What Distributed Software Teams need to know and when: an Empirical Study

Kevin Dullemond, Ben van Gameren
Delft University of Technology
IHomer
The Netherlands
{k.dullemond, b.j.a.vangameren}@tudelft.nl

***Abstract*—Just like in co-located teams, software engineers in distributed teams need a variety of information about the project and their team members to collaborate effectively. In contrast with the co-located situation however, acquiring and maintaining sufficient awareness is more difficult. Therefore technological support is developed to assist them with this. However, such support walks a fine line: if it provides too little information software engineers will not be able to collaborate effectively, yet if it provides too much, an information overload can occur. To further complicate matters, the information needs of software engineers dynamically change based on their current activity, context and focus. Therefore we assist tool developers by investigating and reporting on the prioritization of information for distributed software engineers based on their current activity and status. Finally, we illustrate the applicability of the findings by describing how to apply them in a support tool for distributed software engineers.**

I. Introduction

It is becoming increasingly common for collaborative Software Engineering teams to no longer conduct their work from a single office building. This happens both due to the globalization of business [1], [2], [3] and because people increasingly work from home [4]. Like their co-located counterparts, members of distributed teams need access to certain information to collaborate effectively [5], [6]. However, while in a co-located setting gathering this information can be done naturally with relatively little effort and obtrusion, the distributed setting requires special care because information that is necessary needs to be gathered explicitly.

Therefore, to support the information needs of distributed software engineers, specialized tooling has been developed. However the need to explicitly look up information can be particularly obtrusive to software engineers because this requires them to repeatedly interpreted the same information and spend cognitive effort to decide if something noteworthy has happened. This can be mitigated by making the changes available and communicating these to the software engineers. Even when taking this approach however, choices will need to be made regarding what information is necessary to be able to create a comprehensive Collaborative Development Environment without overloading the software engineers with information [7], [8]. To be able to make these choices it is important to be able to prioritize the different information needs of distributed software engineers. Therefore, to be better able to support distributed software engineering teams, the **goal** of this paper is:

"To investigate the prioritization of different information needs of distributed software engineers."

In this prioritization the importance of information is likely to be strongly related to (i) how involved a software engineer is with the project the information is about, (ii) his current status and (iii) the frequency the information generally changes. Therefore, to reach the goal we have defined the following research questions:

RQ1 What is the most important information a distributed software engineer needs to know to carry out his work and collaborate effectively with his team members?

RQ2 What is the relation between the involvement of a distributed software engineer with a project and the information he needs to know about that project?

- **RQ2a** What is the most important information a distributed software engineer needs about a project he is currently working on?
- **RQ2b** What is the most important information a distributed software engineer needs about a project he is part of but not currently working on?
- **RQ2c** What is the most important information a distributed software engineer needs about another project in the organization?

RQ3 Is there a difference between the information needs of a software engineer when he is on holiday, enjoying free time and commuting?

- **RQ3a** What is the most important information a distributed software engineer needs to know when he is on holiday?
- **RQ3b** What is the most important information a distributed software engineer needs to know when he is enjoying free time?
- **RQ3c** What is the most important information a distributed software engineer needs to know when he is traveling to or from his work location or customers?

RQ4 Is there a relation between the importance of updates about a specific type of information and the frequency it changes?

Answering these research questions will help in reaching the research goal by eliciting: the information needs of distributed software engineers **(RQ1)**, the mutual importance of these both inside **(RQ2)** and outside of working hours **(RQ3)**, and the relation between the prioritization and the frequency an information item changes **(RQ4)**. We investigate this empirically at a distributed Software Engineering company by gathering a list of the most important information items in a focus group and subsequently performing a questionnaire to prioritize this list.

This paper is structured as follows. First in section II we discuss background and related work of this research. Following this, in section III we discuss the research site and methods of data collection and analysis we used in the study. Subsequently we present the results and findings in section IV and a discuss how we intend to apply the findings in a tool we are developing called Iris, in section V. Finally, we discuss the threats to the validity of this study in section VI and present conclusions and discuss future research in section VII.

II. Background and Related Work

For Software Engineering teams it is vital to stay up to date on each others' status and the overall project status. This knowledge provides a context for individual team members to carry out their collaborative activities in coordination with the rest of the team. It includes things such as information about the other members in the project team, their activities and the overall project status. Overall, this knowledge is often referred to as awareness, defined by Dourish and Bly as: *"An understanding of the activities of others which provides a context for your own activity"* [9].

Much of such information is propagated naturally in the software team when working co-located, although specific mechanisms such as stand-up meetings in Scrum [10] are used as well. However, when the software team works distributed this complicates making the information available in the team significantly and much attention needs to be placed in keeping each other up to date on changes that have occurred or are occurring [11]. This has been recognized by tool developers in the GSE domain, who attempt to resolve this by making use of shared artifacts [12]. Firstly, there exist formal development tools which are also commonly used by co-located software teams such as issue management systems (e.g. Fogbugz[1] and Bugzilla[2]) and code repositories (e.g. Subversion[3] and Git[4]). Secondly, there are specialized tools for distributed software engineers. Examples are FastDash [13] and Expertise Browser [14]. FastDash provides a spatial representation of a shared code base to the team members and displays such things as which team members have source files checked out, which files are being viewed, and what methods and classes are currently being changed. Expertise Browser is a tool to find domain experts for a set of software artifacts and to inspect their expertise profile. Another good example of a technique used by tooling to support the sharing of awareness information using shared artifacts is social tagging. Social tagging is the collaborative activity of marking shared content with tags as a way to organize content for future navigation, filtering, or search [15]. It can be found in Jazz[5] and TagSEA [16].

This is just a sample of many of such tools with quite diverse isolated purposes. Instead of using a high variety of different single purpose tools, Booch and Brown [17] propose the use of a Collaborative Development Environment: *"a virtual space wherein all the stakeholders of a project - even if distributed by time or distance - may negotiate, brainstorm, discuss, share knowledge, and generally labor together to carry out some task, most often to create an executable deliverable and its supporting artifacts"*. When moving to such an integrated solution, however, there exists a threat in overloading the software engineers with information (although one could argue this is already the case with the set of specialized tools).

The threat of an overload of information has long been recognized in computer mediated information systems [7]. In the general context it is defined as *"information presented at a rate too fast for a person to process"* [18]. The existence of information overload has been shown in many disciplines. Eppler and Mengis [8] for example provide a review of literature about information overload in organization science, accounting, marketing and Management Information Systems. Researchers in the field of tool support for sharing awareness have also recognized the threat of information overload (e.g. [19], [20]). One method of dealing with the overload of information is the use of the delta mechanism as introduced in the 1970s in the context of version control systems to save on bandwidth usage [21], [22]. In the same fashion the mechanism can also be used to save *'cognitive bandwidth'* of software engineers by only communicating the changes to them. These changes will frequently provide the software engineer with sufficient information as Biehl et al. [13] states: *"Key information items are the items that change on a daily, hourly, or minute-by-minute basis"* and *"while most information items change on a weekly or monthly basis, the most often used are those that change on a daily, hourly or minute-by-minute basis"*. In fact, some tool developers have recognized this. For example, Jazz [23] uses RSS feeds to allow software developers to subscribe to a range of workspace-related events of potential interest, e.g. that a change has been made to a project artifact or that a project build has failed.

Using the delta mechanism however does not resolve the need to decide what information is important. With so many diverse topics covered by these specialistics tools it is important to decide what information is important and when. Holmes and Walker agree [24]: *"Awareness is often impeded at two ends of the spectrum: a lack of information, when the changes only become apparent when a build breaks or bugs appear; or an excess of information, where the changes are announced but the majority of the changes are not relevant to the developer in her particular project and context. The middle-ground is unpopulated: we lack automated support for developer-specific awareness (DSA)"*. Additionally in their paper on information overload from 1985 Hiltz and Turoff [7] already recognized that computer mediated communication systems need structuring to avoid an overload of information and that this structuring should be imposed by individuals and

[1] http://www.fogcreek.com/fogbugz/

[2] http://www.bugzilla.org/

[3] http://www.subversion.tigris.org

[4] http://www.git-scm.com

[5] http://www-01.ibm.com/software/rational/jazz/

user groups according to their needs and abilities, rather than through general software features.

The study in this paper aims to find out what information needs distributed software engineers have, what their mutual prioritization is and how this prioritization changes. Related to this research Aranda et al. [25] have researched the specific information needs distributed software engineers have about their co-workers. The research presented in this paper mainly differs from this research in two ways: (i) in scope as we aim to present a prioritization of all information important to distributed software engineers and (ii) the fact that our research includes the status of software engineers and their involvement with the context the information is about.

III. Research Site and Method

A. Research Site

This study is carried out at IHomer, a Dutch Software Engineering company which exists since August of 2008 in which it is common practice to work from home. All employees are Dutch and live in the Netherlands therefore challenges caused by different timezones and by different cultural backgrounds are rare. The company uses a flat organizational structure in which all employees are responsible for all business decisions such as the strategy, vision and core values. Because of this approach, all employees have a relatively high entrepreneurial spirit. The flat organizational structure differs with the more common hierarchical structure in which employees are mainly responsible for the specific role they fulfill. In the company however cross functional teams are the standard and the different roles are shared between team members.

At the moment the company employs 21 people who work on a variety of products, projects and contracts. These people maintain close personal relationships with each other. While it is common practice to work from home, the employees try to get together once a week on Tuesdays to meet face-to-face at an office to stay connected. The company has grown over the past years and initially on Tuesdays everyone discussed what they were doing. This worked well until the company size reached 16, and then sub teams were formed to keep this face-to-face communication more tractable. Teams cluster according to various factors: projects and related technologies being two of them. The largest team consists of 7 people but the overall team is still very close with personnel moving between teams and teams exchanging projects as needed.

B. Research Method

To reach the research goal and answer the research questions we used two methods to acquire the empirical data in this study: a focus group and a questionnaire.

1) Focus Group: We performed a focus group [26] to discover the information needs in a specific distributed Software Engineering company. The main advantage of using this method is that it enables the participants to build on the responses and ideas of others, which increases the richness of the information gained and the chance of reaching a complete set of information needs [27].

The focus group we performed lasted approximately 1.5 hours and we selected 4 out of the 21 software engineers based on availability. We The first two authors are employees of the company but were explicitly excluded in the selection process. Additionally, the first author moderated the focus group. In carrying out the focus group we followed a structured approach[6]. We asked the participants about the most important information items in their work on three consecutive levels of abstraction: information about a project, information about the organization and information about people. We used this structuring to assist participants in arriving at an as complete set of important information items as possible. Finally, the focus group was conducted in a separate closed office to protect the focus group from outside influences.

2) Questionnaire: We choose to use a questionnaire [28] because this method makes it feasible to include the opinions of a relatively large group of people by using a standardized set of questions. In this case we were able to include the opinions of everyone in the company. In the questionnaire[7] we asked the respondents to rate how important they consider receiving updates on each of the information items found in the focus group, based on their current activity or status. Additionally we asked for their perception of how often each of the information items generally changes. Finally, we also asked whether they thought there were information items missing from the lists because not everyone could participate in the focus group. To determine the relative importance we asked the respondents to rate it on a 5-point Likert scale [29] ranging from *'very unimportant'* to *'very important'*. We included a *'no-opinion'* option to prevent people with no opinion on a specific question to answer it anyway and 'pollute' the data in this fashion [30]. We sent the questionnaire to all 19 employees of the company who are not authoring this paper and all 19 returned the questionnaire. The results of the questionnaire are available online in anonymized form[8].

Because of the limited size of our sample we cannot draw statistically significant conclusions. We have chosen to use an approach analogous to the approach taken by Aranda et al. [25]. In this study they conducted a questionnaire with a sample size similar to ours (23 participants) about how useful it is to know certain characteristics of your distributed colleagues and define a *"Usefulness indicator"* (UI) to rank the different data items in their questionnaire. Analogous to this we define the *"Importance Indicator"* (II) as follows:

$II_i = (VI_i + Ii)/(VI_i + Ii + N_i + NVI_i + NI_i)$ where:

1) i indicates the data item identification
2) VI_i indicates the number of people that considered data i *"Very Important"*
3) I_i indicates the number of people that considered data i *"Important"*
4) N_i indicates the number of people that considered data i *"Neutral"*
5) NVI_i indicates the number of people that considered data i *"Not Very Important"*
6) NI_i indicates the number of people that considered data i *"Not Important at all"*

[6] See http://www.aspic.nl/ICGSE2013/FocusGroupGuide.pdf for a translation

[7] See http://www.aspic.nl/ICGSE2013/Questionnaire.pdf for a translation

[8] See http://aspic.nl/ICGSE2013/QuestionnaireResults.csv for the questionnaire results in a comma separated file which can for instance be opened with Microsoft Excel

IV. Findings

In this section present the results and findings of the study to answer the research questions defined in section I. We structure this section based on the research questions, answering each of the four main research questions in a separate subsection.

A. The most important information items

Research question 1 is: *"What is the most important information a distributed software engineer needs to know to carry out his work and collaborate effectively with his team members?"*

We answer this research question directly by presenting the lists of most important information items found in the focus group in tables I, II and III respectively:

TABLE I. Project Specific Information Items

Technological agreements *e.g. on programming language, frameworks or standards to use*
Requirements
Risks *(project specific)*
Process agreements *e.g. roles, stakeholders, the process type*
Issues *(tasks)*
System under construction
Source *(repository)*
Build status *e.g. build succeeded/failed*
Deployment Status *e.g. currently deployed version, is it running?*
Planning
Deadlines
Meetings
Status
Hours worked on the project
Milestones
Phase of project *e.g. starting up, active, commissioning, done*
Project related communication with the customer *e.g. mail, phone calls, transcripts*
Project related communication with the team *e.g. mail, phone calls, transcripts*

TABLE II. Organization Specific Information Items

Risks *(organization specific)*
Customer relations
Billing status *e.g. sent out, paid, overdue*
Organizational development
Action points *e.g. things that need to be prepared or researched*
Planning
Organizational meetings *e.g. date, content*
Business opportunities *e.g. possible new customers or projects*
Applicants *e.g. possible new colleagues*
Financials *e.g. liquidity, investments, Year-To-Date figures, forecast*

Additionally, in the questionnaire we asked the respondents to indicate whether they thought there were information items missing from the list. The most answers given here indicated the information items are quite broad and therefore more specific items are missing (e.g. splitting financials and adding communication with subcontractors). This is true but intentional because the scope of this study is wide and the size of the questionnaire needed to stay manageable. It is a good idea to repeat the questionnaire in a more confined context with more detailed information items. Finally, one of the respondents indicated he found it important to include the severity of the change in an information item in the decision whether it is important to him. He stated: *"If there are problems then I want to know about it even when on holiday or when I'm enjoying my free time, but for other things there need to be boundaries between my work and private life."*.

TABLE III. Personal Information Items

Contact information *e.g. mail, phone number, preferred means of contact based on specific situations*
Approachability *e.g. what means of contact are available? Can he/she be disrupted?*
Current activity
Planning
Agenda *e.g. planned activities*
Holidays
Idleness *e.g. when is there no billable work available for him/her?*
Seniority *e.g. junior, senior, years of experience*
Happiness *e.g. mood, is he/she content in general?*
Personal situation *e.g. what is going on in his/her life?*
Personal information *e.g. hobbies, name of spouse and children, age*
Team *e.g. in what team does he/she work?*
Knowledge/Skills/Expertise

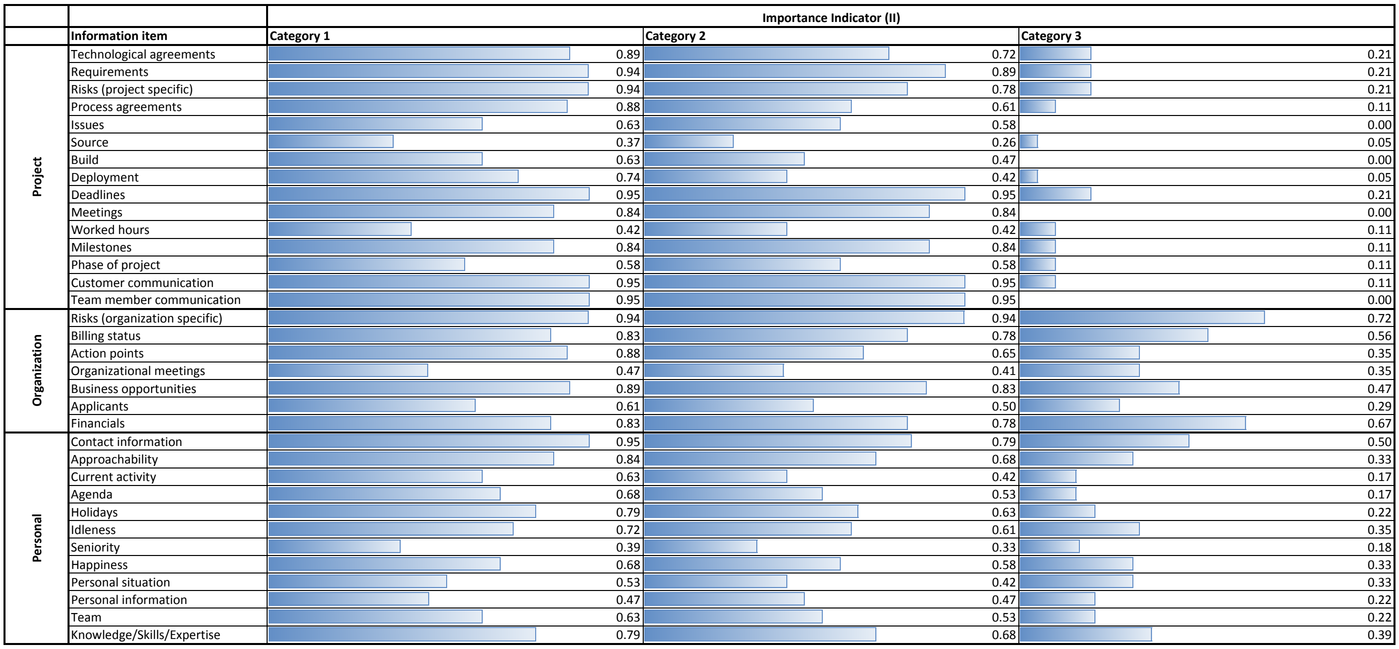
B. Project involvement and the importance of information

Research question 2 is: *"What is the relation between the involvement of a distributed software engineer with a project and the information he needs to know about that project?"*

We answer this by presenting figure 1. In this figure we show the Importance Indicator of each of the information items for each of three categories of involvement:

- **Category 1** Means the information is directly related to a project the engineer is working on **right now**
- **Category 2** Means the information is directly related to a project the engineer is part of. However the engineer is **not** working on that project **at this moment**
- **Category 3** Means the information is related to a project in the company, but the engineer is **not a part of that project**

In this figure we see a couple of things. Firstly we see that the more information lies outside of the context of distributed software engineers, the less important it is to stay up-to-date on it. We can see this from the gradual decrease in importance across the three categories. Secondly we see the reduction in importance is larger between the second and third category than between the first and second ones. From this we can conclude information about projects people are part of but not working on at the moment, is still quite important. Finally, we see that the reduction in importance across the three categories is lowest for information items related to the organization. This makes sense because, out of the three categories, these



		Importance Indicator (II)		
	Information item	Category 1	Category 2	Category 3
Project	Technological agreements	0.89	0.72	0.21
	Requirements	0.94	0.89	0.21
	Risks (project specific)	0.94	0.78	0.21
	Process agreements	0.88	0.61	0.11
	Issues	0.63	0.58	0.00
	Source	0.37	0.26	0.05
	Build	0.63	0.47	0.00
	Deployment	0.74	0.42	0.05
	Deadlines	0.95	0.95	0.21
	Meetings	0.84	0.84	0.00
	Worked hours	0.42	0.42	0.11
	Milestones	0.84	0.84	0.11
	Phase of project	0.58	0.58	0.11
	Customer communication	0.95	0.95	0.11
	Team member communication	0.95	0.95	0.00
Organization	Risks (organization specific)	0.94	0.94	0.72
	Billing status	0.83	0.78	0.56
	Action points	0.88	0.65	0.35
	Organizational meetings	0.47	0.41	0.35
	Business opportunities	0.89	0.83	0.47
	Applicants	0.61	0.50	0.29
	Financials	0.83	0.78	0.67
Personal	Contact information	0.95	0.79	0.50
	Approachability	0.84	0.68	0.33
	Current activity	0.63	0.42	0.17
	Agenda	0.68	0.53	0.17
	Holidays	0.79	0.63	0.22
	Idleness	0.72	0.61	0.35
	Seniority	0.39	0.33	0.18
	Happiness	0.68	0.58	0.33
	Personal situation	0.53	0.42	0.33
	Personal information	0.47	0.47	0.22
	Team	0.63	0.53	0.22
	Knowledge/Skills/Expertise	0.79	0.68	0.39

Fig. 1. Importance of information based on project involvement

information items are the least tied to the project setting[9]. Even so, we still see a reduction in importance as these information items get less related to the projects respondents are involved in.

In section I we defined three subquestions:

RQ2a What is the most important information a distributed software engineer needs about a project he is currently working on?

RQ2b What is the most important information a distributed software engineer needs about a project he is part of but not currently working on?

RQ2c What is the most important information a distributed software engineer needs about another project in the organization?

With respect to **RQ2a** we see updates on quite a lot of information items are considered important or very important. If we take the (arbitrary) Importance Indicator value of 0.9 as a cut-off point we get a top-7 of most important information items: Updates on (1) Requirements, (2) Risks, (3) Deadlines, (4) Customer Communication and (5) Team Communication from project, Organizational Risks from Organization and Contact Information from Personal. If we move our attention to the second category for **RQ2b** we see a slight reduction in the importance of the information items. Now only four items make the cut-off of 0.9, namely (1) Deadlines, (2) Customer Communication and (3) Team Communication from project and Risks from organization. Updates on information about colleagues that are not working on the same project at the moment seem to be less important. Finally, with respect to **RQ2c** we see no items are left that have a score higher than 0.9. The three most important information items to receive updates on when the information is not related to a project a distributed software engineer is a part of, are all organizational in nature: Risks (0.72), Financials (0.67) and Billing status (0.32). It is also quite interesting to see all updates on information items about projects in which a distributed software engineer is not a member are considered to be unimportant.

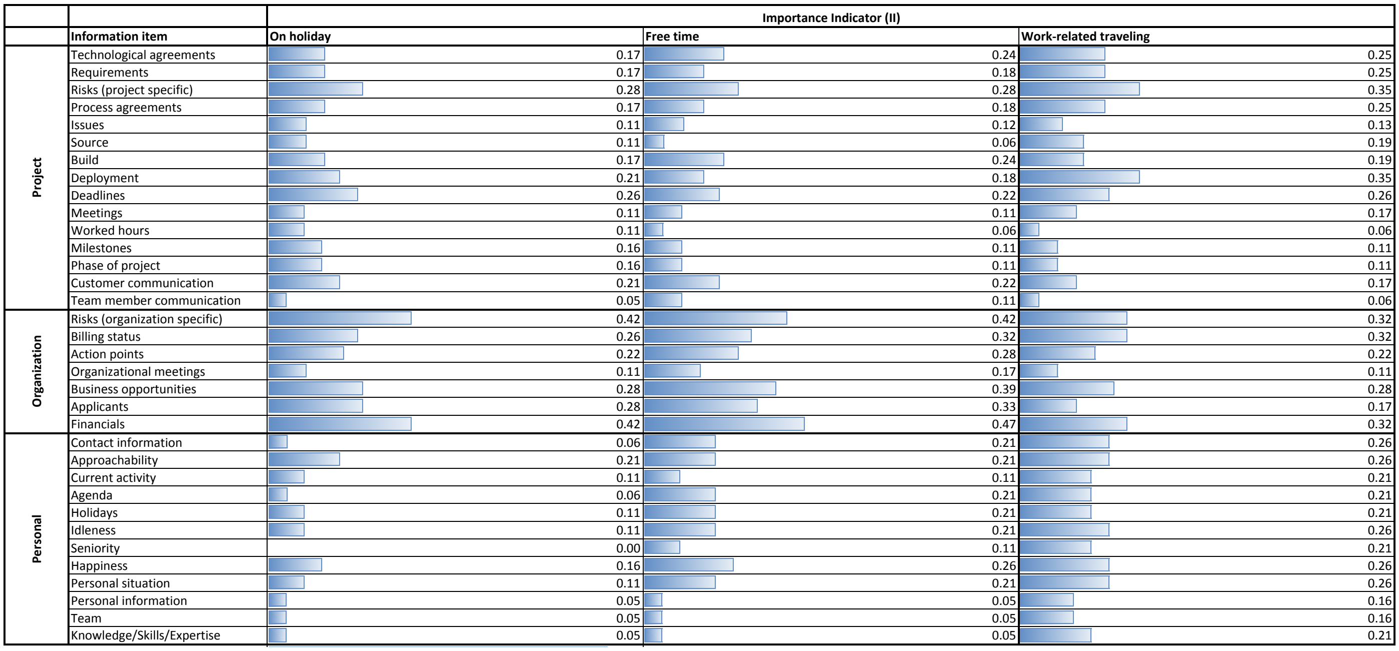
C. *The importance of information outside of working hours*

Research question 3 is: *"Is there a difference between the information needs of a software engineer when he is on holiday, enjoying free time and traveling to and from his work location or customers?"*

We answer this by presenting figure 2. In this figure we show the Importance Indicator of each of the information items dependent on three statuses of a software engineer in which he is not performing his core activity: (i) being on holiday, (ii) enjoying free time and (iii) performing work-related traveling.

Again, in this figure we can see a couple of things. Firstly, we see the importance of all items is generally far lower than when the distributed software engineers are performing their core activity (compare with the values in figure 1). Secondly we see that in general, updates on the information items are considered more important during work-related travel than during free time or a holiday. This could indicate that people feel closest to their work during work-related travel followed by free time and finally holiday. There is a notable exception to this trend as the importance indicators for updates about organizational information items are relatively stable across the three categories. This can be because the people in the company we performed the study at are entrepreneurial and therefore find organizational updates equally significant no matter what non-work related activity they are doing.

[9] to see that organizational information items are tied to the project setting, imagine someone applying for a job opening connected to a specific project.



		Importance Indicator (II)		
	Information item	On holiday	Free time	Work-related traveling
Project	Technological agreements	0.17	0.24	0.25
	Requirements	0.17	0.18	0.25
	Risks (project specific)	0.28	0.28	0.35
	Process agreements	0.17	0.18	0.25
	Issues	0.11	0.12	0.13
	Source	0.11	0.06	0.19
	Build	0.17	0.24	0.19
	Deployment	0.21	0.18	0.35
	Deadlines	0.26	0.22	0.26
	Meetings	0.11	0.11	0.17
	Worked hours	0.11	0.06	0.06
	Milestones	0.16	0.11	0.11
	Phase of project	0.16	0.11	0.11
	Customer communication	0.21	0.22	0.17
	Team member communication	0.05	0.11	0.06
Organization	Risks (organization specific)	0.42	0.42	0.32
	Billing status	0.26	0.32	0.32
	Action points	0.22	0.28	0.22
	Organizational meetings	0.11	0.17	0.11
	Business opportunities	0.28	0.39	0.28
	Applicants	0.28	0.33	0.17
	Financials	0.42	0.47	0.32
Personal	Contact information	0.06	0.21	0.26
	Approachability	0.21	0.21	0.26
	Current activity	0.11	0.11	0.21
	Agenda	0.06	0.21	0.21
	Holidays	0.11	0.21	0.21
	Idleness	0.11	0.21	0.26
	Seniority	0.00	0.11	0.21
	Happiness	0.16	0.26	0.26
	Personal situation	0.11	0.21	0.26
	Personal information	0.05	0.05	0.16
	Team	0.05	0.05	0.16
	Knowledge/Skills/Expertise	0.05	0.05	0.21

Fig. 2. Importance of information based on whether the software engineer is on holiday, enjoying free time or doing work-related traveling

Finally we want to emphasize that even though the importance indicator values are significantly lower than for the categories looked at for the previous research question, the values are still considerable. Take note that the value indicates the ratio of respondents that indicated they find updates on that information item in that situation either important or very important. We find it noteworthy that for instance 42% of the respondents considered it important or very important to stay up-to-date on organization-wide risks and financials during their holiday.

In section I we defined three subquestions:

RQ3a What is the most important information a distributed software engineer needs to know when he is on holiday?

RQ3b What is the most important information a distributed software engineer needs to know when he is enjoying free time?

RQ3c What is the most important information a distributed software engineer needs to know when he is traveling to or from his work location or customers?

With respect to all three of the subquestions we see the values are far lower than for **RQ2**. The most important information items are:

- **For holiday**
  Organization specific risks, Financials, Project specific risks, Business opportunities and Applicants
- **For free time**
  Financials, Organization specific risks, Business opportunities, Applicants and Billing status
- **For work-related travel**
  Project specific risks, Deployment, Organization specific risks, Billing status and Financials

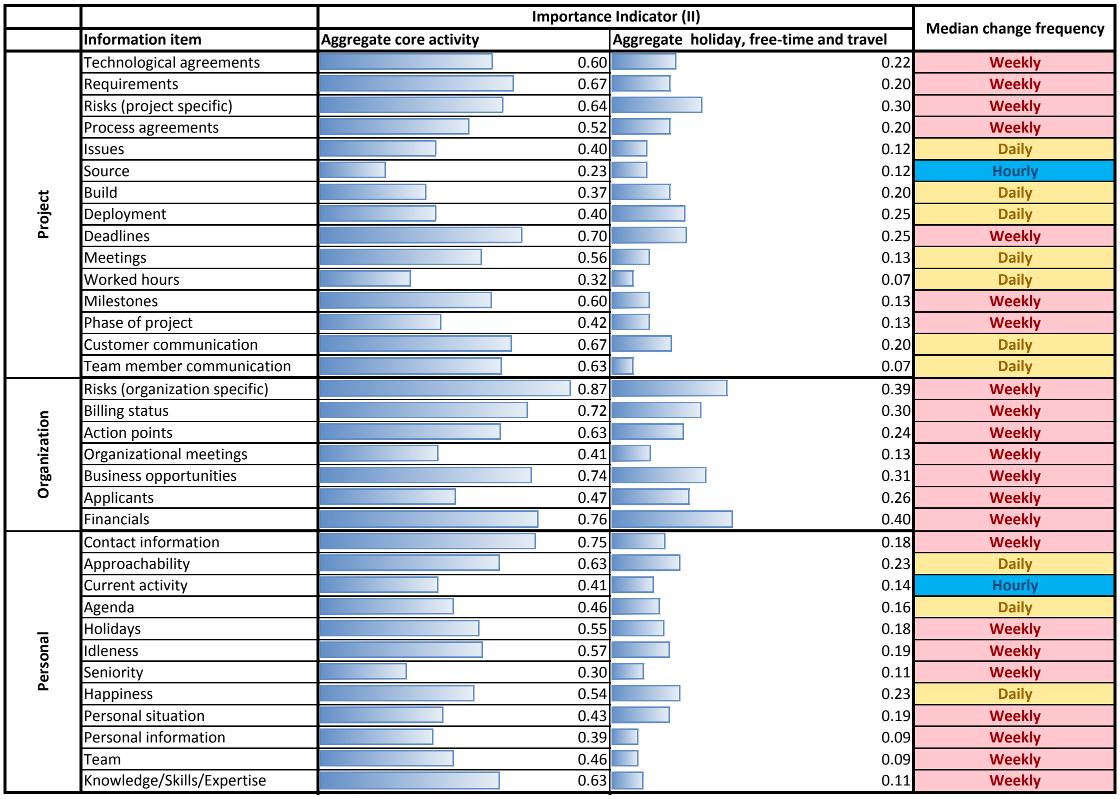
The main things we notice in these three lists is that while for holiday and free time the emphasis lies on updates on organizational information items, this is slightly less the case for work-related traveling where the first two in the list are project related. We think this is because, out of the three categories, work-related travel is generally closest to starting to do your core activity and that is when the practicalities of project related information gain importance in relation to the more general organizational information items.

D. Relation between importance of updates and frequency of change

Research question 4 is: *"Is there a relation between the importance of updates about a specific type of information and the frequency it changes?"*

To answer this research question we present figure 3. In this figure we show for each of the information items the median frequency the respondents indicated the item changes. The options the respondents could choose from are: *'minute-to-minute'*, *'hourly'*, *'daily'*, *'monthly'* and *'less frequently'*. In the figure, we see most items change on a weekly basis on aggregate while some items also change daily or hourly. We also see the organization related information items change relatively less frequently, while the project-related information items overall change more frequently.

Subsequently, we present the aggregate importance indicators across the core-activity-related categories discussed in subsection B and the non-core-activity-related categories discussed in subsection C (holiday, free time and work-related travel). Similarly to the importance indicators presented in the previous sections, the aggregate importance indicator is calculated for each information item by computing the relation of the number of *'Very Important'* and *'Important'* votes to the



		Importance Indicator (II)		Median change frequency
	Information item	Aggregate core activity	Aggregate holiday, free-time and travel	
Project	Technological agreements	0.60	0.22	Weekly
	Requirements	0.67	0.20	Weekly
	Risks (project specific)	0.64	0.30	Weekly
	Process agreements	0.52	0.20	Weekly
	Issues	0.40	0.12	Daily
	Source	0.23	0.12	Hourly
	Build	0.37	0.20	Daily
	Deployment	0.40	0.25	Daily
	Deadlines	0.70	0.25	Weekly
	Meetings	0.56	0.13	Daily
	Worked hours	0.32	0.07	Daily
	Milestones	0.60	0.13	Weekly
	Phase of project	0.42	0.13	Weekly
	Customer communication	0.67	0.20	Daily
	Team member communication	0.63	0.07	Daily
Organization	Risks (organization specific)	0.87	0.39	Weekly
	Billing status	0.72	0.30	Weekly
	Action points	0.63	0.24	Weekly
	Organizational meetings	0.41	0.13	Weekly
	Business opportunities	0.74	0.31	Weekly
	Applicants	0.47	0.26	Weekly
	Financials	0.76	0.40	Weekly
Personal	Contact information	0.75	0.18	Weekly
	Approachability	0.63	0.23	Daily
	Current activity	0.41	0.14	Hourly
	Agenda	0.46	0.16	Daily
	Holidays	0.55	0.18	Weekly
	Idleness	0.57	0.19	Weekly
	Seniority	0.30	0.11	Weekly
	Happiness	0.54	0.23	Daily
	Personal situation	0.43	0.19	Weekly
	Personal information	0.39	0.09	Weekly
	Team	0.46	0.09	Weekly
	Knowledge/Skills/Expertise	0.63	0.11	Weekly

Fig. 3. Importance of information based on whether the software engineer is on holiday, enjoying free time or doing work-related traveling

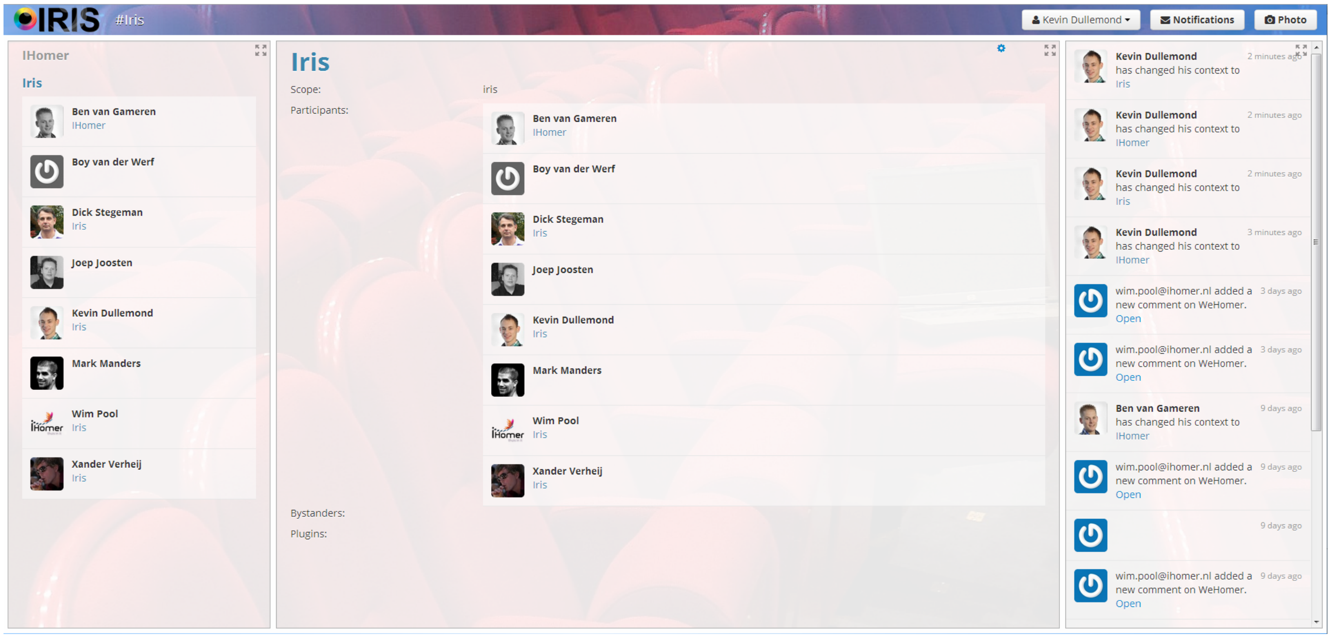
total number of responses excluding *'No opinion'*. We present this metric to try and draw conclusions regarding the relation between the importance of information items and the frequency in which they change as predicted by Biehl et al. [13]. They state: *"Key information items are the items that change on a daily, hourly, or minute-by-minute basis"* and *"while most information items change on a weekly or monthly basis, the most often used are those that change on a daily, hourly or minute-by-minute basis"*. However, we cannot find evidence in our data that such a relation exists. Additional research is required to find whether it does indeed exist.

V. Applying the findings in Iris

As we have discussed previously, one of the reasons for performing this study is to help ourselves in the development of a tool called Iris. We present this application to illustrate how the findings in this paper can be applied. Iris is a Collaborative Development Environment we are currently developing, with support for collaboration at its core. We have reported on an earlier version of Iris before in [31] and a screenshot of the current version of Iris can be seen in figure 4. In this figure it can be seen the user interface is split in three sections. The left section depicts the different contexts a user is part of and allows the user to navigate between these. A context can be defined on a variety of abstraction levels as long as it confines some sort of activity. In the current version of Iris we have configured contexts to be the different projects that exist in the company. When the user navigates to a certain context he moves into it and automatically moves out of his previous context. Subsequently, the interface is altered to reflect the information that is applicable to the context he moved to. For example the user can see the users that are members of the context he switched to and who of them is active in the same context at the moment.

The middle section of the user interface is intended to show information specific to the selected context. As can be seen in the screenshot we are still working on this. Examples of things that can be shown here are the issues in the Issue Management system and information about the latest build associated with the project. Finally, in the section on the right, updates on information items are shown. Again these information items are confined by the context in which the user is currently active. So, it is possible to see updates on: (i) people that enter or leave the context, (ii) builds that succeed or fail, (iii) commits that are made on the code repository and (iv) issues that are assigned and resolved in the Issue Management System.

We intend to apply the findings of the study reported on in this paper in a number of ways. Firstly, we intend to use the list of prioritized information items to decide for which information items we should build support first. We are developing the system incrementally and want to add functionality that brings the most value to the end users. Subsequently, as we add more and more types of information to the system, users will have to configure what types of information they consider interesting in what particular situation. The reason users will have to configure this themselves is because what information is important will differ individually between users.



IHomer

Iris

Ben van Gameren
IHomer

Boy van der Werf

Dick Stegeman
Iris

Joep Joosten

Kevin Dullemond
Iris

Mark Manders

Wim Pool
Iris

Xander Verheij
Iris

Iris

Scope: iris

Participants:

Ben van Gameren
IHomer

Boy van der Werf

Dick Stegeman
Iris

Joep Joosten

Kevin Dullemond
Iris

Mark Manders

Wim Pool
Iris

Xander Verheij
Iris

Bystanders:

Plugins:

Kevin Dullemond 2 minutes ago
has changed his context to
Iris

Kevin Dullemond 2 minutes ago
has changed his context to
IHomer

Kevin Dullemond 2 minutes ago
has changed his context to
Iris

Kevin Dullemond 3 minutes ago
has changed his context to
IHomer

wim.pool@ihomer.nl added a new comment on WeHomer. 3 days ago
Open

wim.pool@ihomer.nl added a new comment on WeHomer. 3 days ago
Open

Ben van Gameren 9 days ago
has changed his context to
IHomer

wim.pool@ihomer.nl added a new comment on WeHomer. 9 days ago
Open

9 days ago

wim.pool@ihomer.nl added a new comment on WeHomer. 9 days ago
Open

Fig. 4. The user interface of Iris

Configuring such a list, however, can require a lot of effort from a user. Therefore it is highly valuable to provide users with a default configuration which is right for the majority of the settings. Subsequently they can tweak the settings to match their individual preferences, which requires far less effort.

Furthermore, we are still discovering what statuses and contexts are valuable to recognize in the system. The results of this study provide us with valuable insights on what people find important, both when performing their core activity and when traveling or going on holiday. For instance, we found evidence that updates on certain organizational information items are equally important no matter how related it is to the project a distributed software engineer is currently working on. In contrast to this, we found the importance of information about projects and people is far higher if it is related to a project a distributed software engineer is currently working on or at least part of. Both these findings are valuable insights to decide what information should be blocked by a *'virtual office wall'* [32] and what information should be allowed to pass.

Subsequently, also data on the relative value of different types of information outside of office hours can be applied to Iris. For instance, the data showed updates on project and personal information items to be more valuable during work-related travel than during free time or holidays. Further we found the value of updates about organizational information types to be relatively stable across these three categories. Finally, we learned there is not much difference between free-time and holiday when it comes to the importance of updates on the information types we researched.

VI. Threats to Validity

Threats to external validity can exist at each of the levels of generalization in a study. In our study, a threat to external validity exists in the generalization of the single distributed Software Engineering company to all distributed Software Engineering companies. Furthermore the company we performed the study at is only geographically distributed and does not encounter working in different time zones or people with different cultural backgrounds because every employee is Dutch and works from the Netherlands. Finally, results are also more difficult to generalize beyond this single company because the company is setup in a uncommon fashion. The company is a flat organization (non-hierarchical), all employees own part of company and all decisions relevant to the entire company are taken jointly by all employees. This could have, for instance, resulted in the relatively high amount of entrepreneurial information items. To be able to better generalize beyond the setting we performed the study in, the study should be repeated in other companies as well. With respect to the generalization of the sampled data to the population of the company, our work is much less threatened. For the focus group we sampled 4 out of the 19 applicable people (the first two authors are excluded from taking part) in the company and for the questionnaire our response rate was 100% (19 out of 19).

Furthermore, there exist threats to construct validity in our study. Firstly, we attempted to mitigate threats to reliability by describing our research site and methods and making the focus group guide and questionnaire design available. Next to this we also make all the returned questionnaires available in anonymized form. We do this to make both our data gathering methods and the analysis of our data, repeatable. A final threat to construct validity is that two of the main authors of this paper are also employees of the company. An advantage of this is that the researchers posses insight knowledge and can leverage this to analyze the data more accurately. A disadvantage is that the researchers might not be completely impartial due to their involvement in the setting. Overall, it is our opinion the advantages outweigh the disadvantages.

Finally, there is also a threat to internal validity, because the people that participated in the focus group also participated in the subsequent questionnaire. This could have biased the results due to a learning effect caused by repeated testing.

VII. Conclusions and Future Work

The main contributions of this paper are the answers to the research questions. First, we showed the most important information items for distributed software engineers to receive updates on in tables I, II and III respectively. Subsequently, we researched the mutual importance of these. Firstly, we did this depending on the involvement of the distributed software engineers with the project the information is related to. We recognized three involvement levels: (i) a project an engineer is working on right now, (ii) a project an engineer is part of but not working on right now and (iii) a project an engineer is not part of. The most interesting findings based on this are:

- The more related something is to the context of a distributed software engineer, the higher the value to receive updates on it
- Receiving updates on a project a distributed software engineer is part of, but not currently working on, is still valuable
- Receiving updates on organizational information items is valuable no matter what relation the information has to the project a distributed software engineer is currently working on
- A large and diverse set of information items is very important if they are related to the project a distributed software engineer is currently working on
- Project specific updates of projects in which someone is not a member are considered unimportant

Similarly, we also investigated the mutual importance of the information items when a distributed software engineer is not performing his core activity. We did this for three specific cases: (i) when the engineer is on holiday, (ii) when the engineer is enjoying free time and (iii) when the engineer is performing work-related traveling. The most interesting findings based on this are:

- The importance of updates on the information items is far lower when a distributed software engineer is not performing his core activity
- Updates on project-related information are considered more important during work-related travel than during free time or holiday
- Updates on organizational information are considered equally important during free time, holiday and work-related travel

Following this, we also investigated the relation between the importance of updates on a specific type of information and the frequency it changes, but could not find such a relation. We did present how frequent the information items researched in this study change, as perceived by the participants of the study. We found that most of the investigated information types change weekly, while some change daily and hourly as well.

Concerning future work, it is interesting to repeat the study in more settings and compare the results. We are particularly interested whether results will differ in: (i) larger organizations, (ii) more hierarchical organizations and (iii) teams spread across different time zones and cultures. Finally, we illustrated how the findings in this paper can be applied, by discussing how we intend to apply them ourselves in the development of a Collaborative Development Environment called Iris. Following this implementation we intend to evaluate its value to distributed software engineers in a practical case setting.

References

[1] E. Carmel, *Global software teams: collaborating across borders and time zones*. Upper Saddle River: Prentice Hall PTR, 1999.

[2] J. Herbsleb and D. Moitra, “Guest Editors’ Introduction: Global Software Development,” *IEEE Software*, vol. 18, no. 2, pp. 16–20, 2001.

[3] J. Herbsleb, “Global Software Engineering: The Future of Socio-technical Coordination,” in *Proceedings of the IEEE 2007 Workshop on the Future of Software Engineering*. IEEE Computer Society Press, 2007, pp. 188–198.

[4] The Dieringer Research Group Inc., “Telework Trendlines 2009: A Survey Brief by WorldatWork,” 2009.

[5] K. Schmidt, “The Problem with ‘Awareness’: Introductory Remarks on ‘Awareness in CSCW’,” *Computer Supported Cooperative Work*, vol. 11, no. 3-4, pp. 285 – 298, 2002.

[6] A. Syri, “Tailoring cooperation support through mediators,” in *Proceedings of the 1997 European Conference on Computer Supported Cooperative Work*. Kluwer Academic Publishers, 1997, pp. 157–172.

[7] S. R. Hiltz and M. Turoff, “Structuring computer-mediated communication systems to avoid information overload,” *Communications of the ACM*, vol. 28, no. 7, pp. 680–689, 1985.

[8] M. J. Eppler and J. Mengis, “The concept of information overload: A review of literature from organization science, accounting, marketing, mis, and related disciplines,” *The information society*, vol. 20, no. 5, pp. 325–344, 2004.

[9] P. Dourish and S. Bly, “Portholes: supporting awareness in a distributed work group,” in *Proceedings of the ACM CHI 1992 Conference on Human Factors in Computing Systems*. ACM Press, 1992, pp. 541–547.

[10] K. Schwaber and J. Sutherland, “Scrum guide,” *Scrum Alliance*, 2011.

[11] C. Dentel, M. Nordio, and B. Meyer, “News and notification: Propagating relevant changes to developers,” *Software Engineering Laboratory: Open Source Eiffel Studio, ETH Zürich*, 2012.

[12] J. C. Tang, “Findings from observational studies of collaborative work,” *International Journal of Man-machine studies*, vol. 34, no. 2, pp. 143–160, 1991.

[13] J. T. Biehl, M. Czerwinski, G. Smith, and G. Robertson, “Fastdash: a visual dashboard for fostering awareness in software teams,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2007, pp. 1313–1322.

[14] A. Mockus and J. D. Herbsleb, “Expertise browser: a quantitative approach to identifying expertise,” in *Proceedings of the 24th International Conference on Software Engineering*. ACM, 2002, pp. 503–512.

[15] J. Yew, F. Gibson, and S. Teasley, “Learning by tagging: group knowledge formation in a self-organizing learning community,” in *Proceedings of the 7th international conference on Learning sciences*. International Society of the Learning Sciences, 2006, pp. 1010–1011.

[16] M. Storey, L. Cheng, I. Bull, and P. Rigby, “Shared waypoints and social tagging to support collaboration in software development,” in *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*. ACM, 2006, pp. 195–198.

[17] G. Booch and A. W. Brown, “Collaborative development environments,” *Advances in Computers*, vol. 59, pp. 1–27, 2003.

[18] T. B. Sheridan and W. Ferrell, “Man machine system: Information control and decision models of human performance,” 1974.

[19] L. Hattori and M. Lanza, “Syde: A tool for collaborative software development,” in *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, vol. 2. IEEE, 2010, pp. 235–238.

[20] K. Dullemond and B. van Gameren, “An industrial evaluation of technological support for overhearing conversations in global software engineering,” in *Proceedings of the 2012 International Conference on Global Software Engineering*. IEEE Computer Society Press, 2012, pp. 65–74.

[21] M. J. Rochkind, “The source code control system,” *Software Engineering, IEEE Transactions on*, no. 4, pp. 364–370, 1975.

[22] W. F. Tichy, “Design, implementation, and evaluation of a revision control system,” in *Proceedings of the 6th international conference on Software engineering*. IEEE Computer Society Press, 1982, pp. 58–67.

[23] R. Frost, “Jazz and the eclipse way of collaboration,” *Software, IEEE*, vol. 24, no. 6, pp. 114–117, 2007.

[24] R. Holmes and R. J. Walker, “Promoting developer-specific awareness,” in *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering*. ACM, 2008, pp. 61–64.

[25] G. N. Aranda, A. Vizcaino, R. R. Palacio, and A. L. Moran, “What information would you like to know about your co-worker? a case study,” in *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*. IEEE, 2010, pp. 135–144.

[26] J. Kontio, L. Lehtola, and J. Bragge, “Using the focus group method in software engineering: Obtaining practitioner and user experiences,” *Empirical Software Engineering, International Symposium on*, vol. 0, pp. 271–280, 2004.

[27] J. Langford and D. McDonaugh, *Focus Groups: Supporting Effective Product Development*. Taylor and Francis, 2003.

[28] A. Fink, *How to manage, analyze, and interpret survey data*, 2nd ed. London: Sage, 2003.

[29] R. Likert, “A technique for the measurement of attitudes.” *Archives of psychology*, vol. 22, no. 140, pp. 1–55, 1932.

[30] K. D. Bailey, *Methods of Social Research*. New York: Free Press, 1978.

[31] K. Dullemond, B. van Gameren, and R. van Solingen, “Collaboration should become a first-class citizen in support environments for software engineers,” in *Proceedings of the 2012 International Conference on Collaborative Computing: Networking, Applications and Worksharing*. IEEE, 2012, pp. 398–405.

[32] B. van Gameren, K. Dullemond, and R. van Solingen, “Auto-erecting virtual office walls,” in *Proceedings of the 2012 International Conference on Collaborative Computing: Networking, Applications and Worksharing*. IEEE, 2012, pp. 391–397.