

Knock Brush! Perceived Impact of Push-based Notifications on Software Developers at Home and at the Office

Vanessa Vella^a and Chris Porter^b

Department of Computer Information Systems, Faculty of ICT, University of Malta, Malta

Keywords: Digital Interruptions, Notifications, Software Developers, Human Factors.

Abstract: Responding to a digital interruption requires software developers to transfer their attention from their ongoing task to the contents of the notification: a shift which will disrupt their original flow of work. This paper distinguishes between different types of notifications (intrusions and interruptions) and reflects upon the results of a survey with 88 respondents. This study contributes to research by providing insights consistent with literature, particularly related to how users react to push-based notifications during varying contexts, how the software developers perceive the cost and benefits of a notification, what features stand out in a notification and the strategies used to continue their work following a notification.

1 INTRODUCTION


Push-based notifications can be distinguished by their content: how relevant is the message right now; timing: how convenient is it that this message is delivered at 10:00; and trigger, referring to the impact this notification will have on the user, the options the user can choose from, to interact with this notification, and the likelihood that a user will ignore the notification (Morrison et al., 2017).


The advent of digital disruptions has the negative connotation of disrupting the user's flow state: the mental state where the software developer is concentrating so deeply on the ongoing task that they fail to notice the environment around them or how long have they been working on this task (Gray and Rumpel, 2017). Needless to say, software developers remain fully aware and focused on their primary task at hand (Okoshi et al., 2017). Nevertheless, the greater issue of digital disruptions is the ability to find a balance between signal and noise, or in other words having the ability to granularly define what an important and actionable notification consists of, while disregarding the less important notifications. Having achieved this state of balance will in turn help the software developer lengthen their attention span thus improving their ability in performing long-term work (Okoshi et al., 2017).

Whilst researchers agree that working in an overwhelming state of noisy interruptions comes at a detriment to productivity, literature has also shown that knowledge workers tie a quicker response time in handling notifications to better interpersonal skills (Fitz et al., 2019). This hypothesises that the ability to handle notifications will in turn be seen as having better task handling by management. For this reason, knowledge workers are more inclined to be aware of any notifications that they may receive to further contribute towards their social accomplishments (Okoshi et al., 2017).

With regard to the study methodology, a quantitative empirical study was implemented, where a survey was drawn with the purpose of acquiring data related to the perceived impact of a desktop digital interruption on a software developer. A mix of open-ended and likert-scale questions were used with the aim to best capture the day-to-day life of a knowledge worker and how the software developer is impacted by a notification, particularly delving into both the direct and mediated effects, if any.

The paper is split up as follows: the next section is an overview of related literature and the definitions adopted in this study. Section 3 details the implementation of this study, while section 4 contains a thorough discussion of the results of this study. Section 5 discusses the study limitations and possible directions for future research. A conclusion section (section 6) presents a synthesis of the main findings.

^a  <https://orcid.org/0000-0003-0654-7237>

^b  <https://orcid.org/0000-0003-3813-2941>

2 BACKGROUND AND LITERATURE REVIEW

2.1 Digital Interruptions

Literature has shown that there are different types of digital interruptions, which can be classified as follows (Addas and Pinsonneault, 2015):

- IT Intrusions: interruptions which are of a perceived, induced or mediated nature; for instance any external event which is irrelevant to the ongoing task. This is further structured as follows:
 - Informational Intrusion: a digital interruption having content related to a secondary task,
 - Actionable Intrusion: a digital interruption which causes the user to stop working on their task and start working on a secondary task,
 - System Intrusion: a digital interruption which indicates an error in the system or the lack of availability of system resources; this will thus disrupt the user's flow of work.
- IT Interventions: any interruption which disrupts a knowledge worker's ongoing task since it reveals a discrepancy between what is expected and what the user will deliver. IT Interventions can be further split as follows:
 - Informational Intervention: a digital intervention which contains data that is directly relevant to the user's primary task,
 - Actionable Intervention: a digital interruption that instigates two-way communication and is directly related to the ongoing task: most often actionable interventions reduce the discrepancy between what the user needs to finish this task and how to finish this task.

Hence, a user can respond to a notification in four possible ways (Mehrotra et al., 2016):

- Handle it immediately,
- Acknowledge it, but keep working on their task, and handle it later,
- Decline it, that is, explicitly close the notification,
- Withdraw it, that is, implicitly refuse to handle it.

2.2 Task Analysis

In Human-Computer Interaction (HCI), task analysis can be defined as *"creating a sufficiently rich and abstract representation of a task situation as it exists or*

might exist to allow reasoning about the task situation without having to refer to the actual task situation itself." (Haan, 1998) This definition directly encompasses the fact that interaction in HCI often times occurs at a high-level of abstraction.

Consequently, multi-task analysis can be defined in terms of the primary and secondary tasks it is made up of; a primary task is the task which takes priority, meaning the task which requires the larger amount of mental and physical resources (de Haan et al., 1991). Meanwhile, a secondary task is a smaller task which is related to the primary task's scope, and completion of this task, directly supports completion of the primary task (de Haan et al., 1991).

2.3 Cognitive Load in Software

Cognitive load is a measure of the amount of effort being used by the working memory — working memory is the part of the cognitive system that processes the information regarding the user's primary task (Kalyuga and Singh, 2016). Working memory is prone to an overload, meaning that it is susceptible to discarding information if it is no longer being processed. Software development tasks such as programming and code debugging are dependent on the working memory. Often times, the software developer will have to abstract an algorithm and think of the next steps, whilst already making plans to refactor the code that they have developed earlier. This process is referred to as dividing and conquering by Petre et al. (2016) who believe that software developers essentially address smaller problems independently, whilst still reflecting on the associations between different code blocks. Consequently, developers are able to make adjustments in different code points in scope of the larger task. Petre et al. (2016) then proceed to notice that high-performing teams often collaborate in order to identify, as early as possible, any divergences in specifications that might occur if one team makes changes whilst the other does not.

Maintainable code is code that follows practices, conforms to established design patterns and is readable (Code, 2018). Software developers are accustomed to reading code and building mental models depending on what they are developing. The mental models are in turn converted to code representation when the developer manages to implement the changes. This process is heavily dependent on the software developer's abstraction skills, that is, code reading is a learning process. Therefore, if the code is understandable, the knowledge worker is said to make use of less extrinsic load, and consequently lessen cognitive load (Kalyuga and Singh, 2016). By defi-

tion, extrinsic load is one of three subparts of cognitive load and refers to the unnecessary work that the learning process is making use of — which could have easily been allocated to the learning process itself. The other subparts are (a) intrinsic load, which refers to the complexity of the learning process itself and the user's ability in gaining new knowledge, and (b) germane load, which is the effort required to store sub-parts of the learning process in long term memory (Kalyuga and Singh, 2016).

2.4 Current Implementation of Notifications in Operating Systems

All three major operating systems (OSs) have a similar implementation for push-notifications, meaning, allowing installed applications to interrupt the user with a message when an event occurs, new data is available, or the status of an object has changed. Nevertheless, there are slight differences in the implementation and configuration between the OSs. For instance, notifications on macOS (Apple, 2020) are shown at the top-right corner of the screen, and listed in the notification centre. An actionable notification in macOS is called an alert, while an informational one is called a banner (Apple, 2020). Notifications in Windows (2020) are shown at the bottom-right corner of the screen, and listed in the action center; these can be either actionable or informational, depending on the application which pushed it. Notifications in Linux, particularly GUI based implementations such as Ubuntu, also show informational notifications at the top-right, but so far have not implemented actionable notifications (Ubuntu, 2020). A recent notification list is also available in Ubuntu (2020).

One important feature related to notifications which is implemented in all three mentioned OSs is the *Do Not Disturb* feature – as labelled in macOS (Apple, 2020) and Linux (Ubuntu, 2020), while labelled *Focus Assist* in Windows (2020).

3 SCOPE AND METHODOLOGY

The general objective of this research is to understand how desktop notifications affect knowledge workers during a state of flow. Consequently, this study focuses on the daily use of a desktop system from the GUI (Graphical User Interface). Thomas Davenport (2005) defines knowledge workers as individuals who "have high degrees of expertise, education, or experience, and the primary purpose of their jobs involves the creation, distribution, or application of knowledge." Davenport (2005) continues to argue that by

considering software developers as knowledge workers, implicitly development activities are thereby regarded as knowledge creation processes. Based on the above, and for the scope of this research, we limit our definition of knowledge workers to software developers.

For this reason, this study aimed at building a better first-hand understanding of the different notification taxonomies. Following a critical review of related literature while also familiarising better with the domain, a survey was drawn in order to acquire the perceptions and experiences related to digital interruptions. The survey questions were designed to encourage, as much as possible, software developers to open up on their experiences of interruptions during immersive tasks and how they dealt with picking up from where they left off. For this reason, qualitative questions were used, at times supplemented with quantitative probes on specific aspects as informed from existing literature.

In terms of resources, this survey made use of a freely available platform: Google Forms. The survey administered the informed consent and a brief overview of the scope of the study. No personal identifiable data was collected, however a number of questions were included related to experience in the ICT sector. In terms of distribution mechanisms, the survey was sent as an online link to a number of ICT companies, as well as shared on social media, including Facebook and LinkedIn.

4 FINDINGS

In this section the survey results will be collated, discussed and compared with the relevant literature.

4.1 General Observations

88 respondents took the time to answer the survey, two of which contain spam data and will be excluded from the results, therefore bringing the total number of participants down to 86.

The participants come from varying ICT backgrounds with 14 participants (16.3%) having less than a year of experience in the field. Conversely, 21 participants (24.4%) have more than 7 years of experience in the ICT sector. The rest of the participants were in the 1-3 (25.6%) and 4-6 (33.7%) years-of-experience brackets.

The majority of the participants (81%) specified that they used Windows as one of their development platforms, if not the only one, followed by Linux (29.8%) and macOS (14.3%).

The most common examples of desktop notifications received by the software developers were new email notifications (86%) and instant message chats (83.7%) from communication channels. Other examples of push-notifications include system updates (75.6%), task updates from productivity systems such as Jira (47.7%), and notifications unrelated to work, including phone calls, third-party weather sites, articles, and social media (38.4%). The platform which is most frequently used to communicate with colleagues is emails as a structured platform and Microsoft Teams for instant-messaging. Table 1 contains a breakdown of the mentioned platforms; note that the participants had the option to choose multiple services.

Table 1: Communication platforms mentioned by the participants, ordered by popularity descending.

Platform	% of Participants
Emails	69.77%
Microsoft Teams	65.12%
Slack	51.16%
Skype	40.70%
WhatsApp	40.70%
Zoom	25.58%
Google Meet	22.09%
Discord	17.44%
Facebook Messenger	17.44%
Google Hangouts	12.79%
Phone Calls	3.49%
Rocket Chat	2.33%
Lifesize	1.16%
LinkedIn	1.16%
Teamviewer	1.16%
Telegram	1.16%

4.2 Do Not Disturb Usage

The participants were asked whether they make use of the *Do Not Disturb* feature and more than a third of the software developers specified that they do occasionally use this feature (36%). 29 developers (34%) specified that they make use of the feature quite regularly, while 26 (30%) reported that they never make use of this feature. When probed further to specify the scenario where the feature is switched on, almost a quarter of the participants mentioned meetings and screen-sharing occasions (25.6%), and when the frequency of notifications suddenly increases due to, for instance, a spike in group chat activity. Participants also mentioned a number of scenarios where this feature is used, including deploying code to production, troubleshooting and debugging code, as well as code comprehension and reviewing. All of these tasks can

be considered to be immersive in nature, which can have significant side effects if errors are introduced. This corroborates observations presented in (Baethge and Rigotti, 2013), which state that interruptions increase stress and cognitive load since these disrupt the software developer's state of flow and thus require the utilization of additional resources and effort to achieve the goal of the primary task.

4.3 Notification Characteristics

It is also important to note that the frequency of instant messaging notifications is rather high, with 57 participants (66.3%) stating that they receive more than one instant messaging (IM) notification daily, while 19 (22.1%) receive at least one IM notification every day. For this reason the participants were asked to specify in what ways do they determine whether a notification has a higher priority. The following is the list of features identified:

- **Who the Sender Is:** particularly if it's a team member, the client or contacts marked as VIP (i.e. flagged so that all communication from this contact passes through regardless of whether do-not-disturb is switched on),
- **The Application Generating the Notification:** participants observed that certain applications, such as Microsoft Teams, are considered to generate work-related notifications, while other platforms, such as Messenger, are considered to treat the social context,
- **Notification Target:** notifications which represent direct messages are perceived as more important than messages in a group chat, similarly when a developer is "tagged" or mentioned in the group chat, as opposed to a general message to all members,
- **The Nature of the Notification:** whether it is informational or actionable,
- **Colour Scheme:** red icons imply a higher priority,
- **Directly Related to an Ongoing Task or Deadline,**
- **The Blurb:** the information contained in the pop-up including the title and the topic, especially if it is a production issue,
- **Urgency Level:** The business priority or status attributed to the task update (in the case of a task update), or a flag in Microsoft Outlook emails.

Similarly, when asked how do they determine if a notification is of a lower priority, the software develop-

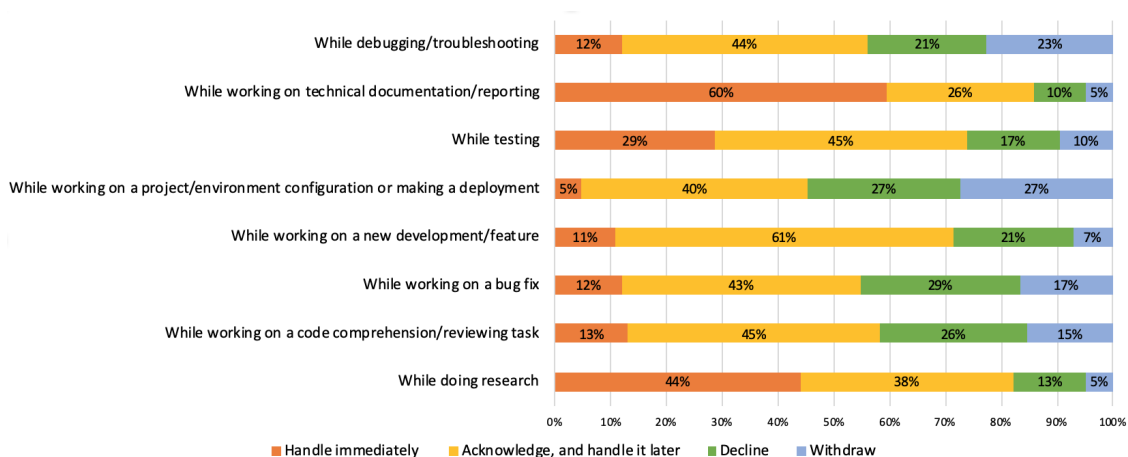


Figure 1: Stacked bar chart showing how software developers react to digital interruptions during different types of primary tasks.

ers noted that automated notifications, system messages and spam or promotional messages often go unnoticed or completely disregarded, with little to no disruption to their state of flow. The above features list ties in to how important these traits can be in assigning priority (Paul et al., 2015). Paul et al. (2015) argue that interruptive notifications can decrease a user’s attention by offering a distracting focal point. On the other hand, the authors (Paul et al., 2015) also specify that having a second focal point is not necessarily detrimental, in fact, notifications are known to support task handling and decrease user stress levels by providing more information on the system’s current status.

4.4 Impact on the Primary Task

The context surrounding the digital disruption is one of the larger determinant factors in the user’s decision as to the way they will tackle it and when. The participants were asked to distinguish and mark the ways in which they generally handle notifications in varying contexts. One method which seems to be frequently adopted is the *acknowledge and postpone* protocol, which corroborates findings in (Mehrotra et al., 2016). This shows that the majority of the participants take some extra measures and implement contingency strategies before moving on to handle the notification. A second interesting result to point out is that the *handle immediately* protocol (Mehrotra et al., 2016) is normally performed when the software developer is working on less critical tasks, including research and documentation. A graphical representation of these results can be found in figure 1. In a follow-up question, all participants agreed that they will read through notifications which are related to the ongoing task, such that they will be able to take

the necessary information into consideration. Conversely, digital interruptions which are unrelated to the primary task are checked for urgency and then dismissed if these are of lesser priority. One software developer also addressed the fact that when stress levels are high, a notification is able to alleviate unnecessary stress. This is in-line with findings in literature (Adidas and Pinsonneault, 2015) where notifications are known to be an aiding sign in identifying when a software developer might want to switch to another task.

Furthermore, Paul et al. (2015) also discussed that software developers require a *“no notification zone”* - meaning that they will occasionally need to work in a context that is sensitive to interruptions. This state is otherwise known in this study as being in a state of flow, since the user is working on an immersive task. The majority of the software developers noted that during coding, especially, when making good progress, the users find themselves in a state of flow. Interestingly, one software developer explained the feeling as follows: *“Implementing a set of requirements I understand clearly, using a programming language I am familiar with.”* – where the software developer is hinting that s/he is feeling prepared and unless a disruption occurs, the task will be wholly completed. Other immersive tasks reported by participants include initiating larger projects, designing data models and methodologies, data migration processes, production deployments, debugging and troubleshooting, repetitive tasks (*“small and mundane”*), prototyping new technologies, creating UI mock-ups and user-flow diagrams, creating specifications for a new task, code reviewing and refactoring, scripting, identification of optimal architectures, and working on complex algorithms such as thread synchronisation.

Literature shows that digital interruptions do fragment the execution flow of the primary task by expecting users to switch their attention to the secondary tasks (Sonnetag et al., 2018). This consequently has led to the need that software developers implement contingency strategies so that they will be able to continue from where they have left off. State saving of the primary task helps the knowledge worker resume the suspended tasks quicker (Sonnetag et al., 2018). While not all software developers implement contingency strategies (18.6%), since they are able to recall what the primary task was, the rest of the participants recalled implementing certain protocols when a context-switch is required. The following is the list of methodologies mentioned, while figure 2 graphically depicts the number of frequency of participants per approach mentioned:

- (A) Split their work into smaller projects that can be tackled as a whole, in anticipation of digital interruptions,
- (B) Adding short notes (or voice notes) in Microsoft OneNote, or in the Task Management tool, or physically on a paper or sticky note,
- (C) Code versioning and including meaningful and descriptive commit messages,
- (D) Adding comments and TODO comments in the code,
- (E) Leaving the IDE open and include some highlighting to know where to pick up again, or pin code files,
- (F) Before starting to code, the software developer writes a pseudo-code version of their implementation, so they can consult it along the way or after an interruption,
- (G) Use multiple desktops, so that the state of one desktop is not affected,
- (H) Purposefully include an error in the code, for example write gibberish code so that the software developer knows on which line of code they were working on recently,
- (I) Organise browser tabs based on which task takes priority, and pin the ones which need to be attended to later.

Figure 3 shows that the participants specified that related instance messages and related emails are less distracting as opposed to software updates and unrelated emails. Nevertheless, conflicting results were found in literature in this regard; Paul et al. (2015) stated notifications related to the software developers' primary tasks provide little to no benefit and can be disruptive. Nevertheless, the results obtained in this

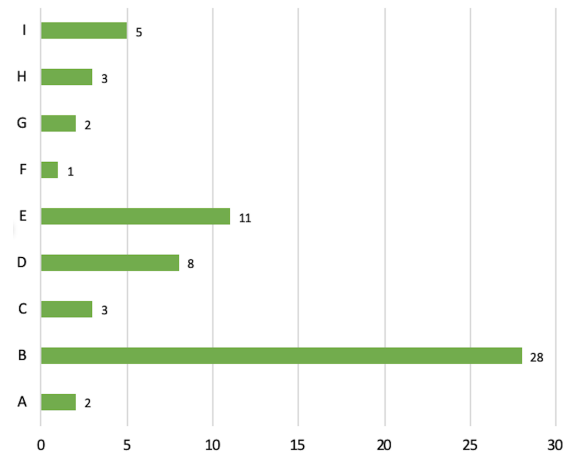


Figure 2: Frequency chart showing the methodologies adopted by the participants when a context-switch is required.

study corroborate Mark et al.'s (2005) work, where the researchers focused on fragmented workflows and found out that interruptions outside the users' work sphere are more disruptive than those related to software developers' primary task.

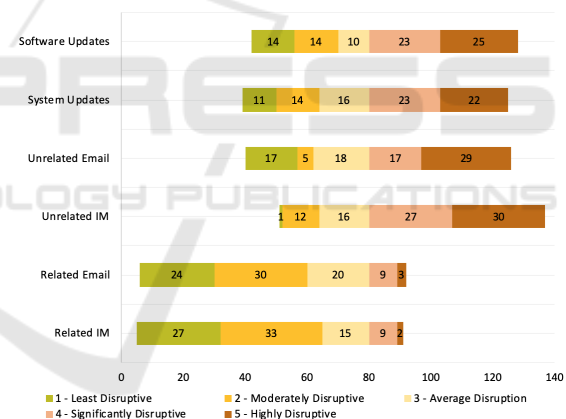


Figure 3: Likert plot showing how software developers perceive types of notifications, with 5 being the most disruptive and 1 being the least.

4.5 Impact on Users' Emotional State

This study showed that 66 participants (76.7%) in fact check that all the necessary communication channels are open and working before they start their workday. 14 software developers (16.3%) noted that this check is occasionally performed, while only 6 (7%) denied checking. This result suggests that the software developers do in fact anticipate having noisy days, and ensure that all the relevant notifications do pass-through. As the number of push-notifications increases, literature has shown that software developers have the increased tendency to anticipate interrup-

tions, thus working harder during notification downtime to compensate for when the cost of distraction is too high (Petre et al., 2016).

In the same context, the software developers were asked to specify whether they experience a fear-of-missing-out (FOMO), which could arise from a reduced notification-frequency when compared to a typical workday. Interestingly, only 17 participants (19.77%) admitted that they feel fearful in such a case. Similarly, 19 participants (22.09%) specified that they do occasionally have feelings of FOMO. Nevertheless, the majority (56.8%) denied having FOMO feelings. The portion of the cases who admitted to having FOMO, or occasionally, were then asked to reflect on the incident. One software developer admitted that since the notification handling strategy implemented is not perfect, the Chief Technical Officer (CTO) felt that extra attention was given to ensure that any high priority support task notifications were monitored for and handled on-time.

Furthermore, a few participants (4.7%) noted that FOMO is also dependent on the working environment – the participants argued that working in an office helps in this regard since you can get the occasional “tap-on-the-shoulder” from a colleague. Meanwhile, the participants were asked whether they considered push-notifications more crucial when working from home, since they are one of the few ways to shift the software developer’s focus on higher prioritised tasks. The majority of the participants (61.6%) do feel that digital interruptions are critical when working from home. 16 software developers (18.6%) expressed that they do occasionally view notifications as important in this context, while 17 (19.8%) dismissed the statement altogether. The question that ensued was to understand what strategies do the software developers implement to ensure that they will receive all the relevant notifications when working from home. The users specified that they attempt to ensure having fast and stable internet and VPN connections. Other developers noted that they implement the same strategy as when working from the office, with dedicating part of the screen to communication channels, occasionally checking the notification tray, and switching off the do-not-disturb (DND) functionality. One software developer also noted that raising the system volume would help in identifying incoming notifications, even if they are missed visually (e.g. produced within a browser tab). Sound is inherently one of the multimodal cues that are often times included alongside the pop-up and the badge number on the tray icon, amongst others (Addas and Pinsonneault, 2015).

When asked to list any differences, if any, with regard to non-work related digital interruptions in the

context of working from home, a few software developers (7%) pondered on the fact that since social interaction becomes more limited, the software developer receiving a non-work related notification is more likely to handle it, than ignoring or dismissing it. This research is inline to a recent study (Ohme et al., 2020) that has shown that socially-distanced users are making more frequent and longer use of news platforms (54% increase), messaging platforms (64% increase) and social media applications (72% increase).

5 LIMITATIONS AND FUTURE WORK

Notifications are an inevitable part of a knowledge worker’s workday. As discussed in this study, this issue needs to take into consideration a number of different perspectives, including the software developer’s working context, whether the notification contains important information, and the user’s primary task, amongst others. Given that this study collected only survey data, there might be the case that the participants did not feel encouraged to provide factual and unbiased answers (Vaske, 2011). While the questions included were designed to motivate the users to share their opinions on the subject-matter, not all participants may have felt comfortable with sharing, with some opting to write short answers only.

A second study limitation may be related to the limited contextual-specificity of the questions. Since the survey questions were designed to be generic enough to accommodate the various roles within the software development community, it can be the case that not all participants related well to the same questions. The goal of this study was to produce overall insights into the impact of notifications as perceived by the software development community, however, further studies can be devised to investigate particular roles and related contexts.

Stated impact may differ from actual behaviour — more specifically, while users do understand having the option to tweak notification settings, nevertheless, very little is done to maximise their state of flow. Therefore further studies can dig deeper into software developers’ actual behaviour when presented with notifications in different contexts. Possible research questions may look into the impact of different notification types on performance while carrying out tasks of varying levels of difficulty. Studies could look at how, for instance, actionable intrusions and informational interventions affect performance across different groups of knowledge workers based on criteria such as experience.

6 CONCLUSION

Digital interruptions at the workplace pose a number of threats to the quality of the work done, stress levels and cognitive mental workload. This study contributes to research by providing insights consistent with the cognitive factors and human ergonomics community literature, in relation to how software developers perceive digital interruptions and the traits of when notifications occasionally aid the user, while sometimes they disrupt.

This research has shown that notifications related to an ongoing task are perceived as less disruptive, and in fact acknowledged and handled quicker. The inverse also holds, whereby, notifications which are unrelated to the primary task will be viewed as disruptive, consequently having a larger effect on the software developer's state of flow. For this reason, contingency strategies, such as note-taking and code versioning are crucial towards helping the software developers continue their primary task.

In conclusion, notifications definitely hold a significant role in updating knowledge workers on their information services.

ACKNOWLEDGEMENTS

This research is partially funded by the Endeavour Scholarship Scheme (Malta), which is part-financed by the European Union - European Social Fund (ESF).

REFERENCES

- Addas, S. and Pinsonneault, A. (2015). The many faces of information technology interruptions: a taxonomy and preliminary investigation of their performance effects. *Information Systems Journal*, 25(3):231–273.
- Apple (2020). Human interface guidelines - notifications.
- Baethge, A. and Rigotti, T. (2013). Interruptions to work-flow: Their relationship with irritation and satisfaction with performance, and the mediating roles of time pressure and mental demands. *Work & Stress*, 27(1):43–63.
- Code, W. M. E. (2018). Abstraction layered architecture: Writing maintainable embedded code. In *Software Architecture: 12th European Conference on Software Architecture, ECSA 2018, Madrid, Spain, September 24–28, 2018, Proceedings*, volume 11048, page 131. Springer.
- Davenport, T. H. (2005). *Thinking for a living: how to get better performances and results from knowledge workers*. Harvard Business Press.
- de Haan, G., van der Veer, G. C., and van Vliet, J. C. (1991). Formal modelling techniques in human-computer interaction. *Acta Psychologica*, 78(1-3):27–67.
- Fitz, N., Kushlev, K., Jagannathan, R., Lewis, T., Paliwal, D., and Ariely, D. (2019). Batching smartphone notifications can improve well-being. *Computers in Human Behavior*, 101:84–94.
- Gray, J. and Rumpe, B. (2017). The importance of flow in software development.
- Haan, G. d. (1998). Chapter 5: Task Analysis for User Interface Design.
- Kalyuga, S. and Singh, A.-M. (2016). Rethinking the boundaries of cognitive load theory in complex learning. *Educational Psychology Review*, 28(4):831–852.
- Mark, G., Gonzalez, V. M., and Harris, J. (2005). No task left behind? examining the nature of fragmented work. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–330.
- Mehrotra, A., Pejovic, V., Vermeulen, J., Hendley, R., and Musolesi, M. (2016). My phone and me: understanding people's receptivity to mobile notifications. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 1021–1032.
- Morrison, L. G., Hargood, C., Pejovic, V., Geraghty, A. W., Lloyd, S., Goodman, N., Michaelides, D. T., Weston, A., Musolesi, M., Weal, M. J., et al. (2017). The effect of timing and frequency of push notifications on usage of a smartphone-based stress management intervention: an exploratory trial. *PLoS one*, 12(1):e0169162.
- Ohme, J., Vanden Abeele, M. M., Van Gaeveren, K., Durnez, W., and De Marez, L. (2020). Staying informed and bridging “social distance”: Smartphone news use and mobile messaging behaviors of Flemish adults during the first weeks of the covid-19 pandemic. *Socius*, 6:2378023120950190.
- Okoshi, T., Tsubouchi, K., Taji, M., Ichikawa, T., and Tokuda, H. (2017). Attention and engagement-awareness in the wild: A large-scale study with adaptive notifications. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 100–110. IEEE.
- Paul, C. L., Komlodi, A., and Lutters, W. (2015). Interruptive notifications in support of task management. *International Journal of Human-Computer Studies*, 79:20–34.
- Petre, M., van der Hoek, A., and Quach, Y. (2016). *Software Design Decoded: 66 Ways Experts Think*. MIT Press.
- Sonnentag, S., Reinecke, L., Mata, J., and Vorderer, P. (2018). Feeling interrupted—being responsive: How online messages relate to affect at work. *Journal of Organizational Behavior*, 39(3):369–383.
- Ubuntu (2020). Ubuntu documentation.
- Vaske, J. J. (2011). Advantages and disadvantages of internet surveys: Introduction to the special issue. *Human Dimensions of Wildlife*, 16(3):149–153.
- Windows (2020). Take control of your notifications in the windows action center.