Universal Access in the Information Society (2023) 22:511–524
https://doi.org/10.1007/s10209-021-00862-8

LONG PAPER

Constraining peripheral perception in instant messaging during software development by continuous work context extraction

Nghia Pham van[1] · **Valentino Vranić**[1]

Accepted: 29 November 2021 / Published online: 17 January 2022

Abstract
Colocated software development teams benefit from natural work context building, which occurs mainly thanks to the team members, virtually, being forced to listen to what others are talking about. They absorb the information not by directing their attention to the communication, but by being exposed to it and perceiving it peripherally. The same effect of peripheral perception can be enforced with instant messaging, which is a predominant way of communication in distributed teams. However, forcing team members to observe too many and mostly unrelated message notifications can be distracting and causing unnecessary work interruption. This paper presents an approach and tool that ensure peripheral perception in instant messaging constrained by a continuously extracted work context. This is achieved by maintaining a personal work context from developer activities and using this context to filter instant messages to be displayed. A four-week experiment carried out with one of the teams of seven members in the Team Project course at our university indicates that message filtering based on continuous work context extraction performs better over common channel based filtering (as available in Slack). More precisely, message filtering based on continuous work context extraction decreases work interruption and distraction.

Keywords Peripheral perception · Agile software development · Distributed software development · Distributed teams · Work context · Work interruption · Distraction · Instant messaging

1 Introduction

Effective verbal communication is the key to solving problems of unstable requirements. This can be observed in agile software development, which favors individuals and their interaction over the processes and tools [2]. Personal interaction problems exist in colocated teams, too, but distributed development makes them even more difficult [15]. Distributed teams rely mainly on instant messaging [21], although video conferencing tools and collaborative environments (such as CVE[1]) are also available.

It is important to note that not all communication is direct and explicit. Team members are exposed to what others on the team talk about even if that does not concern or involve them directly. We might say they perceive this peripherally. By involuntarily—and even unconsciously—absorbing this information, they learn the work context and become prepared for highly effective direct communication when needed. Cockburn calls this Osmotic Communication [5] as, like in chemical osmosis, immersed in information, people tend to absorb what they need. However, he also points out that people sometimes need quiet time in isolation to focus, which he calls Cone of Silence [6]. In other words, Osmotic Communication can be overwhelming.

This applies to instant messaging, too. Imagine being forced to read all the messages that occur and the moment they occur. Instant messaging tools, such as Slack, typically provide channels, so that team members can narrow down the communication burden to what they consider interesting to them. However, this is a very rough message filtering not based on personal interests that, moreover, tend to change over time. This is exactly the topic of this paper: how to ensure peripheral perception of instant messages, but to decrease the communication burden. For this, a tool support based on Slack is provided that exposes team members to message notifications, facilitating peripheral perception of

✉ Valentino Vranić
vranic@stuba.sk

[1] Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Bratislava, Slovakia

[1] Collaborative Virtual Environment, http://cve.sourceforge.net/

instant messages, but that also enables constraining the displayed message notifications to only those that are of interest to a given team member.

The rest of the paper is organized as follows. Section 2 shows how instant messaging can be perceived as a peripheral perception enabler in distributed teams. Section 3 explains the principles behind constraining peripheral perception according to the work context, identifies the possibilities of extracting a personal work context from developer activities, and presents the implementation of message filtering based on continuous work context extraction. Section 4 describes the setting of the experiment space, along with the initial, small-scale experiment. Section 5 describes the main, team project experiment, its results, and threats to validity of these results. Section 6 compares the approach proposed in this paper to related work. Section 7 concludes the paper and indicates some possibilities for further work.

2 Instant messaging as a peripheral perception enabler in distributed teams

Instant messaging is the main form of communication in distributed software development teams [21]. Part of the reasons for this lies probably in that it allows people to work in real time and to make and test small changes immediately after making them [20]. This immediacy also allows for efficient scheduling and coordination as instant messages tend to be read quickly.

One of the concerns related to written electronic communication in distributed teams is the level of work interruption and distraction it causes. Instant messaging tends to show a large window on top of other active applications, often accompanied by an animation, flashing, and sound notifications. As Garrett and Danziger [11] say, such notifications can be easily dismissed, hidden, or outright disabled, so they are not as distracting as a colleague coming in person or as a telephone call.

Not every work interruption is perceived as negative. In fact, some work interruptions can have a positive effect by bringing critical information. Actually, instant messaging is not a major source of work interruption, as it accounts only for 5% of it [23]. Furthermore, Garrett and Danziger [11] reported that the use of instant messaging reduces the overall work interruption as the same attributes that create opportunities for work interruption also allow for users to more effectively manage the interruption and that users were more likely to do so than to experience instant messaging as disruptive.

However, if instant messaging is to be used to mimic the ambient conversation, in which colocated team members are commonly immersed, this cannot be done by simply notifying all the team members of all messages, as that would cause too much distraction. Mark et al. [18] discovered that albeit people perform good when interrupted even if the context of the work changes significantly, this comes at a price of increased workload, stress, frustration, time pressure, and effort. Even worse, according to Ophir et al. [22], heavy media multitaskers are more susceptible to interference from irrelevant environmental stimuli and from irrelevant representations in memory, which means they become incapable of distinguishing what is important, and what is not.

3 Constraining peripheral perception according to the work context

A shared context exists when team members have access to the same information and share the same tools, work processes, and work culture. It is difficult to correctly interpret the meaning of information and develop mutual understanding when this knowledge is absent. Hinds and Mortensen [13] describe a shared context as an emergent state that develops in a team. They found that the lack of context can make it more difficult to resolve problems before they devolve into a conflict. Face-to-face communication is appropriate for building this context because of its richness. Electronic written communication, however, is more suited for short and simple exchanges of information within an already built context [27].

According to Santoro et al. [26], the context plays an important role in problem solving domains such as software engineering. In collaborative software development, having access to the context is essential to properly cooperate and coordinate with others. Ko et al. [14] report that the most frequently sought information by software developers is the information that increases the awareness of tasks, artifacts, and coworkers.

A work context can be defined as a more specific subset of the entire project context. A work context is focused solely on the current task or work of the team or individual and includes every piece of information that is relevant to this work. Gutwin et al. [12] describe group awareness as the understanding of who is working with whom, what they are doing, and how one's own actions interact with others' actions.

The main contributors to the work context of a software developer are the current project, developer's intent, and artifacts relevant to the current task. Projects consist of a project plan, list of tasks, stories, bugs and issues, as well as project documentation and information gained from formal project meetings. This information is the main deciding factor for the next task for each developer, and therefore decides the work context that the developer will be concerned with in the immediate future. The developer's intent represents knowledge and information about how the developer intends to approach and accomplish the given task. This information can be different from the more general information

contained in the project plan, as the intent also relies on the personal knowledge and experience of the developer that is assigned to the task. Artifacts created over the course of a task or artifacts that were created previously but are relevant to the current task include diagrams, notes, pictures, or any other artifacts apart from source code.

A team creates and maintains a shared work context. It includes knowledge about the current project, what each team member is currently working on, what are they proficient at, the roles of each team member, and other information relevant to the current project as a whole. The information that is relevant to all team members contributes to the shared work context between them. This context changes over time, but a significant part of it is constant throughout the duration of the project.

In addition to the shared work context in a team, each individual team member also maintains a personal work context. It can be understood as a subset of the shared work context. A personal work context includes information relevant to a developer at the current point of the project, i.e., to the current tasks that the developer is working on. It is this assigned task that the developer is currently working on that is the main contributing factor to the personal work context. A personal work context is much more volatile than the shared work context and changes very quickly over time as the developer switches and accomplishes tasks.

A personal work context can be used as a communication filter, i.e., to filter out irrelevant and distracting information from electronic written communication so to improve chances for the critical and relevant information to reach the developer.

Section 3.1 explains the possibilities for extracting the personal work context from developer activities. Section 3.2 describes the implementation of message filtering based on continuous work context extraction contributed by this paper.

3.1 Extracting the personal work context from developer activities

The prerequisite to using a personal work context is its proper extraction. According to Maalej and Sahm [17], software engineers spend only about half of their time on code creation. Software engineers also use on average five tools for a single task and read or change different artifacts, like source code files, bug reports, or diagrams. In a case study done by Zou and Godfrey [28], it was discovered that during a single task, eight code files are read and six are changed on average. Zou and Godfrey also noted that they only observed files within an integrated development environment and it is very likely that the overall number of used artifacts is even higher. Therefore, not only source code, but also other artifacts that people create or change during the task solving have to be considered as a part of the personal work context.

The information about what source code files developers accessed and about their activity within the source code files can be obtained using developer activity tracking tools. These often come as plugins available for various integrated development environments. By periodically accessing the log files of an activity tracking tool, the information about the source code files that a developer has been working on recently or is currently working on can be extracted. The name and path of the source code files also has implications for the personal work context. In Java, for example, packages are represented by folders. Being larger code structures, packages tend to have descriptive names that can be used to obtain information about what the developer is working on. For example, if a developer is accessing files in the *views/users* folder, i.e., in the *views.users* package, we might conclude that the current task is about adding a front-end to the user management. These package names can be treated as keywords and can be searched for in incoming written communication.

Extracting the information about the personal work context from the contents of the source code files is also a possibility. Berta et al. [3] identified use cases in source code by finding relations between use case steps and method names in source code using a dictionary such as WordNet. A similar approach could be used for extracting personal work context. However, this method does not provide sufficiently precise results for automatic extraction and an expert still has to be involved in the process. This method can be used for extracting keywords from source code files, mainly from method names and comments.

Extracting information from visual artifacts such as diagrams or pictures is more difficult. Outside of using optical character recognition (OCR) tools to extract text from these artifacts, it is difficult to extract information from them. One option is to use the metadata of the artifact, such as the name of the file. File names tend to be descriptive for easier searching and archiving and therefore could be used as keyword as well. This could be achieved by selecting a directory for storing artifacts and regularly scanning this file directory. Having code linked to model artifacts, e.g., to actions in activity diagrams [16], might help with this.

Artifacts tend to be shared between team members to better communicate task related information and are therefore stored on a shared repository such as GitHub or Google Drive. The uploaded files can be used as another source of information about a personal work context.

Version control systems are also a rich source of information. Commit messages in collaborative software development have to be descriptive about the files or functionality that was added or changed. By accessing the commit messages of an individual team member, the information about

tasks that the developer finished or is currently working on can be extracted. Commit logs serve as the records of changes made to the project. Each record or commit includes information such as the developer committing the change, files affected, number of changes, and differences between the old and new versions of the files. Many developers read commit logs in order to stay up to date with what is happening on the project and to watch for changes that may affect their own work [12]. Repositories such as GitHub offer APIs which can be used to obtain this information. Version control systems can also be linked with issue tracking tools or project management tools such as JIRA or ScrumDesk in order to have a direct link between team members, the task assigned to them, and their activity and commits.

Information can also be obtained from written electronic communication of a developer. According to Dullemond [8], a large part of a developer's time is spent on communication and seeking information about recent changes. Therefore, valuable information about a personal work context can be extracted from the communication.

Another possibility to collect personal work context information is to have people provide it explicitly. However, relying on developers to set and update their keywords as they change tasks and contexts is not viable in the long run. While the keywords and context would probably be quite accurate, people could forget to update them making the filter not very reliable. According to Maalej and Sahm [17], 60% of their respondents changed focus on hourly basis. Asking for manual input regularly using periodic reminders could also cause distraction and interruption and the messages could be ignored or turned off.

3.2 Implementing message filtering based on continuous work context extraction

For the purposes of the research reported here, a tool for message filtering based on continuous work context extraction called Indikom was developed (the tool and the user guide are available on GitHub[2]). Work context extraction is facilitated through the use of the Rabbit Eclipse plugin,[3] which captures developer activity in Eclipse, and the Activity Tracker plugin[4] for the JetBrains family of integrated development environments, such as RubyMine, PyCharm, or IntelliJ. The plugin tracks the user activity with respect to the list of files accessed, duration in which they were accessed, as well as activities the of the user such as writing and deleting lines of code. For the purposes of Indikom, only the files that were accessed and the time by which they were open were used. These metrics are periodically collected by the integrated development environment plugin. An update can also be invoked manually. The activity track is stored in an XML file.

Another source of the work context being collected by Indikom is the developer activity on GitHub. Indikom is able to access the profile of the developer and extract commits that the developer pushed to the repository. From these commits, the commit message and the list of changed files can be extracted in order to gain the information about the previous activity of the developer.

After loading the user activity supplied by the activity tracking plugin, Indikom extracts the user activity in the form of the files the user has recently accessed and builds a work context from this information. The names of the files that were accessed and the names of the packages they belong to are split into keywords which are used to define the work context. This process is repeated periodically in the background while Indikom is running, in order to keep the work context relevant and up to date. During each of these periodic updates, the files that were not accessed since the last update are given a lower weight in order to signify that these artifacts might no longer be relevant to the current work context. Once the weight is reduced past a certain threshold, these files are removed from the work context entirely.

In order to increase the accuracy of the work context, the keywords extracted from file and package names have to be analyzed and new keywords have to be created. Indikom uses the Wordnik thesaurus[5] in order to obtain a list of synonyms, word forms, and related words to each keyword extracted from the user activity. This process vastly increases the amount of keywords gathered and therefore increases the accuracy of a message filter that uses these keywords. The entire process of work context extraction is shown in Fig. 1.

Indikom connects to a Slack workspace in as a bot and then extracts messages from the chosen channel. Users are able to choose which channel should be monitored for messages from a drop-down menu. All messages are displayed in the main window, which serves as a history of messages.

All incoming messages are filtered by the work context filter. This filter searches for the keywords of the current work context and displays an incoming message in a transparent window at the bottom right of the screen if the text of the message contains one of the keywords. This notification window persists for a few seconds before disappearing.

Indikom also offers the option to have the messages read out instead of displayed. A text to speech converter is used for this. This can be helpful in situations when the user would prefer to hear the message instead of reading it. The

[2] https://github.com/ThePham/Indikom

[3] https://code.google.com/archive/p/rabbit-eclipse/

[4] https://github.com/dkandalov/activity-tracker

[5] https://www.wordnik.com/

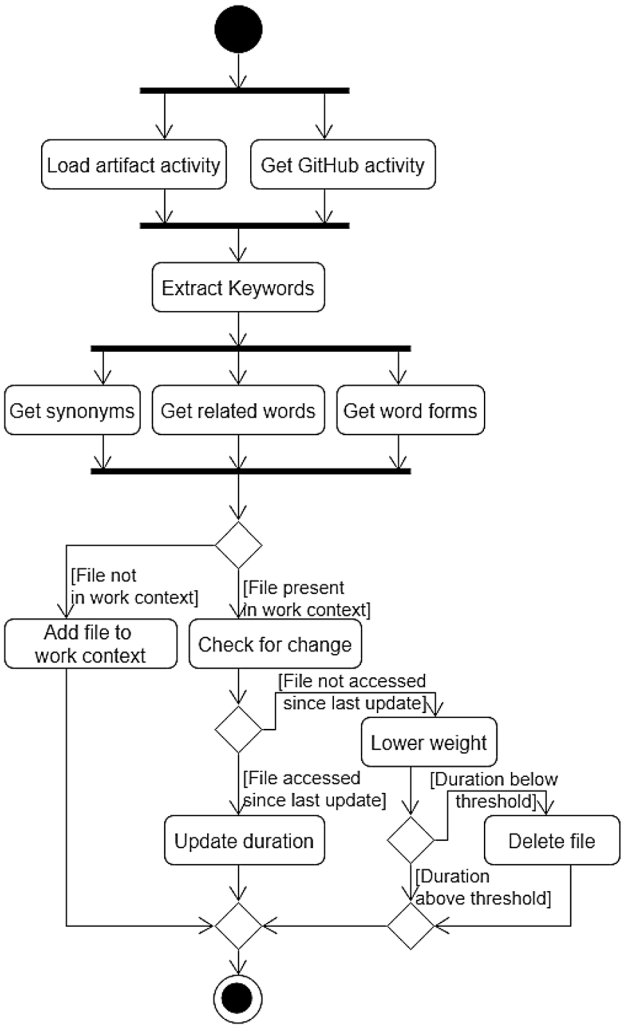

Fig. 1 Work context extraction process as an activity diagram

downside of this approach is that the user has to be able to hear the message being read either through the use of headphones or speakers, and that sound could be more distracting than a visual notification.

Indikom is supplied in the form of an executable JAR file. Upon the first launch, it prompts the user for a workspace token in order to access the Slack workspace and channel that is to be monitored. It also needs to be set up with the location of the log files generated by the activity tracking plugins, API key for the Wordnik thesaurus API, Microsoft's Translator Text service, and GitHub credentials. The configuration is saved locally in a text file and loaded on the next use.

4 Setting the experiment space

In a preparation for an experiment that would enable assessing how message filtering based on continuous work context extraction as proposed in this paper and implemented in Indikom affects work interruption and distraction, which is described in the next section, we first mapped the overall experiment space and run a small-scale experiment.

The experiment space is actually given by the Indikom features. Feature modeling was used to capture these. The corresponding feature diagram showing all the features that can be used to create experiment configurations is presented in Fig. 2.

Experiments can employ different ways of notifying participants of incoming messages. A message filter based on the work context, the channel filtering provided by Slack, or no filtering at all can be used.

The work context sources available for a message filter are the user input, code repository, and developer activity. At least one of these options has to be present in an experiment if the work context filtering is to be used. The developer activity can further be captured by using periodic updates or by manually triggering the update. Forcing the user to

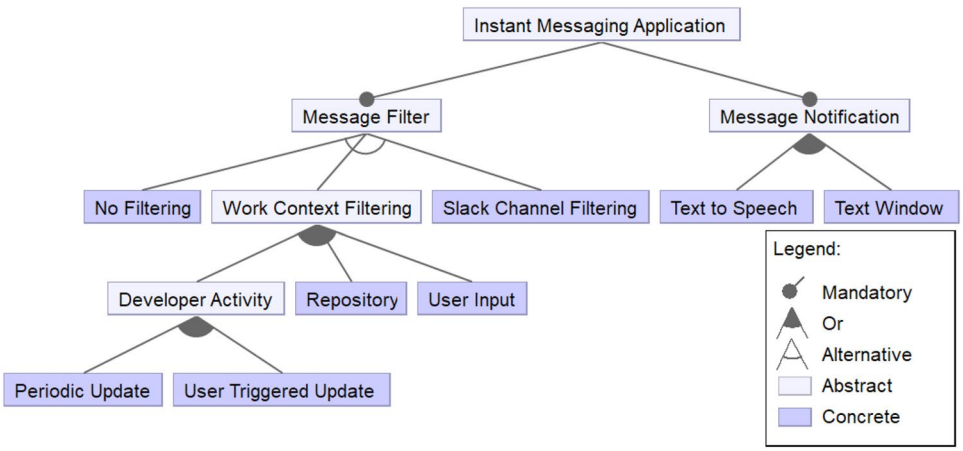

Fig. 2 Indikom feature diagram showing all the features that can be used to create experiment configurations

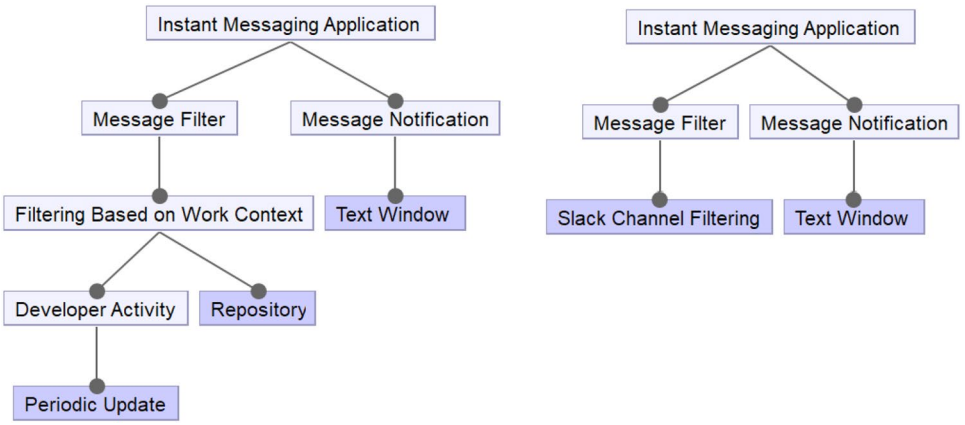

Fig. 3 Indikom configurations featuring message filtering based on continuous work context extraction (left) and common channel based filtering (right)

interact with Indikom during an experiment is not preferred, but can be used in order to evaluate the effectiveness of automatic work context extraction against the user triggering work context updates, as participants will always be best aware of their current work context and when the changes occur. There are two kinds of message notifications: text-to-speech and text window notifications. They can be engaged simultaneously.

A small-scale experiment was carried out with four students of our university with the main purpose of decreasing the technology risk for the actual experiment, which is covered by the next section. All four participants installed Indikom and cloned the project created for the purposes of the experiment. The participants were divided into two pairs and worked on their tasks for ten minutes in order to build a work context before launching Indikom. Subsequently, Indikom updated the work context every ten minutes during the experiment with the entire experiment taking one hour in total. Communication was handled through the use of a Slack workspace. Because the messages sent were in Slovak, a translator was used to translate them into English. The participants were only given general tasks and they were encouraged to discuss the details among them.

Since the number of participants was not big, after the experiment, all the participants were asked about their opinion in a direct and open conversation rather than using a questionnaire. All four participants expressed a positive impression of the approach to extracting and using work context as a message filter implemented in Indikom. They praised that the messages deemed not relevant were not displayed on their screen, while still being preserved in the main window of Indikom, as well as in the Slack client itself so that they had the option to read up on what they missed while working on their own tasks.

Messages that were relevant to the current work were displayed as notifications in the bottom right part of the screen. The participants judged these notifications as not being disruptive, but beneficial to their work. Overall, the participants rated the automatic extraction of the work context and its use as a message filter positively and could imagine using the approach and tool in larger teams and projects. Particular remarks raised by the participants, such as the one that the notification window shifted focus from the other window the participants had open, which caused distraction if a notification was displayed in the middle of writing either code or a message, could be used to improve Indikom.

5 The team project experiment

An experiment was performed in order to compare message filtering based on continuous work context extraction as proposed in this paper and implemented in Indikom and common channel-based filtering as it is available in Slack from the perspective of work interruption and distraction.

Formally, the null hypothesis can be stated as follows:

> The common channel based filtering and message filtering based on continuous work context extraction have an equal effect on work interruption and distraction.

The alternate hypothesis is then as follows:

> Compared to the common channel based filtering, message filtering based on continuous work context extraction decreases work interruption and distraction.

Section 5.1 explains the experiment configuration. Section 5.2 presents and discusses the results. Section 5.3 identifies the threats to validity.

5.1 Experiment configuration

The two configurations of Indikom corresponding to the hypotheses are shown in Fig. 3. The experiment was carried out with seven members of a team participating in the Team Project course within their master's of science level

Table 1 Percentage of relevant messages over the course of one week

Message filtering	Percentage of relevant messages for each participant							Average
Continuous work context extraction	77	84	62	69	72	87	71	74.5%
Common channel	59	66	43	49	40	53	58	52.6%

studies at our university. The team was already working on their project in the term that preceded the experiment and continued to work on it over the course of the experiment. The project was related to the development of a web application using PHP and the Laravel framework.

The initial idea was to employ each configuration for a period of one week. Indeed, in the first week, message filtering based on continuous work context extraction was employed, while in the second week, the common channel-based filtering was employed. However, as the participants were willing to continue with the experiment, we returned to message filtering based on continuous work context extraction for additional two weeks since this was the novel configuration brought by Indikom.

All participants were provided with Indikom and instructions on how to install and use it.

5.1.1 Message tagging

In the first two weeks of the experiment, the participants were asked to tag each on-screen notification in Indikom in order to determine if the message was relevant to their current work context.

5.1.2 Direct observation

A direct observation of the participants at work with Indikom was also carried out during one of the weekly team meetings. The participants were prevented from directly communicating while working on their tasks and were forced to use Indikom in order to communicate and maintain peripheral perception.

5.1.3 Questionnaire

A questionnaire was provided every week in order to collect the data on how much work interruption and distraction Indikom caused (see the appendix).

5.2 Results and discussion

Message tagging. Table 1 shows the percentage of messages that were tagged by experiment participants as relevant to their current work. Both types of filtering were used for one week during the same sprint of the project. Irrelevant messages displayed in notifications are a source of work interruption, so it can be concluded that message filtering based on continuous work context extraction causes significantly less work interruption by successfully filtering out a larger percentage of irrelevant messages, which supports the alternate hypothesis.

Using a two tailed paired t-test at the 0.05 level to compare the two approaches supports rejecting the null hypothesis. The p-value of the paired t-test is 0.0002230, i.e., which means there is only a 0.0223% chance of a type 1 error. Such a small value also supports rejecting the null hypothesis.

5.2.1 Direct observation

Although work interruption occurred during the direct observation of the participants at work with Indikom and participants stopped their work for a few seconds each time a message notification appeared, all participants judged that the messages were mostly relevant and beneficial to their work, making the positive effect on communication be greater than the negative effect of work interruption. The participants often joined in on conversations and were able to answer questions from other team members working on the same tasks. It can, therefore, be concluded that message filtering based on continuous work context extraction as it is implemented in Indikom is an effective way to providing and augmenting peripheral perception in distributed teams and that the benefits of using instant messaging outweigh the work interruption they cause.

5.2.2 Questionnaire

The questionnaire the participants had to take each week included six statements and two questions each of which had to be assessed on a scale 1–5, where 1 meant a strong disagreement or "never" for questions, and 5 meant a strong agreement or "very often" for questions. The questionnaire also included two questions related to whether the participants closed Indikom. The complete answers are available in the appendix.

We depict the average values for each of the statements and questions as bar graphs. Although the averages could not go below 1, the graphs start at 0 in order to better visually distinguish the value of 1. The standard deviations are shown in parentheses by the average weekly values. They ara also depicted as error bars .

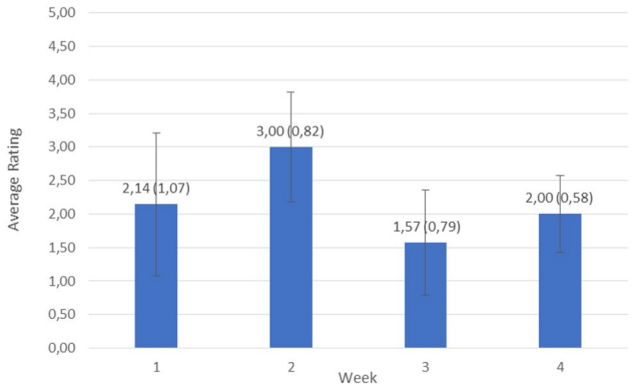

Fig. 4 Average rating for the statement "The message notifications were distracting and interrupted my work" (Q1)

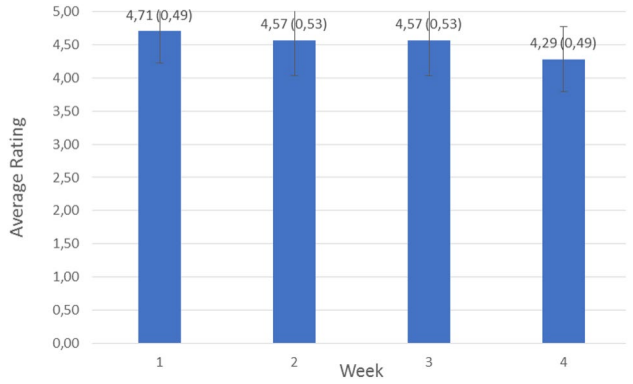

Fig. 5 Average rating for the statement "Message notifications were noticeable and I looked at them when they appeared" (Q2)

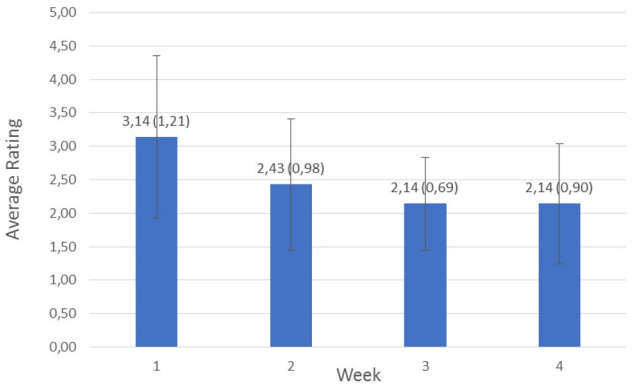

Fig. 6 Average rating for the statement "Message notifications took some time to get used to" (Q3)

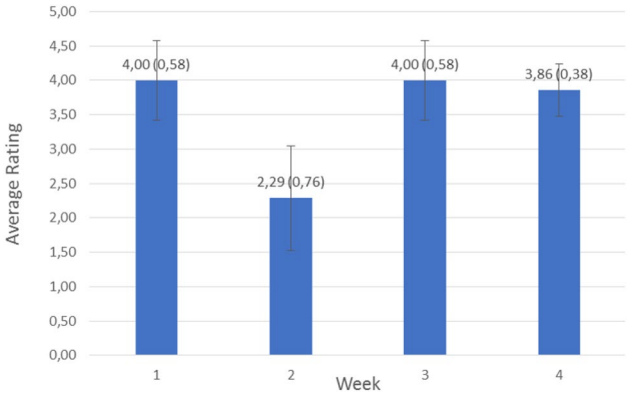

Fig. 7 Average rating for the statement "Message notifications showed me relevant information for my current task" (Q4)

As shown in Fig. 4, the rating for the statement "The message notifications were distracting and interrupted my work" (Q1) also supports the claim that the message notifications were considered not distracting. The common channel-based filtering used in the second week carried a significantly higher level of distraction and interruption, which corresponds to the results of the message tagging. The standard deviations show that compared to the last week, the perception of how distracting message notifications varied somewhat more among the participants during the first three weeks of the experiment. This indicates that developers could get used to message notifications quite quickly.

The average rating for the statement "Message notifications were noticeable and I looked at them when they appeared" (Q2) can be seen in Fig. 5. Virtually all participants considered message notifications as very noticeable and looked at them when they appeared. This is supported by the relatively low standard deviations at this statement.

As shown in Fig. 6, which shows the average ratings for the statement "Message notifications took some time to get used to" (Q3), the participants got used to message notifications quite quickly. The standard deviations are a bit higher, but this may be caused by different perceptions of what does "some time" mean. Nevertheless, as we said, the decreasing standard deviations at statement Q1 also indicate that developers could get used to message notifications quite quickly.

From the average rating for the statement "Message notifications showed me relevant information for my current task" (Q4), it can be concluded that the work context based filtering is able to filter out more irrelevant messages and thus significantly reduce work interruption and distraction compared to the common channel based filtering. Figure 7 shows the difference in the number of relevant messages displayed as perceived by the participants. These results further support the alternate hypothesis and correspond to the results presented in Table 1. This is supported by the standard deviations for this statement, which were quite low for all weeks, with a particularly low value in the last week.

The average rating for the statement "A lot of irrelevant messages were displayed" (Q5) also supports the alternate hypothesis. The results displayed in Fig. 8 show a

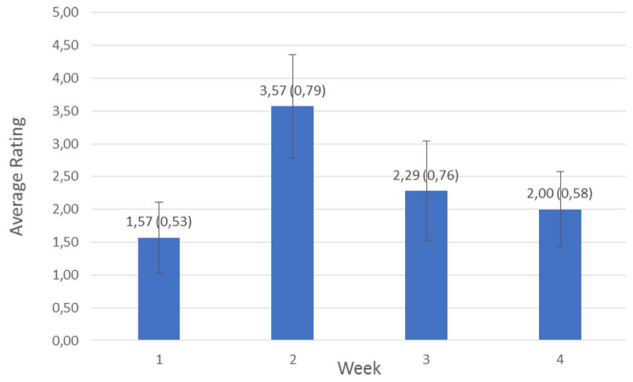

Fig. 8 Average rating for the statement "A lot of irrelevant messages were displayed" (Q5)

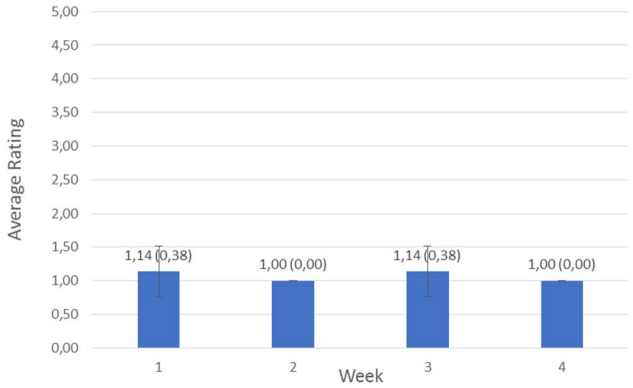

Fig. 9 Average rating for the statement "Developer activity tracking felt like an invasion of privacy" (Q6)

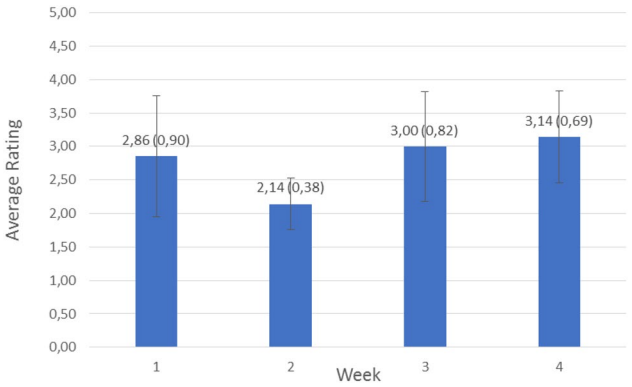

Fig. 10 Average rating for the question "How often did you enter the conversation based on message notifications (1–5)?" (Q9)

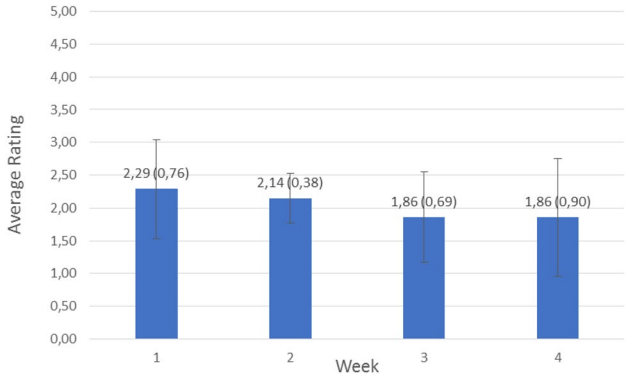

Fig. 11 Average rating for the question "How often did you change something in your current work based on a message notification (1–5)?" (Q10)

significantly higher number of irrelevant messages displayed with the common channel based filtering compared to the work context filtering. In addition, the participants mostly did not perceive the tracking of their activity for the purpose of the work context extraction as an invasion of their privacy, as shown in Fig. 9. The standard deviations for statement Q6 were low and decreasing as the weeks of the experiment passed, indicating that the perception of the relevance of the messages being displayed to the participants was objective. The exceptionally low values of the standard deviation for statement Q6 indicate that the participants consistently perceived the approach as noninvasive with respect to their privacy.

Only one participant answered "yes" to the question "Did you close the program during your work?" (Q7) stating that it was distracting and he needed to focus (Q8).

The participants were also asked about the frequency of them joining a conversation (Q9) and changing something in their current work based on a message notification (Q10). The results can be seen in Figs. 10 and 11. The participants were more reluctant to joining conversations when the common channel based filtering was used. This can be attributed to a higher number of irrelevant messages displayed. The number of times they changed something in their work remained on a similar level throughout the experiment due to the tasks of each individual team member being defined during a team meeting before they started working on the task. The standard deviations vary a bit more with these two questions. For question Q10, the highest standard deviation is the one for the last week. This might be attributed to varying opportunities to join conversations and make modifications based on messages. Longer observation would be necessary to be able to draw some other conclusions from this.

Table 2 shows the correlation between the questionnaire results. Only the numerically graded statements and questions were considered. There is a positive correlation between Q1 and Q2 indicating that a lot of irrelevant messages increases work interruption and distraction. There is a strong negative correlation between Q1 and Q4 indicating that if a participant judged messages as relevant, then

Table 2 Correlation between the questionnaire results

	*Q*1	*Q*2	*Q*3	*Q*4	*Q*5	*Q*6	*Q*9	*Q*10
Q1	1							
Q2	0.410693	1						
Q3	0.536505	0.367475	1					
Q4	−0.79386	−0.31912	−0.32006	1				
Q5	0.602756	0.342224	0.125759	−0.86692	1			
Q6	0.096077	0.258199	0.151407	0.142189	−0.10196	1		
Q9	−0.54444	−0.35089	−0.33982	0.572402	−0.32626	−0.10256	1	
Q10	−0.0101	−0.05638	0.028653	0.084864	−0.01929	0.189242	0.218357	1

the message notifications were not perceived as distracting, which supports the alternate hypothesis. A negative correlation between Q1 and Q9 also indicates that the participants tended to join conversations less often if the messages and notifications were perceived as distracting. There is also a positive correlation between Q4 and Q9 indicating that if the messages displayed were relevant, then participants tended to join a conversation more often.

From the questionnaire results, it can be concluded that peripheral perception was facilitated through the use of instant messaging and on-screen message notifications. Message filtering based on continuous work context extraction was successful in augmenting peripheral perception by filtering out more irrelevant messages compared to the common channel based message filter provided by Slack and therefore was able to further decrease work interruption and distraction. Due to important and relevant information being able to reach the participants, it can be concluded that work and communication effectiveness was increased, but this is difficult to quantify. Overall, the results provide no real grounds for sustaining the null hypothesis, while they all speak in favor of the alternate hypothesis.

5.3 Threats to validity

Due to the nature of the experiment, multiple threats to validity arise and have to be addressed. The experiment was limited in scope, both in terms of time (four weeks), number of participants (one team with seven members), and project (a single project). This is a threat to internal validity that can be decreased by increasing the scope of the experiment to take several months with multiple teams working on different projects.

The common channel-based filtering was used only during one week of the experiment. This is also a threat to internal validity that could have been decreased by having the teams switch between message filtering based on continuous work context extraction and common channel based filtering several times in order to be able to better compare the two. However, this was difficult to the within the Team Project course.

Bias may have been introduced since when participants started employing the second approach, they had already become accustomed to dealing with messages. This threat to internal validity was addressed by employing message filtering based on continuous work context extraction as the approach we considered superior, so that when the participants started using the common channel based filtering they should have been better prepared to it.

By allowing the experiment to run with message filtering based on continuous work context extraction for additional two weeks, we also introduced a threat to internal validity. However, this did not affect message tagging, as that ran only for two first weeks.

Another threat to internal validity is the absence of a control group during the experiment. A control group was not possible due to the small number of participants.

The use of the Wordnik thesaurus represents yet another threat to internal validity. This dictionary is a general purpose English dictionary, which does not take into account the context of software development. The keywords gained by using the dictionary can, therefore, be inaccurate. Using natural language processing and a corpus of software engineering related documents could alleviate this problem, but this is outside the scope of this work. This dictionary was chosen due to the ability of its API to return a list of related words and word forms, unlike other online dictionaries that only supported synonym lookup.

All participants being students of the same university is an external threat to validity since this means they are all of the same age and education background. A more varied group of experiment participants would result in data that is more representative with respect to the assumed application of the approach in commercial projects.

All participants being students, and not software development professionals, might be considered as another external threat to validity. While this is a highly debatable issue [9, 10], we might add that our students are commonly employed as software development professionals, which is in particular true for our master's of science level studies. Furthermore, the experiment was targeting distributed software development. In such a setting, one might expect similar working

conditions for both software development professionals and students.

6 Related work

Calefato et al. [4] developed an extension for Visual Studio in order to increase social awareness in a team. Their approach was to aggregate the content from multiple social media into the developer's workspace in order to build trust among team members with the information from social media working as a surrogate for social awareness gained during informal communication. The approach proposed here does not take into account informal communication unrelated to work nor any communication occurring outside official communication channels. However, it can be extended to take into account this important part of peripheral perception.

Röcker [25] used an ambient display to indicate the presence of members in a distributed team. His approach consists of displaying patterns of light on a special wall to indicate the presence of distant team members and the communication between them. He notes that this approach has positive effects on workplace awareness and group communication. This method, however, is only suitable for teams that are not fully distributed because of the requirement of a special wall to indicate the information. The approach proposed here is limited to verbal communication, but peripheral perception is much broader and should take into account also visual stimuli.

Marx and Schmandt [19] reported a dynamic personalized message filtering system capable of effective message prioritization based on current user interests. These actually constitute a general work context, which is picked up regularly (usually on hourly basis) from the calendar, e-mail messages, phone tags, and such, generating a set of regular expressions that are then used to filter messages. This is a similar way of maintaining a work context as in the approach proposed in this paper.

More sophisticated natural language processing approaches, such as those proposed by Adams and Martell [1] or Cooper et al. [7], could be used for keyword extraction. Although the approach proposed here goes beyond simple synonyms by using related words provided by the Wordnik thesaurus, it could benefit from contemporary natural language processing approaches.

Pejovic and Musolesi [24] report a library for Android based mobile devices that can postpone work interruptions to the most convenient moments. The actual communication applications can be registered to be notified of such moments. The moments convenient for work interruption are identified using machine learning to assess the changes in selected mobile context indicators, namely GPS location, accelerometer readings, Bluetooth fingerprints, and WiFi fingerprints (captured by another application), which are considered to constitute the user context, at the time of the notification and user response. Although the actual context here is specific for mobile devices and not related to software development, the idea could be applied to software development. Instead of the mobile context indicators, the work context should be considered there. In a way, postponing work interruption could be used as complementary to the approach proposed in this paper to further postpone instant messages identified as relevant and expose the developer to them at the most convenient moments.

7 Conclusions and further work

Colocated software development teams benefit from natural work context building, which occurs mainly thanks to the team members, virtually, being forced to listen to what others are talking about. They absorb the information not by directing their attention to the communication, but by being exposed to it and perceiving it peripherally. The same effect of peripheral perception can be enforced with instant messaging, which is a predominant way of communication in distributed teams. However, forcing team members to observe too many and mostly unrelated message notifications can be distracting and causing unnecessary work interruption.

This paper presents an approach and tool—called Indikom—that ensure peripheral perception in instant messaging constrained by a continuously extracted work context. This is achieved by maintaining a personal work context from developer activities and using this context to filter instant messages to be displayed. Specifically, Indikom uses dedicated plugins for Eclipse and the JetBrains family of integrated development environments to extract the list of files accessed, duration in which they were accessed, as well as activities the of the user such as writing and deleting lines of code. It also extracts the developer activity from GitHub. It uses the Wordnik thesaurus in order to better cover the keywords that define the personal work context. Indikom connects to a Slack workspace to extract messages. It displays the notifications of incoming messages in a transparent window if they contain one of the keywords.

A four-week experiment carried out with one of the teams of seven members in the Team Project course at our university indicates that message filtering based on continuous work context extraction performs better over common channel based filtering (as available in Slack). More precisely, message filtering based on continuous work context extraction decreased work interruption and distraction.

In the future, a more extensive experiment in terms of time and participants (their number and variety of

Table 3 Questionnaire answers

Week	Member	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q9	Q10
1	1	2	4	4	4	2	1	no	3	3
	2	1	5	3	4	2	1	no	4	2
	3	3	5	4	4	1	2	yes	2	2
	4	2	5	2	4	2	1	no	3	3
	5	4	5	5	3	2	1	no	2	2
	6	1	4	2	5	1	1	no	4	3
	7	2	5	2	4	1	1	no	2	1
2	1	3	4	4	2	2	3	no	2	2
	2	4	5	3	2	4	1	no	2	2
	3	3	5	3	2	4	1	no	3	3
	4	2	4	1	3	3	1	no	2	2
	5	2	5	2	3	3	1	no	2	2
	6	4	5	4	1	5	1	no	2	2
	7	3	4	2	3	3	1	no	2	2
3	1	1	4	3	4	2	1	no	3	2
	2	1	4	2	5	1	1	no	4	2
	3	1	4	2	4	2	1	no	3	1
	4	2	5	1	4	3	2	no	4	2
	5	2	5	2	4	3	1	no	3	3
	6	3	5	3	3	3	1	no	2	1
	7	1	5	2	4	2	1	no	2	2
4	1	2	4	2	4	1	1	no	3	3
	2	2	5	3	4	2	1	no	2	2
	3	2	4	2	4	2	1	no	4	1
	4	3	5	3	3	3	1	no	3	1
	5	2	4	1	4	2	1	no	4	3
	6	2	4	3	4	2	1	no	3	2
	7	1	4	1	4	2	1	no	3	1

background) could be performed in order to validate the results obtained in the experiments reported here. Using eye tracking and EEG devices would also allow to assess more objectively how the participants perceived the message notifications and how distracted they were by them. The use of visual stimuli, such as periodical displaying of the desktop or windows of development tools, could further enhance peripheral perception in distributed teams by mimicking the way colocated team members accidentally spot each other's monitors.

Questionnaire and answers

A weekly questionnaire on how much work interruption and distraction Indikom caused consisted of the following questions:

Q1 The message notifications were distracting and interrupted my work (1–5).
Q2 Message notifications were noticeable and I looked at them when they appeared (1–5).
Q3 Message notifications took some time to get used to (1–5).
Q4 Message notifications showed me relevant information for my current task (1–5).
Q5 A lot of irrelevant messages were displayed (1–5).
Q6 Developer activity tracking felt like an invasion of privacy (1–5).
Q7 Did you close the program during your work (yes/no)?
Q8 If you answered "yes" to [Q7, explain why (1–5).
Q9 How often did you enter the conversation based on message notifications (1–5, 1 = not at all, 5 = very often)?
Q10 How often did you change something in your current work based on a message notification (1–5, 1 = not at all, 5 = very often)?

Table 3 shows the questionnaire answers. Each row represents an answer provided by one participant. A Likert-type scale was used in most questions, with 1 representing the least agreement with the statement, and 5 representing a full

agreement. Only one participant answered "yes" to the question "Did you close the program during your work?" (Q7) stating that it was distracting an he needed to focus (Q8).

Acknowledgements The work reported here was supported by the Scientific Grant Agency of Slovak Republic (VEGA) under grant No. VG 1/0759/19, Slovak Research and Development Agency under the contract No. APVV-16-0213, and Operational Programme Integrated Infrastructure for the project Research and Development of Software Solution with the Application of Blockchain Technology in the Field of International Rail and Container Transport of Goods (ITMS: 313022V816), co-funded by the European Regional Development Fund (ERDF).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval The study involved monitoring students carrying out their work tasks, which were a part of their study duties. Since students are normally monitored, this study did not introduce any additional ethical burden. All the results have been anonymized. All students agreed to participate in the study and were fully aware of the monitoring and its purpose.

References

1. Adams, P.H., Martell, C.H.: Topic detection and extraction in chat. In: Proceedings of 2008 IEEE International Conference on Semantic Computing, ICSC '08. IEEE, Santa Clara, CA, USA (2008)
2. Beck, K., et al.: Manifesto for agile software development. http://agilemanifesto.org/ (2001)
3. Berta, P., Bystrický, M., Krempaský, M., Vranić, V.: Employing issues and commits for in-code sentence based use case identification and remodularization. In: Proceedings of 5th European Conference on the Engineering of Computer-Based Systems, ECBS 2017. ACM, Larnaca, Cyprus (2017)
4. Calefato, F., Lanubile, F., Sanitate, N., Santoro, G.: Augmenting social awareness in a collaborative development environment. In: Proceedings of 2012 5th International Workshop on Co-operative and Human Aspects of Software Engineering, CHASE 2012. IEEE, Zurich, Switzerland (2012)
5. Cockburn, A.: Agile software development: the cooperative game, 2nd edn. Addison-Wesley, USA (2006)
6. Cockburn, A.: The cone of silence and related project management strategies. http://web.archive.org/web/20170613023457/http://alistair.cockburn.us/The+cone+of+silence+and+related+project+management+strategies (2008)
7. Cooper, R., Ali, S., Bi, C.: Extracting information from short messages. In: Proceedings of 10th International Conference on Application of Natural Language to Information Systems, NLDB 2005, LNCS 3513. Springer, Alicante, Spain (2005)
8. Dullemond, K., van Gameren, B., van Solingen, R.: Supporting distributed software engineering in a fully distributed organization. In: Proceedings of 5th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2012, ICSE 2012 Workshop. IEEE, Zurich, Switzerland (2012)
9. Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A., Oivo, M.: Empirical software engineering experts on the use of students and professionals in experiments. Empir. Softw. Eng. **23**, 452–489 (2018)
10. Feldt, R., Zimmermann, T., Bergersen, G.R., Falessi, D., Jedlitschka, A., Juristo, N., Münch, J., Oivo, M., Runeson, P., Shepperd, M., Sjøberg, D.I.K., Turhan, B.: Four commentaries on the use of students and professionals in empirical software engineering experiments. Empir. Softw. Eng. **23**, 3801–3820 (2018)
11. Garrett, R.K., Danziger, J.N.: IM = interruption management? instant messaging and disruption in the workplace. J. Computer-Mediated Commun. **13**(1), 23–42 (2007)
12. Gutwin, C., Penner, R., Schneider, K.: Group awareness in distributed software development. In: Proceedings of 2004 ACM Conference on Computer Supported Cooperative Work, CSCW '04. ACM, Chicago, IL, USA (2004)
13. Hinds, P.J., Mortensen, M.: Understanding conflict in geographically distributed teams: the moderating effects of shared identity, shared context, and spontaneous communication. Organ. Sci. **16**(3), 290–307 (2005)
14. Ko, A.J., DeLine, R., Venolia, G.: Information needs in collocated software development teams. In: Proceedings of 29th International Conference on Software Engineering, ICSE'07. IEEE, Minneapolis, MN, USA (2007)
15. Korkala, M., Abrahamsson, P.: Communication in distributed agile development: A case study. In: Proceedings of 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2007. IEEE, Lübeck, Germany (2007)
16. Lang, J., Spišák, D.: Activity diagram as an orientation catalyst within source code. Acta Polytechnica Hungarica **18**(3), 127–146 (2021)
17. Maalej, W., Sahm, A.: Assisting engineers in switching artifacts by using task semantic and interaction history. In: Proceedings of 2nd International Workshop on Recommendation Systems for Software Engineering, RSSE '10. ACM, Cape Town, South Africa (2010)
18. Mark, G., Gudith, D., Klocke, U.: The cost of interrupted work: More speed and stress. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '08. ACM, Florence, Italy (2008)
19. Marx, M., Schmandt, C.: CLUES: Dynamic personalized message filtering. In: Proceedings of 1996 ACM Conference on Computer Supported Cooperative Work. ACM, Boston, Massachusetts, USA (1996)
20. Nardi, B.A., Whittaker, S., Bradner, E.: Interaction and outeraction: Instant messaging in action. In: Proceedings of 2000 ACM Conference on Computer Supported Cooperative Work, CSCW '00. ACM, Philadelphia, Pennsylvania, USA (2000)
21. Niinimaki, T.: Face-to-face, email and instant messaging in distributed agile software development project. In: 2011 IEEE 6th International Conference on Global Software Engineering, ICGSE '11. IEEE, Helsinki, Finland (2011)

22. Ophir, E., Nass, C., Wagner, A.D.: Cognitive control in media multitaskers. Proc. Nat. Acad. Sci. (PNAS) **106**(37), 370–379 (2009)
23. Ou, C.X.J., Davison, R.M.: Interactive or interruptive? instant messaging at work. Decis. Support Syst. **52**(1), 61–72 (2011)
24. Pejovic, V., Musolesi, M.: InterruptMe: Designing intelligent prompting mechanisms for pervasive applications. In: Proceedings of 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '14. ACM, Seattle, Washington (2014)
25. Röcker, C., Prante, T., Streitz, N.A., van Alphen, D.: Using ambient displays and smart artefacts to support community interaction in distributed teams. In: Proceedings of OZCHI Conference 2004. Wollongong, NSW, Australia (2004)
26. Santoro, F.M., Brezillon, P., de Araujo, R.M.: Management of shared context dynamics in software design. In: Proceedings of 9th International Conference on Computer Supported Cooperative Work in Design. IEEE, Coventry, UK (2005)
27. Zack, M.H.: Electronic messaging and communication effectiveness in an ongoing work group. Inf. Manage. **26**(4), 231–241 (1994)
28. Zou, L., Godfrey, M.W.: An industrial case study of program artifacts viewed during maintenance tasks. In: Proceedings of 13th Working Conference on Reverse Engineering, WCRE '06. IEEE, Benevento, Italy (2006)