information sought out.

Based on the figure, it is possible to notice that questions about implementation and business rules combined are around 40% of the total time spent in information seeking activities, while 45% of the time is spent sharing information about *executed or finished* activities (for example, activity status – 14%) or activities *to be performed* (as in information sharing – 9%). Checking email (3%) was considered an information seeking activity because it is the most used way by which the team leader schedules meetings and informs the development team about the decisions taken in a meeting, i.e., meeting reports are sent by email. This means that Sigma software engineers access e-mail many times a day to check when new messages arrive because they may contain important information about their current or future work.

Figure 5 also indicates that Sigma software engineers spend 45% of their total time per observation with communication-related activities to search information or, more specifically, about 15% with actions which help developers to (a) keep

themselves aware of their colleagues' work, (b) share knowledge, and, finally, (c) manage dependencies between their activities. As discussed in section 3.2, Sigma software engineers are co-located, thus the situation is favorable for interaction and awareness of work colleagues' activities.

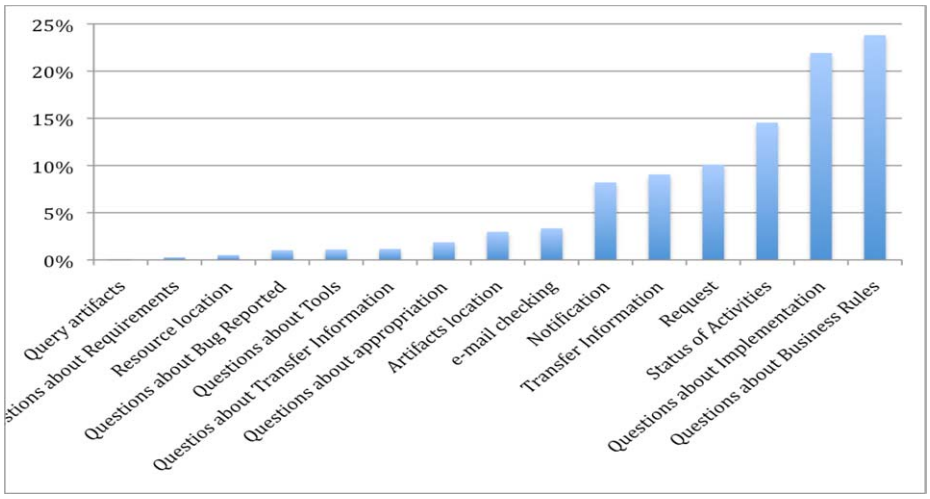

Figure 5: Percentage of time spent in seeking information

5 Discussion

Recent technological advances allowed new communication technologies to become popular in the workplace. For instance, Handel and Herbsleb [Handel, 02] discuss how instant messaging (IM) is more and more present in organizations similarly to what happened to e-mail. We should also note that the way IM is used in organizations is quite diversified [Nardi, 00]: for instance, to verify other people's availability to talk, to keep one aware of her colleagues' activities, and so on.

Our results indicate that software development work - despite or because of recent technological advancements - still is a predominantly collaborative activity. Interacting with colleagues is an important and necessary aspect of a software developer's work, and, in fact, corresponds to about 45% of the observed time at Sigma. In addition, according to our data, software engineers' work requires a lot of communication and information seeking. Furthermore, collaborative and individual activities are intertwined suggesting that approaches supporting informal collaboration with the ability to move into closely-coupled collaboration when necessary, like those of [Gutwin, 08] and [Geyer, 08], are potentially useful for software developers.

Our results point to low usage of communication tools. This is somewhat surprising since studies in Human Computer Interaction and Computer Supported Collaborative Work indicate that e-mail is the most used tools by professionals in many areas, including software development [Ducheneaut, 01] [Whittaker, 06]. Nevertheless, Sigma professionals use e-mail in only 1% of their time, far away to the

same result found by González and Mark which reported 9.2% time spent in e-mail [González, 04]. Notice that Sigma professionals use e-mail in 8% of their observed time regarding the software process. A possible explanation for these results is the recent adoption of instant messaging tools in work settings [Handel, 02], i.e., instead of using e-mail, Sigma software engineers would prefer to use IM to talk to their colleagues. This explanation, however, is not supported by our data, since the usage of such messaging tools at Sigma is also low.

Another explanation for that result is that the observed software engineers work in the same physical location, i.e., talking (informally or formally) face-to-face requires low effort and, as such, is more practical than using collaborative tools. For instance, our study identified that IM is also used to *invoke* colleagues for talking face to face – as discussed in Section 4.2. This is similar to what was reported by González and Mark in which 3% of the software engineers' time was spent in informal talk "across" the cubicle walls [González, 04]. Or yet, the IM tool is used only for initiating a conversation, as an interview excerpt suggests: "*When I need, I even prefer* [to talk face-to-face]. *When someone sends a doubt by IM I call him/her: 'come here'… So it* [a face-to-face conversation] *can happen every day, every hour"*. This is consistent with what Nardi and colleagues [Nardi, 00] labeled "outeraction", i.e., the usage of IM to negotiate availability for later face-to-face interaction. Our results regarding the low usage of collaborative tools are also consistent with those from Cherry and Robillard [Cherry, 09], which describe that face-to-face communication in a collocated context, is far more used than other available communication media like instant messaging, phone, and so on.

Another result from our study suggests that Sigma developers barely use process tools. In particular, the software process tool used in the Sigma organization provided information about the next task to be performed by a developer and allowed him/her to find out about the tasks being developed by other developers as well the current state of each task. However, as suggested by previous work in the CSCW literature [Handel, 02] [González, 04], communication is a means of keeping people aware of their activities. It occurs in many ways: face-to-face – either informally or during meetings – or through the usage of collaborative tools. Since all members of the team are collocated in a large open area, it is not surprising to find out that the observed software developers would rather *communicate* about the state of the work than use a process tool to find out this information. Furthermore, our interview data also suggests that Sigma developers wait for weekly meetings – scheduled by the team leader – where project status is discussed, and the next activities and phases are assigned to each team member. In other words, Sigma developers do not worry about the tasks to be performed according to the software process, because they are aware that important aspects about it will be discussed on the weekly meetings.

This result about software process tools is particularly relevant for researchers interested in building such tools. Our data suggests that these tools, if they are to be used by software developers in collocated environments, should be easily accessible and integrated into their tools used for daily work. Furthermore, depending on the work setting (collocated vs. distributed), they might only be used for accountability purposes [Dourish, 01]. Furthermore, this result is aligned with agile methods, which emphasize communication and coordination activities [Coplien, 05] without necessarily requiring tool support.

Collocation of software developers also explains another result: according to Figure 5, Sigma software engineers spend 45% of their time with communication-related activities to search information or, more specifically, about 15% with actions which support developers to (a) keep themselves aware of their colleagues' work, (b) share knowledge, and (c) manage dependencies between their activities. This, the time spent in engaging in finding awareness information [Dourish, 92], is an important contribution of this work, since in previous studies, we were not able to identify a percentage to this activity that would allow us to compare our results with.

In general, it is possible to observe that the collocation of software developers in the observed team has a strong influence n the way they work and use collaborative and software process tools. This suggests that office space should be designed taking into account different aspects of knowledge management [Maier, 08]. Previous research in software development suggests that the software architecture is an important aspect to be considered when dealing with coordination of software development work[3]. For instance, there is significantly higher frequency of communication between software developers whose code is interdependent than between software developers whose code is independent [Morelli, 95] [Sosa, 02] [de Souza, 04]. Seat assignment could then be based on the architecture of the system being developed so that developers writing interconnected code would seat closer than others.

We also compared our results with those from Perry and colleagues [Perry, 94]. In Perry's study, the results are presented as bloxplot graphics, but without the presentation of the actual values. So, we used the average between the lowest and the highest values to estimate the actual value. This was done in two of Perry's graphics: the first one with the time spent with tools and the second one with the time when a tool was used. Perry considered a distinction between provoked and suffered interruptions, so we aggregated these aspects in order to be able to compare Perry's results with our results.

According to this comparison, it is possible to find some similarities between Perry's results and ours. For instance, the high percentage of face-to-face communication (64% in Perry's and 75% in our data) and the percentage of telephone use (16% in both). Considering that Sigma software engineers do not use voice mail and that IM was not popular in 1994, we could not compare these specific aspects. Finally, there is one detail that differs from our findings which is about e-mail usage (19% at Perry's research against 1% at ours), as Figures 6 and 7 indicate.

---

3 This has become known as Conway's Law.

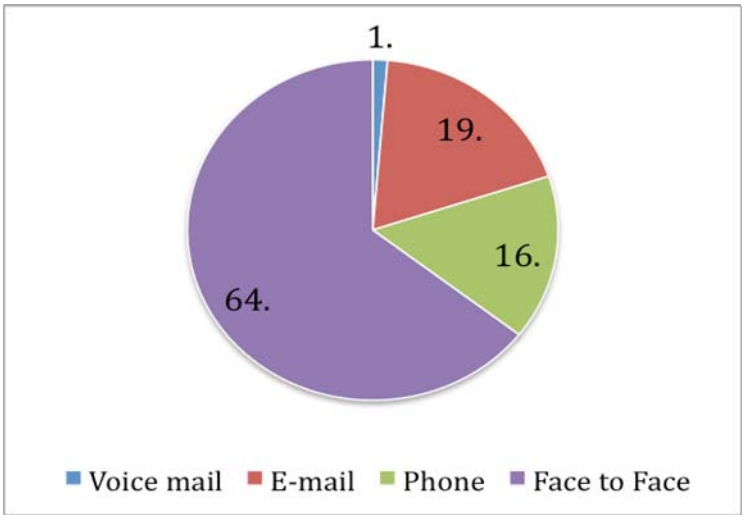

Figure 6: Communication Media used by engineers at Perry et al (1994)

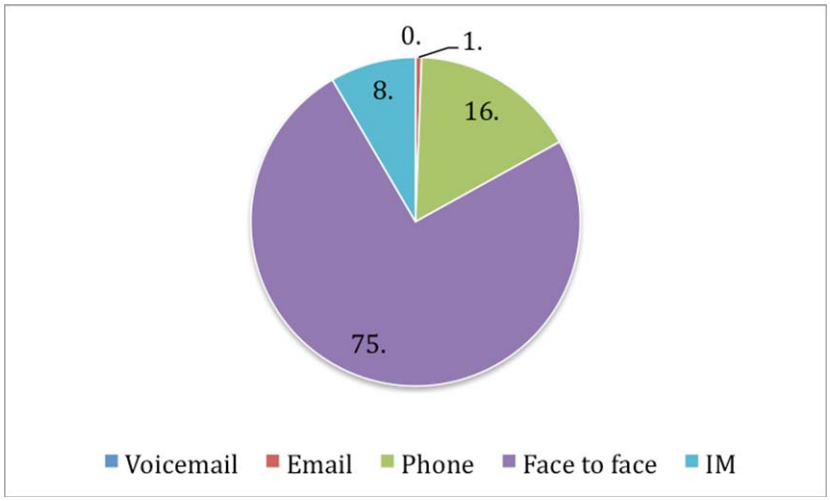

Figure 7: Communication Media used by engineers at Sigma

Finally, it is also important to discuss aspects regarding interruptions in software developers' work. Again, our results are consistent to previous research [González, 04]: software developers (and knowledge workers in general) work about 3 minutes without being interrupted. These interruptions have two reasons: someone else might initiate them, or they occur when a software developer decides to stop his work to engage in a collaborative activity[4]. In any case, this result is particularly surprising

[4] González and Mark [González, 04] call this a task switch.

when one considers the amount of mental effort that one person spends when returning to his work prior to the interruption [Perlow, 99].

6 Final Remarks and Future Work

This paper reported the results of an empirical study aimed to understand software developers' practices regarding collaboration, information seeking, and communication. This study is based on data collected from a software development team from a large Brazilian organization. Our results are consistent with previous work on the area: software development work is inexorably a collaborative activity that is intertwined with individual activities; information seeking is an important part of a software developer's daily work; interruptions do occur often in their daily work; and, finally, collocation of developers facilitates the coordination reducing the need to use certain tools. Furthermore, the identification of the time spent by developers to seek information about their colleagues' activities – time to become aware of colleagues [Dourish, 1992], is an important contribution of this paper.

At first, one can argue that as an observational study, our results can not be generalized because they are associated to a particular context, i.e., with singular characteristics either by the toolset used or by its physical and organizational structure. In other words, the results described in this paper may vary according to the observed organization since cultural, social and organizational issues influence activities' execution [Perlow, 99]. However, the consistency of the presented results with previous research in different organizations, teams, and countries suggest that these results are true for software engineering in general. In fact, we need to stress that studies like the one presented are important because they help to create a corpus of evidence regarding software developers' work.

Acknowledgements

We thank Sigma software engineers for their help during this study. The first author was supported by a fellowship from the Fundação de Amparo à Pesquisa do Estado do Pará (FAPESPA). The second author was supported by the Brazilian Government under grant CNPq 473220/2008-3 and by the Fundação de Amparo à Pesquisa do Estado do Pará (FAPESPA) through "Edital Universal N° 003/2008". This work was also supported by Asociación Mexicana de Cultura A.C.

References

[Ahern, 03] Ahern, D., Clouse A., Turner, R.: CMMI® Distilled: A Practical Introduction to Integrated Process Improvement. Addison-Wesley, 2003.

[Brooks, 74] Brooks, F. P.: The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley, 1974.

[Carmel, 10] Carmel, E. and Prikladnicki, R., Does Time Zone Proximity Matter for Brazil? A Study of the Brazilian I.T. Industry, July 22, 2010. Available at SSRN: http://ssrn.com/abstract=1647305

[Cataldo, 06] Cataldo, M., et al., Identification of Coordination Requirements: implications for the Design of Collaboration and Awareness Tools, in 20th Conference on Computer Supported Cooperative Work. 2006, ACM Press: Banff, Alberta, Canada.

[Cataldo, 08] Cataldo, M., J.D. Herbsleb, and K.M. Carley, Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity, in Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement. 2008, ACM: Kaiserslautern, Germany.

[Cherry, 09] Cherry, S. and Robillard, P. N. 2009. Audio-video recording of ad hoc software development team interactions. In *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering* (May 17 - 17, 2009). CHASE.

[Coplien, 05] Coplien, J.O. and N.B. Harrison, Organizational Patterns of Agile Software Development. 2005, Upper Sadle River, NJ: Pearson Prentice Hall.

[Curtis, 88] Curtis, B., Krasner, H., Iscoe, N.: A field study of the software design process for large systems. In CACM 31, Nov. 11, 1988, 1268-1287.

[Damian, 06] Damian, D., Moitra, D.: Guest Editors' Introduction: Global Software Development: How Far Have We Come? IEEE Software 23(5): 17-19 (2006)

[De Marco, 99] DeMarco, T. and Lister, T. Peopleware—Productive Projects and Teams, Dorset House, 1999.

[de Souza, 04] de Souza, C.R.B., et al. Sometimes You Need to See Through Walls - A Field Study of Application Programming Interfaces. in Conference on Computer-Supported Cooperative Work. 2004. Chicago, IL, USA: ACM Press.

[de Souza, 09] de Souza, C., Sharp, H. et al., Introduction to the Special Issue on Cooperative and Human Aspects of Software Development, IEEE Software, Nov / Dec. 2009.

[Dourish, 01] Dourish, P. Process Descriptions as Organisational Accounting Devices: The Dual Use of Workflow Technologies. In *Proceedings of the ACM Conference on Supporting Group Work GROUP'01* (Boulder, CO), 52-60, 2001.

[Ducheneaut, 01] Ducheneaut, N., Bellotti, V.: Email as habitat: An exploration of embedded personal information management. In ACM Interactions, 2001, 30-38.

[Geyer, 08] Geyer, W., Silva Filho, R. S., Brownholtz, B., Redmiles, D. F. The Trade-Offs of Blending Synchronous and Asynchronous Communication Services to Support Contextual Collaboration, Journal of Universal Computer Science, Special issue on Groupware: Issues and Applications with a selection of papers presented at 12th International Workshop on Groupware, V. 14, No. 1, March 2008, pp. 4-26.

[González, 04] González, V., Mark, G.: Constant, constant, multi-tasking craziness: managing multiple working spheres. In Proceedings of the Conference on Human factors in computing systems. ACM, Vienna, 2004.

[Gutwin, 08] Gutwin, C. Greenberg, S. et al.: Supporting Informal Collaboration in Shared-Workspace Groupware. Journal of Universal Computer Science. Vol. 14, number 9, 1411-1434, 2008.

[Handel, 02] Handel, M., Herbsleb, J.: What is Chat doing in the workplace? In ACM Conference on Computer-Supported Cooperative Work, 2002, 1-10.

[Herbslebm,01] Herbsleb, J. D., Moitra, D. Guest Editors' Introduction: Global Software Development. IEEE Software 18(2): (2001)

[Hertzum, 02] Hertzum, M.: The importance of trust in software engineers' assessment and choice of information sources. In Information and Organization, 2002, v12 i1, 1-18.

[IBM, 09] IBM Rational Unified Process (RUP), 2009, http://www-01.ibm.com/software/awdtools/rup/

[Jorgensen, 89] Jorgensen, D. L. (1989). Participant Observation: A Methodology for Human Studies. Thousand Oaks, CA, SAGE publications.

[Ko, 07] Ko, A., DeLine, R., Venolia, G. : Information Needs in Collocated Software Development Teams. In International Conference on Software Engineering, 2007, 344-353.

[Maier, 08] Maier, R., Thalmann, S., et al: Optimizing Assignment of Knowledge Workers to Office Space Using Knowledge Management Criteria: The Flexible Office Case. Journal of Universal Computer Science. Vol. 14, number 4, 508-525, 2008.

[McCracken, 88] McCracken, G. (1988). The Long Interview. Thousand Oaks, CA, SAGE Publications.

[McGrath, 95] McGrath, J.: Methodology Matters: Doing Research in the Behavioral and Social Sciences. In Baecker, R. & Buxton, W.A.S. (Eds.) Readings in Human-Computer Interaction: An Interdisciplinary Approach. 2nd edition. Morgan Kaufman Publishers, 1995.

[Morelli, 95] Morelli, M.D., S.D. Eppinger, and R.K. Gulati, Predicting Technical Communication in Product Development Organizations. IEEE Transactions on Engineering Management, 1995. 42(3): p. 215-222.

[Nardi, 00] Nardi, B., Whittaker, S., Bradner, E. : Interaction and outeraction: Instant Messaging in Action. ACM Conference on Computer Supported Cooperative Work, 2000, 79-88.

[Perlow, 99] Perlow, L.: The Time Famine: Toward a Sociology of Work Time. In Administrative Science Quarterly, 1999, p. 57-81.

[Perry, 94] Perry, D., Staudenmayer, N., Votta, L.: People, Organizations, and Process Improvement. IEEE Software, 1994, 36-45.

[PMI, 04] A Guide to the Project Management Body of Knowledge (PMBOK® Guide), 2004, Project Management Institute, Third Edition.

[Seaman, 99] Seaman, C.: Qualitative Methods in Empirical Studies of Software Engineering. In IEEE TSE, 1999, vol. 25, no. 4, p. 557-572.

[Sosa, 02] Sosa, M.E., et al., Factors that influence Technical Communication in Distributed Product Development: An Empirical Study in the Telecommunications Industry. IEEE Transactions on Engineering Management, 2002. 49(1): p. 45-58.

[Vessey, 95] Vessey, I., Sravanapudi, A.: CASE tools as collaborative support technologies. In CACM 38, Jan. 1th, 1995, 83-95.

[Weinberg, 71] Weinberg, G. *The Psychology of Programming*, Dorset House, 1971.

[Whittaker, 99] Whittaker, S., Schwarz, H. Meetings of the Board: The Impact of Scheduling Medium on Long Term Group Coordination in Software Development. Journal of Computer Supported Cooperative Work 8(3): 175-205 (1999)

[Whittaker, 06] Whittaker, S., Bellotti, V., Gwizdka, J.: Email as personal information management. In CACM. 2006; 49 (1): 68-73.