Two Sides of the Same Coin: Software Developers' Perceptions of Task Switching and Task Interruption

Zahra Shakeri Hossein Abad*, Mohammad Noaeen[†], Didar Zowghi[‡], Behrouz H. Far[†], Ken Barker*

* Department of Computer Science, University of Calgary, Canada, {zshakeri, kbarker}@ucalgary.ca

[†] Department of Electrical and Computer Engineering, University of Calgary, Canada, {mohammad.noaeen, far}@ucalgary.ca

[‡] Faculty of Engineering and IT, University of Technology Sydney, Australia, didar.zowghi@uts.edu.au

ABSTRACT

In the constantly evolving world of software development, switching back and forth between tasks has become the norm. While task switching often allows developers to perform tasks effectively and may increase creativity via the flexible pathway, there are also consequences to frequent task-switching. For high-momentum tasks like software development, "flow", the highly productive state of concentration, is paramount. Each switch distracts the developers' flow, requiring them to switch mental state and an additional immersion period to get back into the flow. However, the wasted time due to time fragmentation caused by task switching is largely invisible and unnoticed by developers and managers. We conducted a survey with 141 software developers to investigate their perceptions of differences between task switching and task interruption and to explore whether they perceive task switchings as disruptive as interruptions. We found that practitioners perceive considerable similarities between the disruptiveness of task switching (either planned or unplanned) and random interruptions. The high level of cognitive cost and low performance are the main consequences of task switching articulated by our respondents. Our findings broaden the understanding of flow change among software practitioners in terms of the characteristics and categories of disruptive switches as well as the consequences of interruptions caused by daily meetings.

KEYWORDS

Task switching, task interruption, performance, stand-up meeting

1 INTRODUCTION

Frequently, software development organizations are approving more projects to do than they have resources available to perform them. Thus, they need to assign software developers to multiple projects simultaneously to be able to handle all of the on-going projects. While this might seem a great solution, it overlooks the cost of "task switching" and time fragmentation caused by these switches. DeMarco and Lister [9] define "flow" as a must-have for high-momentum tasks with a high level of cognitive demand involvement such as software development tasks. They also argue that once developers are locked in their current task, for each switch they need at least 15 minutes of concentration to be able to get back into the flow of the primary task. During this immersion period, they are very vulnerable to interruptions and environmental noise and are not really doing work [9]. Parnin and Rugaber [13] studied 10,000 programming sessions from 86 programmers and found that

a developer's work, in a typical day, is cleaved into many short fragments, with a significant amount of time (i.e. 15-30 minutes) required to attain the flow before resuming switched tasks.

However, as the cost of fragmented time and frequent task switching is largely invisible, software practitioners, particularly in management positions are generally thought to be unconcerned about these costs and believe that as long as a flow change (e.g. task switchings, daily meetings) is pre-planned and necessary, it does not have any negative impact on developers' performance and productivity [9]. To further investigate this perception, in this paper we gather data about software practitioners' perceptions of task switchings (either planned or unplanned) and interruptions through a survey of 141 professional software developers. Our analysis of the survey responses found that developers perceive that "task switching" and "interruptions" have similar consequences on their productivity and the cognitive cost caused by frequent task switching negatively impacts their performance in terms of increasing their error rate and the required time for completing the switched tasks.

Moreover, as motivated by our industry partner and as previous studies have revealed that "daily meetings" are yet another form of interruption in workplace [10, 16], we were interested to get software practitioners' views of the extent of this interruption. From this data, we found that the majority of developers perceive these meetings to take up more time than other interruptions and that they negatively impact their performance. Moreover, from this data, we gained some insight into the efficient timing of these meetings to help developers better manage their daily work flow.

2 BACKGROUND AND RELATED WORK

In this section, we describe basic concepts required to interpret the data obtained from survey respondents, followed by an overview of the related work.

2.1 Terminology

Task switching is commonly considered as the act of toggling between separate tasks with different mental sets, attending to each independently [17]. A *task interruption* can be defined as any event that briefly shifts the attention of the subject from the on-going task towards some secondary external events [7].

Altmann and Trafton [6] have posited a theory of "memory for goals" (called ACT-R) which explains the process of task switching in terms of general memory mechanisms of activation and associative priming. The central idea of this theory is that when people switch to a new task, the "goal" (i.e. what the current task is) of the new task and its associated "problem state" (i.e. the information required for performing this task [14]) must be strengthened in memory, to the point where their activation rises above existing

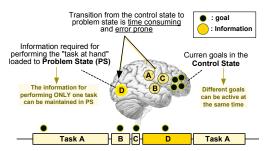


Figure 1: Clarification of the concepts of "task switching", "problem state" and "goal memory". ©Shakeri H.A et al.

goals. Goals and associated information for achieving these goals are maintained in distinct areas of the brain. As illustrated in Figure 1, following the ACT-R theory, while human's cognition allows multiple goals to be active at the same time, only one problem state can be maintained at a time. Figure 1 shows an example of a sequential task switching and the general memory mechanism of this process. Once a developer switches from Task A to Task B, the problem state (required information for performing this task) of Task A will be faded away (light yellow in this figure) and will be transferred from the "working memory" to the "declarative memory". The same process will happen to nested switches: $B \to C \to D$. When the developer is working on Task D, the problem state maintains only the information required for performing this task and the problem states associated with tasks A, B, and C have to be swapped out when there is a switch between tasks. At the same time, the control state containing goals and information will decay according to the memory mechanisms incorporated into ACT-R [6]. Thus, the required information for resuming a switched task that has been suspended and has fallen out of the active use will require more time and effort to recall. This process is not always successful; errors may occur when restoring the mental process back to the primary task.

2.2 Related Work

In this section, we discuss related work that investigated the problem of task switching and interruptions in software development projects and evaluated their findings using industrial datasets.

Abad et al. [2, 3] conducted a set of experiments to evaluate the disruptiveness of task switching and interruptions in Requirements Engineering (RE) tasks to identify factors which make these interruptions more disruptive. They found that RE tasks are more vulnerable to the consequences of interruptions and several factors such as the source of the interruption (i.e. self/external), the granularity of the interrupted task, and the temporal aspects of the switching process (e.g. task's progress status) may impact the disruptiveness. Vasilescu et al. [18] studied the effects of task switchings on developers' productivity. They found that the rate and the number of projects in a task switching process are influential factors on developers' productivity. They also surveyed developers to understand the main reasons for, and perceptions of, multitaskings in software development. Participants of this study described the interrelationships and dependencies between projects as the most common reason for their task switchings.

Regarding resumption strategies, Parnin and Rugaber [13] conducted an analysis on 86 programmers to understand the various

strategies and coping mechanisms that developers need to manage interrupted programming tasks. They found that only a small percentage of interrupted programming tasks were resumed in less than a minute. Chong and Siino [8] compared interruption patterns among paired and solo programmers. Their study indicates significant differences between the pair programmers and solo programmers in terms of the length, type, time, context and strategies for handling task interruptions. They proposed that as a substantial number of interruptions are self-initiated, working in pairs may have potential support for interruption handling. Likewise, Stray et al. [16] conducted a grounded theory study of 12 agile teams to explore developers' perceptions of daily stand-up meetings. They found that one of the prominent negative attitudes toward these meetings is that they are considered as an interruption to daily tasks and often occupy too much time relative to the gains from the meetings. While the past research provided a wealth of insight on task switching and interruptions, we could not find any study that investigated practitioners' perceptions of "task switching" and "interruption" as two distinct concepts.

3 RESEARCH DESIGN

In this study, we aim to address the following research questions:

- RQ1- Do software practitioners have a shared understanding of what " task interruption" means?
- **RQ2-** Do software practitioners consider "task switching" as disruptive as task interruptions?
- **RQ3-** Do software practitioners consider switching to "daily standup meetings" as disruptive as a task interruption?

3.1 Survey Design and Participants

To address our RQs, we designed and implemented an online survey, using Survey Monkey, to investigate software developers' perceptions and experiences of the disruptiveness of task interruptions and task switchings. The survey contained 15 questions including multiple choice, Likert scale, and open-ended questions: 5 on the practitioners' background and experience, 4 on perceptions of task switching and task interruptions, 3 on perceptions of the disruptiveness of daily meetings, and 3 on the management of task interruptions in a working environment.

We sent the online survey to professional software developers working at eight software development companies of various sizes (e.g. Microsoft, Tableau Software, Ericsson, Bosch, Cisco, and CMG). To incentivize participants for high-quality data, we held a raffle for the on-line participants to win two \$50 Amazon gift certificates.

We collected 141 valid answers from 10 countries, the highest population coming from the United States and Hungary with 48% and 21% of participants, respectively. Of 141 respondents, 80% were male and 20% were female, 106 (75%) reported the size of their company greater than 1000 employees, 12 (9%) between 100 and 1000, 23 (16%) less than 100. The average professional software development experience per participant was 10.9 (range 1 to 40) years. The primary work area of all participants was development: 93 (66%) listed their job as a programmer, 21 (15%) as a software architect, 17 (12%) as a tester, 7 (5%) as a project manager and 3 (2%) as a requirements engineer. On an average day, 62 (44%) contribute

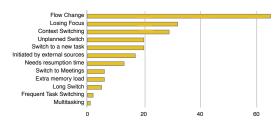


Figure 2: Coding frequency for open-ended question about the concept of "task interruption"

to one project, 38 (27%) to two, 23 (16%) to three, and 18 (13%) to four or more projects.

3.2 Analysis

For the open-ended questions, we iterated through these questions using open coding, axial coding, and selective coding (i.e. the grounded theory methods [5]). To implement this process and to coalesce all of the references, we used NVivo [11], a qualitative analysis software package to code and analyze qualitative data. Additionally, to assess the correlation between participants' responses to survey questions, we used Spearman's rank test and considered $|\rho| \geq 0.50$ with p-value < 0.05 as a strong significant correlation coefficient. Following the suggestion by Kitchenham et al. [12], to compare the distribution of responses between different categories of respondents, we used kernel density plot instead of boxplots and used Skew and Kurtosis scores [15] to interpret these plots.

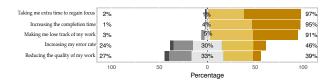
4 RESULTS AND DISCUSSION

In this section, we detail and discuss the results of our investigation related to each research question (RQ1-3).

4.1 RQ1- Defining Task Interruption

To answer this RQ, we asked respondents three open-ended questions to probe whether there was a shared understanding of what a "task interruption" is. The first two authors coded each response with at least one and at most five codes from a list of codes extracted during the first iteration of the coding process.

"Flow change" and "losing focus" (Figure 2) are the two most commonly occurring codes on the open-ended question about the concept of task interruption. 80 (57%) stated that a task interruption would be anything that breaks the flow of their thought, and distracts them from the current work. 40 (28%) defined task interruption as any event that disrupts their focus on the "task at hand" and prevents them from working steadily on a single (or a set of) problem(s) in code: "A task interruption would be anything that takes my focus away from accomplishing the task goal. Particularly when I'm at a point in the task that I've cached a sufficient amount of information about the current state of the problem at hand in my memory. The interruption usually resets my cache and I need to start the task a few steps back from where I left off." Switching to a different task with a different context (context switching) and averting attention to another task (new task) are the next most frequent characteristics we noted for task interruption (context switching: 36 [26%]; new task: 25 [18%]): "Task interruption for me is something that requires context switching, that takes time and diverts my attention from the task I was working on, leading to an emptying and reloading of my



Response Strongly Disagree Disagree Neutral Agree Strongly Agree
Figure 3: RQ1- Perceived consequences of task interruption

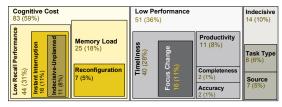


Figure 4: RQ2- Main reasons provided by respondents for considering task switching as interruption.

brain's RAM". This is consistent with the general memory mechanism of task switching we illustrated in Figure 1, following the ACT-R and Memory-of-Goals theories [6, 14].

25 (18%) of respondents defined a task interruption as a spontaneous change in their work context, any unplanned activity which requires changing the thinking from the current task to another, such as coworkers coming to ask a question, instant messaging that requires a response, emails, or meetings. Apart from the context or the source of the interruption, some respondents characterized a task interruption as an event that requires restoring the mental process back to a given task: "When I have to stop doing something, and when I come back to it I have to spend at least a minute re-acquainting myself to the work". Of 141 responses to the statement "A disruptive task interruption negatively impacts my performance after resuming the primary task by ...", 97% agreed or strongly agreed that "the need for extra time to regain the focus", 95% that "the need for extra time to complete the task", and 91% that "losing the track of work" are the consequences of interruptions on their performance (Figure 3). Comparing to the open-ended question on the concept of task interruption, this suggests that there is a widespread agreement on characteristics of task interruption among respondents, including "flow change", "lose focus", and "timeliness". Increasing the error rate and reducing the work quality are next in the list with each of them being 46% and 39% of agreement, respectively.

4.2 RQ2-Task Interruption vs Task Switching

When asked the participants "Do you consider task switchings as a type of interruption?", 115 (81%) participants answered "Yes". We categorized the extracted concepts (during the coding process) from the responses to this question based on their similarities and identified the following main characteristics that make "task switchings" as disruptive as "task interruptions":

4.2.1 Cognitive Cost. 83 (59%) of the participants stated that task switchings, no matter if they are planned or ad-hoc, are a type of interruption because they cause an extra cognitive load: "Definitely it is an interruption. Focusing on one task and finishing it is by far better than trying to fix everything at the same time. Focus/Concentration goes out of the window when trying to fix several different problems at the same time, be it in Software Development or in the kitchen while cooking several meals, eventually, something is bound to burn". As illustrated in Figure 4, we categorized the

codes in the "cognitive cost" category based on their similarities and identified two main reasons of cognitive costs caused by task switching and were mentioned by our participants:

Low Recall Performance— 44 (31%) participants stated that switching to a new task would make their short-term memory need to "forget" about the interrupted task's context, so they can focus on the new information required to perform the new task. Thus, collecting thoughts to get to where they left off can be disruptive and time-consuming. For example, one participant stated: "When working on a task, I tend to become cognitively involved in that one task. So at that time in the moment, I have a deep understanding of the task at hand. But when I am required to task switch I lose that intimacy with the task I am working on. This hurts because when I come back to that task I spend some time getting back into that state of deep understanding". 16 (11%) of the participants stated that task switching negatively impacts the recall performance if the switching request is unplanned (spontaneous) and happens randomly and so often (e.g. having to handle an incoming priority question, a support request from a tester), as in: "Task switching requires some sort of wrap up on the current task and switching focus on a different topic that requires proper attention. If this occurs in an ad-hoc way, it can be considered as a type of interruption".

Memory Load— Extra memory load was perceived by 25 (18%) of the respondents as a source of cognitive cost which makes task switchings as disruptive as task interruptions: "If I turn my attention away from my code, even for a few seconds, I'll need a significant amount of time to get back to focus". Among these respondents, 7 (5%) commented on the cognitive cost of the resumption process. For example: "It takes time to build up the context for any given task. To shelve edited files and re-build the code-base for the given task takes time and effort. There is an upfront cost to starting any task".

4.2.2 Low Performance. In addition to cognitive cost, as a factor which negatively impacts the disruptiveness of task switchings, 51 (36%) respondents stated that "low performance" caused by losing focus is a consequence of task switchings, as in: "since most tasks differ in their context, switching between tasks decreases performance by forcing loading and unloading of context per switch". We categorized responses in this category to the following main sub-categories which cause a low performance after resuming the switched task:

Timeliness— 40 (28%) of the respondents stated that task switching is time-consuming and often does not increase performance to the point of justifying the time cost involved in the switching process: "There's a good amount of ramp-up time in each switch. It takes approximately 5-20 minutes to get into the flow state on the task at hand every time there is a switch". 16 (40%) of these respondents (11% of total) stated that task switching is an interruption on their thoughts (from one task to another) which causes them to lose focus and reduces their overall performance: "I like to focus on what I'm doing until it's done. If I task switch, it causes me to interrupt my workflow and turns my attention away from my code, even for a few seconds, I'll need a significant amount of time to get back to focus".

Productivity— 11(8%) of the participants stated that they are more productive if they are doing only one thing at a time: "I find myself to be most productive and creative when I can be "in the groove" of a single piece of work for more than a day."

Completeness and Accuracy— For each of completeness and accuracy categories we received two responses which indicate that task switching might negatively impact the completeness of the primary task: "At times the interruption makes me forget to go back to the previous task and finish it", or might increase the error rate: "When I have tried to perform two tasks concurrently, switching between them, I find that I make fairly substantial mistakes because I confuse the details of one task with the details of another".

4.2.3 Indecisive. Among respondents who answered "Yes" to our question about considering task switchings as disruptive as task interruptions, 14 (10%) were indecisive when it came to question about their choice. 8 (6%) stated that the type and the granularity of the on-going and new tasks impact their answer to this question: "Depends on the type of task. If the task is totally new then it bothers otherwise (like a switch to a bug fix issue) it is not a big deal." Moreover, 7 (5%) respondents indicated that the source of the task switching (i.e. initiated by themselves, or by external events such as a production issue or helping someone track down a bug) impacts the disruptiveness of the switch: "If I'm been asked to switch tasks, it is an interruption. If I have made the decision myself for reasons such as taking a break from a task or wanting to tackle an easier/harder task first, I don't consider it an interruption".

4.3 RQ3- Daily Stand-up Meetings

To explore developers' perceptions of the disruptiveness of daily meetings, we asked them about the impact of switching to these meetings on their primary task's performance after the resumption process. Moreover, we asked them to identify a time when they feel the meetings are less disruptive to their development tasks.

4.3.1 Disruptiveness of Daily Meetings. We asked respondents to complete the following sentence stem: "If I switch my current ongoing task to a daily team meeting, this interruption has a . . . impact on my primary task's performance after resuming this task". 64 (45%) stated that switching to daily meetings has a negative or very negative impact on their performance after resuming the primary task. These respondents believed that discussions in these meetings are more general and unrelated to specific tasks which make daily meetings not informative enough for most people: "It's an interruption like any others unless I needed help and got it in the meeting. Time after meetings is usually my least productive". Moreover, based on the results of our coding process, 5 respondents stated that not only do these meetings take up their time, they stop them from engaging in meaningful work schedule before and after: "Say there is a meeting at 11, and 1 pm and you usually eat lunch at 12. You simply will not be able to start a task that needs 90 minutes of focus from 10am-2pm. And rarely can you estimate that. I tend to think of "real work" in 2-hour blocks and anything less will just be "busy work" like responding to emails and helping others". This is consistent with what Stray et al. [16] and Solingen et al. [10] found: developers found the time of daily meetings disruptive (as it results in an undesirable long resumption lag).

On the other hand, 32 (23%) respondents acknowledged that daily meetings have a (very) positive impact on their performance after resuming the primary task. Getting context on where the rest of the team is at, getting help to unblock each other, and clarifying tasks and their priority are the main reasons provided by these

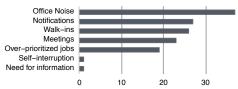


Figure 5: Coding frequency for open-ended question about the most disruptive interruption

respondents: "while they might negatively impact my task performance, the utility of stand-up outweighs that. Regular meetings keep everyone on the same page, so what may take 2 min in the stand-up could take 15+ in a chat". Moreover, 45 (31%) participants believed that daily meetings are something planned and expected, and have no impact on their performance: "If it's a regular meeting and it's a short one, I don't expect an impact. My brain knows it's coming".

4.3.2 A Productive Meeting Time. To further investigate the association between the time and the disruptive impact of daily meetings, and as requested by our industrial partner after reporting the results of our second pilot study, we asked respondents about temporal aspects of the daily meetings in their company. When asked "Is there any specific (pre-defined) time slot for meetings in your company?", 99 (70%) responded "No", 42 (30%) answered "Yes" [30 (21%): "Yes, usually before noon"; 12 (9%): "Yes, usually afternoon"]. Moreover, to check the correlation between participants' response to this question and to the question about the disruptiveness of switching to daily meetings, we used the Spearman's *correlation rank test* and defined $|\rho| \ge 0.50$ as a strong correlation coefficient. The results of this test show that there is a week correlation between the perceived disruptiveness of daily meetings ranked by participants and the schedule of meetings in the company they work for (rho=0.2, *p-value*= 0.04).

Moreover, we asked participants which time for scheduling daily meetings has the least negative impact on their performance among "morning, before starting daily tasks", "anytime in the morning", "right after noon", and "afternoon, as the last daily task". The majority of respondents (100, 71%) perceived "morning meetings" less disruptive to their daily tasks, among which 62 (44%) respondents stated that it might be better to get the meeting done before starting any tasks than to interrupt them to have a meeting that may or may not be productive: "I tend to do my best work in the late morning and early afternoon. Since our team stand-up is in the morning, it's helpful to know what I need to do and then have a long uninterrupted period of time to do it in". Scheduling the meetings for "right after noon" comes next with 22 (15%) responses. These respondents believe that there already is a disruption at noon (eating lunch), thus, by scheduling the meetings around noon they can have a nice contiguous block of disruptions rather than have the meetings all spaced out: "I try to batch my meetings together to make long spans of uninterrupted time. Lunchtime is the best time to synchronize with a common interruption; morning is harder because not everyone gets in at the same time". Scheduling the meetings as the last daily task comes last with 20 (15%) of responses. Seven respondents in this category believed that meetings at the later time of a day give developers more time to prepare and finish final touches on their work before they have to discuss them. However, the rest of respondents perceived this time help decrease the disruptiveness of daily meetings: "Meetings are mind-numbing and should be done

when no further thinking is required afterward. Mornings tend to be more productive for me to get focused work done".

5 PRACTITIONERS' CORNER

5.1 Lessons Learned

Following important lessons are aggregated:

- Regardless of the interruption characteristics and type, as stated by our participants, flow change and losing focus are the main reasons which make interruptions costly and harmful to the overall performance of the primary task. This suggests that methods for securing and rebuilding developers' focus can provide value.
- The majority (81%) of respondents of our study perceive that "task switching" is as disruptive as "interruption" because of the *cognitive cost* (e.g. low recall performance, memory load) and *low performance* (e.g. timeliness and the negative impact of task switching on productivity, completeness, and accuracy) caused by task switchings.
- Respondents who perceived that switching to daily meetings negatively impact their performance on the primary task are working in a company without any pre-defined schedule for daily meetings. Losing the flow of the on-going task to attend a meeting appears to be a reason contributing to daily meetings being perceived as disruptive: "Meetings are the #1 productivity killer, especially the daily stand-ups. These meetings often sneak up on me since I'm so engaged in my current task, which means that I most likely did not stop at a good place (i.e. I stop in the middle of a subtask)".
- The timing and content of daily meetings are perceived to be influential on the disruptiveness of these meetings. Frequent meetings and large meetings that have more to do with process and organizational changes tend to be more disruptive than planned and short meetings. The results of our analysis suggest that it might be better to get the meeting done before starting any tasks than to interrupt them to have a meeting that may or may not be productive.

5.2 Recommendations

Sources of Interruptions: When we asked respondents about the most disruptive sources of interruptions in their working environment, as illustrated in Figure 5, the majority of respondents stated that "office noise" such as overhearing co-workers' phone calls in an open office, or the random chatter of colleagues on the other side of the office is the most disruptive source of distraction to their work: "Just wish we could find a way to manage "focus time" more explicitly. We work in an open office so that everyone knows about everyone's progress, but this also means that someone is sitting facing me. When she looks up it feels like looking at me, which is kind of awkward ... too bad we don't have an "I am focusing" signal". This is in conformance with Glass's finding: "The working environment has a profound impact on productivity and product quality" [9]. The other most commonly occurring codes on respondents' answers to this question, as illustrated in Figure 5, are notifications, walk-ins, meetings, over-prioritized jobs, self-interruption, and need for information.

Notifications: To manage the disruptiveness of notifications, we propose that it may be less disruptive in some situations (e.g.

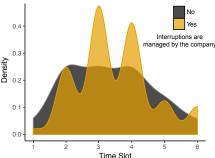


Figure 6: Answers to question: "On a typical working day, how often you switch your software development tasks?" [1]: Less than every 10 minutes. [2]: Every 10-30 minutes, [3]: Every 30-60 minutes, [4]: Every 1-2 hours, [5]: Every 2-4 hours, [6]: Others.

where a developer just initiated a task) to delay receiving/addressing notifications, as stated by one of our participants: "when I join a team, I turn off all notifications, and then selectively turn notifications back on as I find they reflect information I need in a more timely fashion. While this does degrade my performance a bit, I do this for cases where I decide that degradation is worth the information— and I also block out all notifications when I'm super focused".

Frequency of Task Switching: To consider environmental factors such as the working environment and communication methods, before asking about the frequency of task switchings, we first asked participants whether their company use any technique to protect them against task switching and interruptions. 75 (53%) responded "Yes". Having frequent Scrum meetings to focus on only planned items, using noise-canceling headphones, keeping the scope of the meetings minimal, defining library hours during the week, nomeeting Mondays, and improving build speeds are some example techniques mentioned by our respondents. Then, respondents were asked to define the frequency of their task switching. The density plots are shown in Figure 6 summarize and compare the distribution of answers to this question. Comparing the two density plots, it is clear that the shape of distribution of answers from developers who work in a company with no clearly defined strategies for managing interruptions is more homogeneous (Skew_{no}= 0.26, Skew_{ues}=0.43) and light-tailed (Kurtosis_{no}=-0.7, Kurtosis_{ues}=-0.2) across the range of possible scores compared to the "Yes" category, i.e. the concentration of values associated with the "Yes" category is higher towards the "every 30-60 minutes" and "every 1-2 hours" responses. One possible explanation for this could be that respondents in this category are better protected from random and spontaneous interruptions by their company. Considering the unproductive reimmersion time associated with each task switching or interruption, frequent task switching during a day can use up most of that day.

Management: As stated by our participants, paranoia interruptions or anything with some anxiety attached to it would be harder to overcome. For example, if a colleague comes in with an invitation to share a treat, it is way less interruptive that someone calling for a harsh issue, as in: "we had a manager who stack ranked primarily on the number of code reviews (CR) performed. Everyone was on a hair trigger for when CR requests came through, they would drop what they were doing and fight to be in the first four reviewers (the only ones that counted)". People have different degrees of tolerance for interruptions, urgent new tasks or intermittent interruptions can be detrimental to developers' productivity. Our results suggest

setting expectations and boundaries for your team and yourself. The more you can reduce or plan around potential interruptions, the better.

CONCLUSION 6

To investigate how developers perceive and distinguish the concepts of "task switching" and "task interruption", we conducted a survey with 141 professional software developers from across the world. The survey results show that while most participants considered task interruptions and task switchings disruptive (due to the flow change, losing focus, context switching, etc), 79 (56%) respondents reported that, in a typical day, they cannot be fully focused and frequently flip back and forth between different tasks (45 (32%): less than every 30 minutes, 34 (24%): every 30-60 minutes, RQ1). The results also show that practitioners perceive task switchings are as disruptive as spontaneous and random interruptions, regardless of the source and type of the switching (RQ2). The heavy cognitive load and the low performance caused by frequent task switching are the main reasons for this perception. In addition, we explored practitioners' perceptions of the disruptiveness of daily meetings as well as their perceptions of a productive meeting time. While some participants considered daily meetings useful in terms of clarifying daily tasks and gaining valuable input on how they should proceed, the majority of respondents perceived the impact of daily meetings to be (very) negative (RQ3). The findings of this study are in line with our previous studies [1-4] on the disruptiveness of task switching and task interruptions.

REFERENCES

- [1] Zahra Shakeri Hossein Abad, Oliver Karras, Kurt Schneider, Ken Barker, and Mike Bauer. 2018. Task Interruption in Software Development Projects: What Makes some Interruptions More Disruptive than Others?. In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering (EASE'18). ACM.
- Zahra Shakeri Hossein Abad, Guenther Ruhe, and Mike Bauer. 2017. Task Interruptions in Requirements Engineering: Reality versus Perceptions!. In Requirements Engineering Conference (RE), 2017 IEEE 25th International. IEEE, 6–15
- Zahra Shakeri Hossein Abad, Guenther Ruhe, and Mike Bauer, 2017. Understanding Task Interruptions in Service Oriented Software Development Projects: An Exploratory Study. In Proceedings of the 4th International Workshop on Software Engineering Research and Industrial Practice (SER&IP '17). IEEE Press, 34–40.
- Zahra Shakeri Hossein Abad, Alex Shymka, Jenny Le, Noor Hammad, and Guenther Ruhe. 2017. A Visual Narrative Path from Switching to Resuming a Requirements Engineering Task. In Requirements Engineering Conference (RE), 2017 IEEE 25th International. IEEE, 442-447
- Steve Adolph, Wendy Hall, and Philippe Kruchten. 2011. Using grounded theory to study the experience of software development. Empirical Software Engineering 16, 4 (2011), 487–513.
- Erik M Altmann and I.Gregory Trafton, 2002, Memory for goals: an activation-based model Cognitive Science 26, 1 (2002), 39 - 83
- Carey D Chisholm et al. 2000. Academic Emergency Medicine 7, 11 (2000), 1239-1243.
- Jan Chong and Rosanne Siino. 2006. Interruptions on Software Teams: A Comparison of Paired and Solo Programmers. In Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work. ACM, 29–38.
- Tom DeMarco and Tim Lister. 2013. Peopleware: Productive Projects and Teams. Addison-
- Van Solingen et al. 1998. Interrupts: Just a Minute Never Is. IEEE software 15, 5 (1998), 97-103. Graham R Gibbs. 2002. Qualitative data analysis: Explorations with NVivo. Open University.
- Barbara Kitchenham et al. 2017. Robust Statistical Methods for Empirical Software Engineer ing", journal="Empirical Software Engineering, 22, 2 (2017), 579-630,
- Chris Parnin and Spencer Rugaber. 2011. Resumption Strategies for Interrupted Programming Tasks. Software Quality Journal 19, 1 (2011), 5–34.

 Dario D Salvucci and Niels A Taatgen. 2010. The multitasking mind. Oxford University Press.
- David W Scott. 1979. On Optimal and Data-based Histograms. Biometrika 66, 3 (1979), 605-
- [16] Viktoria Stray, Dag IK Sjøberg, and Tore Dybå. 2016. The Daily Stand-up Meeting: A Grounded Theory Study. Journal of Systems and Software 114 (2016), 101 – 124.
- [17] Colin A. Terry, Punya Mishra, and Cary J. Roseth. 2016. Preference for multitasking, technological dependency, student metacognition, & pervasive technology use: An experimental intervention. Computers in Human Behavior 65 (2016), 241 - 251.
- Bogdan et al. Vasilescu. 2016. The Sky is Not the Limit: Multitasking Across GitHub Projects. In Proceedings of the 38th International Conference on Software Engineering. 994–1005