Breaking the Flow: A Study of Interruptions During Software Engineering Activities

Yimeng Ma
Duke University
Durham, USA
yimeng.ma@duke.edu

Yu Huang
Vanderbilt University
Nashville, USA
yu.huang@vanderbilt.edu

Kevin Leach
Vanderbilt University
Nashville, USA
kevin.leach@vanderbilt.edu

ABSTRACT

In software engineering, interruptions during tasks can have significant implications for productivity and well-being. While previous studies have investigated the effect of interruptions on productivity, to the best of our knowledge, no prior work has yet distinguished the effect of different types of interruptions on software engineering activities.

This study explores the impact of interruptions on software engineering tasks, analyzing in-person and on-screen interruptions with different levels of urgency and dominance. Participants completed code writing, code comprehension, and code review tasks while experiencing interruptions. We collect physiological data using the Empatica EmbracePlus wristband and self-perceived evaluations through surveys. Results show that on-screen interruptions with high dominance of requester significantly increase time spent on code comprehension. In-person and on-screen interruptions combined significantly affect the time spent on code review, with varied effects based on specific interruption combinations. Both interruption type and task significantly influence stress measures, with code comprehension and review tasks associated with lower stress measures compared to code writing. Interestingly, in-person interruptions present a positive impact on physiological measures, indicating reduced stress measures. However, participants' self-perceived stress scores do not align with physiological data, with higher stress reported during in-person interruptions despite lower physiological stress measures. These findings shed light on and emphasize the potential importance of considering the complex relationship between interruptions, objective measures, and subjective experiences in software development. We discuss insights that we hope can inform interruption management and implications on stress among software engineers.

(ChatGPT was used to revise and shorten paragraphs in this manuscript.)

1 INTRODUCTION

Interruptions have been studied in many fields such as social science, psychology, and cognitive science [14, 18, 38, 41, 56]. Many have measured interruptions according to their negative effect as a physical or emotional burden or difficulty. Interruptions lead to more stress, higher frustration, time pressure, and effort when performing fundamental cognitive tasks [8, 42]. In the context of software engineering, interruptions can have a significant impact on developers' perception of a good workday and their productivity [45, 46, 48], which affect their perceived happiness and work satisfaction [27–29]. This implies that interruptions can potentially trigger affective states in developers, which can subsequently influence their productivity, overall performance and satisfaction.

Software development involves common software engineering activities including writing code, comprehending code, and reviewing code [30]. Indeed, during a typical workday, approximately one-fourth of a software developer's time is dedicated to tasks like reading, editing, navigating code, and other code-related activities [46]. Amidst the significance of these code-related tasks, developers often encounter interruptions as they perform them. Logistic duties, such as attending to emails, planning, assisting co-workers, and holding meetings, can divert their attention from coding. Meyer et al. reported that the duration of uninterrupted coding time significantly influence developers' perceived quality of the workday [45]. Consequently, interruptions may have a subsequent impact on their productivity and job satisfaction. This implies different types of interruptions might have varying impact on programmers' productivity.

To better understand the impact of interruptions, we designed an IRB-approved human study focused on two common types of interruptions: *on-screen* and *in-person*. We design on-screen interruptions following commonly-encountered interruptions during workdays like email notifications [9, 19] and, more recently, advertisement pop-ups on websites like StackOverflow and Quora. We also design in-person interruptions to reflect a manager or colleague discussing work.

Notably, the COVID-19 pandemic has prompted a significant shift in work patterns, with 35.2% of the workforce working entirely from home in May 2020 [11]. This shift may lead to an increase in interruptions during work due to the added complexity of coordinating activities virtually [50]. Indeed, Leroy et al. found a large increase in interruptions since-COVID, with women reporting a greater increase in interruptions [40].

Understanding the effects of interruptions requires a multidisciplinary approach that considers various factors such as the type of interruption, individual differences in attention and response to interruptions, and the context of the software engineering activity. Some interruptions may be more disruptive than others depending on their source and urgency. For instance, a notification from a messaging app might be less disruptive than a phone call from a manager. Furthermore, the type of software engineering activity being performed can affect the impact of interruptions. For instance,

ICSE '24, April 14–20, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0217-4/24/04.
https://doi.org/10.1145/3597503.3639079

an interruption during code review might be less disruptive than an interruption when writing code. Individual factors such as attention span, cognitive load, and experience can also influence how interruptions affect software engineers. Some developers may be more easily distracted and have a harder time resuming their work after an interruption, while others may be better at multitasking and able to handle interruptions more efficiently. To fully understand the impact of interruptions on software engineers, we take a comprehensive approach that considers the various factors at play. In doing so, we hope to develop interventions and strategies to mitigate the negative effects of interruptions and promote more productive and healthy work environments for software engineers.

In this paper, we investigate the influences of interruptions in software engineering activities by analyzing the productivity and the quality of the work as well as relating them to physiological data and affective states. In particular, we are interested in the following three research questions:

RQ1: What is the effect of different interruptions on developers' performance across different software engineering tasks?
RQ2: What is the effect of different interruptions on stress measures across different software engineering tasks?
RQ3: Is there a correlation between objective measures of developers' physiological states and their self-reported feelings for different interruptions and tasks?

We perform a controlled human study with 20 participants. In this study, we invite participants to complete a pre-survey followed by a 2-hour in-person session, during which physiological measures were recorded through Empatica EmbracePlus wristband[1]. Participants complete three self-paced, indicative software engineering tasks—code writing, code comprehension, and code review—with the presence of interruptions.

In summary, we note the following findings from our study:

- On-screen interruptions with high dominance of requester notably increase the time spent on code comprehension problems. Moreover, the combined effect of in-person and on-screen interruptions significantly influences the time spent during the code review process.
- Interruptions during code comprehension and code review tasks are associated with higher physiological measures compared to the code writing task. In-person interruptions lead to an increase in physiological measures.
- Participants' self-perceived stress measures exhibit a contradiction with the objective physiological data.

All the study materials, code and de-identified data are available at https://doi.org/10.6084/m9.figshare.24944568.v2.

2 RELATED WORK

In this section, we discuss three lines of related work: (1) productivity in software engineering, (2) effect of interruption on developers' performance, and (3) objective physiological sensors in software engineering studies.

[1]https://www.empatica.com/embraceplus/

2.1 Productivity in Software Engineering

Productivity has been investigated in various software engineering contexts. Murphy-Hill et al. conducted a study across three different companies, using 48 questionnaire items as predictors to assess which factors better predict developers' self-rated productivity [48]. They found job enthusiasm, peer support for new ideas, useful feedback about job performance to be the top three factors. Although "I have few interruptions or distractions while working" appears as one of the predictors and is evident to cause disruptive effects, the concept of interruptions is very general and broad. The specific impact of interruptions on work can vary depending on the context and the individual, and Murphy-Hill et al. did not distinguish different types of interruptions. Furthermore, the study results are completely based on self-reported measures.

Beller et al. compared self-reported productivity and its attributes with time spent working on different applications by software engineers at Microsoft [10]. They reported a gap between perceived productivity and objective measures, and their final model was able to explain less than half of the variance contained in self-reported productivity when expressed as objective measures. Therefore, it is important to build comprehensive method from both objective and subjective productivity measures.

In this study, we encompass time-based metrics to evaluate objective productivity. We also assessed self-perceived productivity in the survey at the conclusion of each software engineering task.

2.2 Effect of Interruptions on Developers' Performance

Although interruptions are generally perceived as a factor that influences productivity, few studies have thoroughly investigated the impact of different sources of interruptions in software engineering. Abad et al. investigated the disruptiveness of task switching in software engineering and requirements engineering through five different studies [2–6]. They found no difference in the influence of interruptions of different duration no matter what types of tasks are being performed. They also performed a retrospective analysis and found that self interruptions (voluntary task switching) are more disruptive than external interruptions. While these investigations shed light on the general influence of self-interruptions and external interruptions, they did not consider the impact of interruptions on different software engineering tasks, nor do they consider urgency or power dynamics associated with interruptions.

Our study takes a novel approach by analyzing the effects of external interruptions, including in-person and on-screen interruptions, on three essential software engineering tasks: code writing, code comprehension, and code review. By exploring interruptions within the context of these specific activities, we aim to provide a deeper understanding of their influence on productivity and stress measures among developers. Moreover, we carefully distinguish external interruptions and consider their urgency or power dynamics, ensuring a comprehensive examination of their impact.

2.3 Objective Sensors in Software Engineering

Some studies in software engineering make use of objective physiological measures such as eye-tracking, functional magnetic resonance imaging (fMRI), and smart wristbands to access objective measurement of physiological responses. Huang et al. used fMRI and functional near-infrared spectroscopy (fNIRS) to understand mental processes associated with data structure manipulation [33]. Similarly, Krueger et al. used fMRI to find the dissimilarity in prose and code writing [37]. Later, Huang et al. also used eye-tracking technology and fMRI to find the differences in code reviews conducted by men and women [32].

Few studies used other objective physiological sensors. Müller et al. relied on a combination of eye tracking tehnology, an Empatica wristband, and an Electroencephalography (EEG) headband to classify developers' emotions and perceived progress during software development change tasks [47]. Girardi et al. used the Empatica E4 wristband to measure the electrodermal activity and heart activity to recognize developers' emotions during programming [25].

In our research, we build upon these previous studies by using the Empatica EmbracePlus wristband to obtain objective measurements of developers' affective states during software engineering tasks. By capturing physiological data such as Heart Rate Variability (HRV), we aim to gain a comprehensive understanding of how interruptions and task types influence developers' stress responses. Furthermore, we compare these objective measurements with participants' self-perceived assessments, providing a nuanced examination of the interplay between subjective experiences and physiological reactions in the software development context.

3 METHOD

In this section, we present a detailed description of the experimental design and procedures used to investigate the impact of interruptions on software engineering tasks. With the approval of the Institutional Review Board (IRB), we specifically recruited individuals from Vanderbilt University who are either majoring or minoring in Computer Science to complete tasks in C++. Furthermore, we conducted a pre-screening process to confirm that participants possessed the necessary basic knowledge of C++ to successfully complete all tasks in our study. Specifically, participants are required to have completed the university's data structures course in C++ or an equivalent course as a prerequisite for participating in the study. The data structures course corresponds to CS2 in the ACM Computing Curricula. Before the study, participants complete a pre-survey that asks them about basic demographic information, the Social Interaction Anxiety Scale (SIAS), and the Cognitive Failures Questionnaire (CFQ) (Section 3.2). Participants who complete the pre-survey come to a lab designed to replicate an office environment and complete an assessment on the lab computer consisting of indicative software development tasks (Section 3.3). As participants complete each task, we deploy (1) *in-person* interruptions in which a confederate goes into the lab to ask innocuous questions of the participant, and (2) *on-screen* interruptions such as notification pop-ups (Section 3.4). During the study, participants wear an Empatica EmbracePlus Wristband that collects their physiological data (Section 3.5). After completing each task, participants complete a post-task survey (Section 3.6). Participants do not know interruptions are designed and intentional until the debriefing session after the study. We discuss each step below in more detail.

3.1 Recruitment

We recruited participants by sending emails to departmental mailing lists and by giving 2-minute presentations at the start of various computer science classes with instructor permission. Potential participants enrolled by sending an email to the study coordinator, at which point they were given the pre-survey materials to complete. Participant identities were not stored in any research data, and was only used to track compensation of participants. Participants were compensated $40 for their time upon completing the study.

3.2 Pre-survey

Before participants come to the lab and complete the software engineering task, they complete a pre-survey electronically. Upon completion of the pre-survey, the participant is scheduled for a two-hour block for the in-person portion of the study. We collect several pieces of information about the background of the participant as described below.

Basic Information. The pre-survey gathers information related to basic demographic details (i.e., age, gender), English proficiency, and programming experience. We require participants to be fluent in English because the instructions and survey instruments are all in English. Programming experience is measured by asking participants to indicate the number of years they have been engaged in programming activities [22]. We also did not proceed with participants unless they completed the data structures course (or an equivalent) at the university.

Psychological Measures. To account for effects associated with mental health, we employed the Social Interaction Anxiety Scale (SIAS) [43] to establish a baseline anxiety level. The SIAS survey consists of 20 questions, each rated from 0 to 4, resulting in total scores ranging from 0 to 80. Generally, higher SIAS scores indicate elevated anxiety and fears related to general social interactions. In particular, when facing in-person interruptions involving face-to-face interactions, individuals with social interaction anxiety might experience greater challenges in effectively managing the interruptions. They may interpret interruptions as more threatening or disruptive, leading to heightened stress levels. Our participant pool is diverse, encompassing a range of SIAS scores as shown in Figure 4, allowing us to explore the potential impact of individual mental health on developers' responses to interruptions. This understanding can inform the development of targeted interventions and support mechanisms for developers in software development environments. To assess the frequency with which participants experienced cognitive failures, we included the Cognitive Failures Questionnaire (CFQ) [12]. The CFQ measurement contain 25 questions rated from 0 to 4, yielding a score from 0 to 100. Scores on the scale predict episodes of absent-mindedness, including slow performance on focused attention tasks, work accidents, and forgetting to save one's data on the computer. All participants scored less than 60, which suggests that, on average, they experienced relatively fewer cognitive failures or lapses in their cognitive functioning.

3.3 Software Engineering Assessment

After completing the pre-survey, participants scheduled a 2-hour in-person session. During their 2-hour session, the participant completes three different software tasks in an office setting with a lab computer that collects their responses. The assessment contains (1) a code writing (programming) task, (2) a code comprehension task, and (3) a code review task, all implemented in C++.

Prior to commencing the first task, participants were instructed to relax and view a soothing video featuring natural scenes, lasting two minutes and 40 seconds, following the methodology of Fritz et al.'s study [24]. This approach was chosen as it has been demonstrated to effectively return participants' physiological features to a baseline level after approximately one minute.

We implemented the assessment instrument using Python Flask. Participants view a web interface containing relevant elements for completing each task, such as text entry fields with C++ syntax highlighting, buttons for building and executing code against a held-out test suite, and buttons for accepting or rejecting proposed code changes. We describe each task below.

3.3.1 Code Writing Task. In the code writing task, the participant is asked to program a Tic-Tac-Toe game in C++ that involves a single file implementation with about 137 lines of code to read, which includes pre-defined functions and structures and instructions. The participant had to complete support for a two-player 3x3 Tic-Tac-Toe game in which players take turns specifying their symbols on the board and validating when a player formed a horizontal, vertical, or diagonal sequence of their symbol. We showed a video introducing the requirements of the Tic-Tac-Toe game before starting.

We created a web-based Integrated Development Environment (IDE) similar to platforms such as LeetCode [39]. We provided enough structure to the participant that they only needed to complete specific functions defined in the starter code according to the requirements they were given. For example, participants had to implement logic to evaluate the board state to determine if a winning condition had been met. This IDE enabled users to submit their code by clicking a button — the browser would submit the code to our server, which would automatically build and evaluate their code against a held-out test suite of 5 test cases. Participants were shown the output of the test cases so that they could refine their solution. For expediency and to encourage thoughtful task completion, we limited participants to 5 submissions total. We stored all submissions and test outputs, as well as every individual keystroke made by the participant.

3.3.2 Code Comprehension Task. In the code comprehension task, participants were presented with three sets of C++ code snippets, each corresponding to a LeetCode problem. The problems were carefully selected to cover a diverse range of important topics in software engineering and were categorized as either easy or medium difficulty based on LeetCode's classification [39]. The order of the code snippets was randomized for each participant to minimize any potential order effects. Table 1 summarizes the three LeetCode problems selected for this task. To gauge the difficulty level of each problem, we considered the acceptance rate, which represents the percentage of LeetCode submissions that pass all the test cases for a particular problem. LeetCode problems with lower acceptance rates are generally considered more challenging, as they may require a deeper understanding of the problem and a more sophisticated solution. The acceptance rate served as an approximation for the difficulty level of each problem.

Table 1: Description of code comprehension problems [39].

Problem Titles	Difficulty	Acceptance %	Related Topics
Single Number	Easy	70.9%	Array, Bit Manipulation
Majority Element	Easy	63.9%	Array, Hash Table, Divide and Conquer, Sorting, Counting
Subarray Sum Equals K	Medium	43.6%	Array, Hash Table, Prefix Sum

The participants are shown two different approaches to solving the same LeetCode-style coding problem. To assess their comprehension, participants were required to answer four questions related to each pair of approaches: two questions related to expected output given a specific input, and two qustions focused on time complexity (i.e., Big-O notation) of the provided approaches. Participants had the opportunity to submit their answers up to 5 times for expediency and to accommodate issues with formatting (e.g., specifying Big-O notation in a brower text input field is nontrivial). After each submission, the number of questions they answered correctly is displayed. An example of the questions that participants encountered for a given problem is shown in Figure 1.

3.3.3 Code Review Task. The code review task contains two parts:

- First, participants are asked to write test cases for a C++ Adelson-Velsky and Landis (AVL) tree implementation in a single C++ file of approximately 250 lines [7].
- Second, participant decide whether to accept proposed changes made to this AVL tree implementation. To ensure a clear understanding of the different rotations of AVL trees, we provided an introductory video on AVL trees before they started the code review task.

First, participants write test cases to cover four different types of AVL tree rotations (i.e., Left, Right, Left-Right, and Right-Left Rotations [20]) and one edge case in which the same value is inserted twice. Similar to the code writing task, participants use a web-based IDE to write test cases as unit tests. We seeded defects in the AVL tree implementation to contain logical errors that are corrected by proposed changes described in the second part. Participants can submit their test cases, which are evaluated with respect to line and branch coverage over the given implementation. The coverage is provided to the participant by showing whether each rotation type or edge case was evaluated by their provided test suite. Participants could submit their test cases up to five times.

Second, participants are informed that there are defects in the provided AVL tree implementation, and must determine whether to accept or reject proposed changes to the implementation. Sadoski et al. [52] showed that many changes to source code made during code reviews were relatively small in size, often modifying only a single line of code. Thus, we show participants 7 different proposed changes to the AVL tree implementation, each of which involved 4 lines of code or fewer. While two of these proposed changes fix existing errors, the other five changes introduce new defects. The combination of both fixes and the introduction of new defects ensures a comprehensive evaluation of participants' code review capabilities. Participants are required to evaluate each proposed change and decide whether to accept it based on its impact on the code's correctness and functionality.

3.4 Interruptions

We implemented interruptions based on previous methods. McFarlane examines four methods for deciding when to interrupt someone during multi-tasked computing, including immediate (requiring an immediate user response), negotiated (user chooses when to attend), mediated (an intelligent agent might determine when best to interrupt) and scheduled (interruptions come at prearranged time intervals) interruptions [44]. Those four methods of interruptions are based on multiple complex factors such as the urgency of the information being conveyed, the participant's preference or availability, and the participant's behavior and context. We adapt McFarlane's framework in designing interruptions of software engineering tasks in a controlled setting.

In our study, we implement four types *on-screen* interruptions (an example is shown in Figure 2) and two types *in-person* interruptions. We further adopt the Eisenhower decision matrix [35] to categorize our interruption methods. Similar to the two dimensions of Eisenhower decision matrix, this matrix evaluates interruptions along two dimensions: *urgency of request* and *dominance of requester*. Figure 3 depicts the estimated urgency of request and dominance of requester for the six interruptions. These interruptions are described in Table 2.

During the execution of the software engineering tasks, we randomly show on-screen interruptions, labeled as On-screen 1–4, which appeared randomly and in various orders throughout the different tasks. On-screen 1 (Experiment Invitation) and On-screen 3 (Sum-up Meeting) were intentionally designed to exhibit a high level of dominance from the requester, simulating messages from the Principal Investigator (PI), while On-screen 2 (ML Ads) and On-screen 4 (Post-survey Reminder) exhibit low dominance of the requester. Additionally, On-screen 3 was crafted with a sense of urgency, requiring participants to promptly fill out their availability. Each on-screen interruption message consists of no more than 100 words. These design choices were made to approximate aspects of real-life real-life situations where software engineers encounter interruptions from authority figures or urgent requests.

Table 2: Description of interruption types used in this study.

Label	Description of the Content
On-screen 1	A message claims the PI wants to invite the participant to another experiment.
On-screen 2	An advertisement to invite the participant to a seminar.
On-screen 3	A message asking the participant to provide their availability for a follow-up meeting about the study.
On-screen 4	A reminder to fill out the post-task survey after the task.
In-person 1	Student confederate enters the room to check on the participant.
In-person 2	Professor confederate enters the room to ask the participant about their availability to meet after the study.

In addition, we also consider in-person interruptions facilitated by a confederate student and a professor. For In-person 1 (Student), the confederate student, a 20-year old Asian female, played the role of a peer entering the room to assess the participant's progress during the task (although all progress is stored and tracked on the backend). For In-person 2 (PI), the confederate professor, a 34-year old White male who is a faculty in the department, played the role of an authority figure entering the room to ask the participant about their availability to meet after completing the study. Participants are aware of the confederate professor's occupation as the confederate introduces himself at the beginning of the interruption. Both these In-person interruptions were intentionally designed with a high sense of urgency, demanding immediate responses, but with a varying degree of requester dominance (i.e., student vs. professor confederate). To keep the consistent study design, the student and professor confederate remain the same for all participants (see Section 5.2 for discussions on limitations). The specific timing of their entry into the room was also randomized to mimic unexpected interruptions that engineers may encounter in their actual work environments.

We note that participants experienced one to three interruptions during each task, and we ensured that two interruptions did not occur simultaneously. This approach aimed to approximate aspects of real-life real-world interruptions, where engineers may face multiple interruptions over time. Our design choices, incorporating both on-screen and in-person interruptions, were intended to capture the diverse effects of interruptions from various sources and contexts on participants' software engineering activities. By simulating these interruptions realistically, we aimed to create an environment that closely resembles the challenges and distractions software engineers may encounter during their day-to-day work.

3.5 Physiological Measures

Physiological measures such as electrodermal activity and heart-related measures provide a objective way to measure each participant's emotional state and cognitive load [1, 26]. We use the Empatica EmbracePlus Wristband to measure heart rate variability (HRV) as it has been widely used in research to collect physiological data [25, 51, 54, 62]. It is a medical-grade wearable device that offers real-time physiological data acquisition, which we use to conduct in-depth modeling, analysis and visualization. The wristband embeds 3-axis accelerometer, electrodermal activity, temperature and gyroscope sensors[2]. It is a non-invasive device worn like a wristwatch, minimizing any discomfort during usage. However, participants are informed of potential mild irritation due to prolonged contact with the band material on the skin, as well as possible fatigue from wearing the wristband. These risks are similar to those associated with smartwatch devices. All data collected from the wristband was stored in encrypted storage and de-identified from each participant.

3.6 Post-task Survey

During the study, participants complete three post-task surveys interspersed after each task completion. We note that participants were unaware that the interruptions were intentional during the study. Therefore, in compliance with the IRB protocol, a debriefing

[2]https://www.empatica.com/embraceplus/

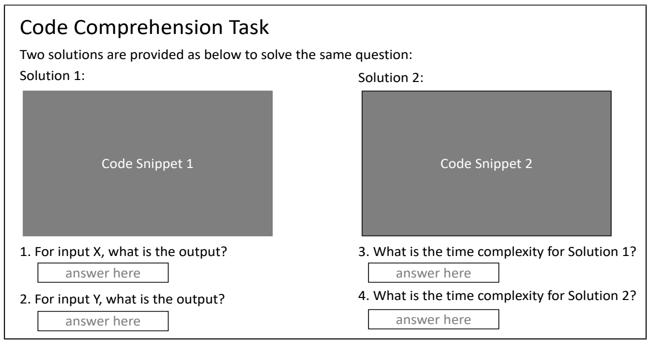
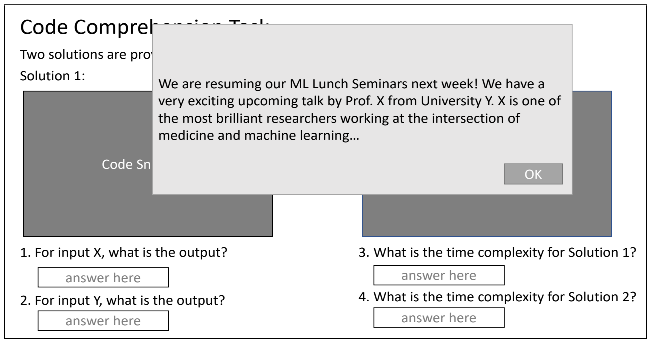

Code Comprehension Task
Two solutions are provided as below to solve the same question:
Solution 1:
Solution 2:
Code Snippet 1
Code Snippet 2
1. For input X, what is the output?
answer here
2. For input Y, what is the output?
answer here
3. What is the time complexity for Solution 1?
answer here
4. What is the time complexity for Solution 2?
answer here

Figure 1: Layout of the code comprehension task stimulus. Each code snippet provided in this assessment consists of fewer than 30 lines of code shown in a browser.


Code Compreh
Two solutions are pro
Solution 1:
Code Sn
OK
1. For input X, what is the output?
answer here
2. For input Y, what is the output?
answer here
3. What is the time complexity for Solution 1?
answer here
4. What is the time complexity for Solution 2?
answer here

We are resuming our ML Lunch Seminars next week! We have a very exciting upcoming talk by Prof. X from University Y. X is one of the most brilliant researchers working at the intersection of medicine and machine learning...

Figure 2: Layout of an on-screen interruption. The light gray box depicts the pop-up that appears on the screen randomly during the task, which can be dismissed by clicking the 'OK' button or providing the requested input.

session is conducted at the conclusion of the study to inform participants about the purpose and design of the study, including the intentional nature of the interruptions. Following the debriefing, participants are asked to complete a final survey, which provides them with an opportunity to reflect on their overall experience and provide any additional feedback or comments.

3.6.1 Code Writing Survey. The first post-task survey is associated with the tic-tac-toe writing task. We ask participants' current affective states by adapting the Positive and Negative Affect Schedule (PANAS) scale [61]. The PANAS scale consists of 20 items that are rated on a scale of 1 to 5. These items are divided into two categories: positive affect and negative affect, with 10 items in each category. However, as the PANAS scale is commonly used to assess a person's affective state over a longer period, following previous studies, we made necessary modifications to capture participants' immediate affective experiences during the task [15]: we asked them to select the PANAS items that best described their current affective state while working on the tic-tac-toe task. This adaptation allowed us to capture participants' current emotional state in response to the task and any encountered interruptions. Additionally, to gain insights into participants' perceived productivity and level of distraction or focus during the task, we included rating scales from 1 to 5 on

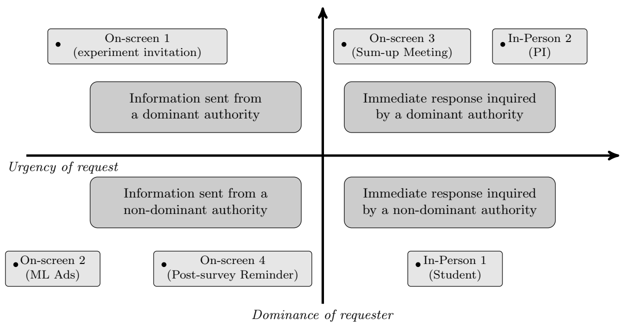

On-screen 1 (experiment invitation)
On-screen 3 (Sum-up Meeting)
In-Person 2 (PI)
Information sent from a dominant authority
Immediate response inquired by a dominant authority
Urgency of request
Information sent from a non-dominant authority
Immediate response inquired by a non-dominant authority
On-screen 2 (ML Ads)
On-screen 4 (Post-survey Reminder)
In-Person 1 (Student)
Dominance of requester

Figure 3: Interruption Matrix Based on Eisenhower Decision Matrix. This matrix evaluates interruptions along two dimensions: urgency of request and dominance of requester.

these factors. Open-ended questions were also included to gather more detailed qualitative feedback, such as "What was the biggest challenge you faced while implementing the tic-tac-toe task and why?" These open-ended questions provided participants with an opportunity to share their experiences, challenges, and perceptions related to the task and the interruptions they encountered.

3.6.2 Code Comprehension Survey. After completing the code comprehension task, we asked PANAS scale items (as in the code writing task). We also asked participants to rate the difficulty of each problem on a scale of 1–100 using a slider element and to rank the difficulty of the three problems.

3.6.3 Code Review Survey. After the code review task, we follow a similar survey structure. Participants were asked to complete the PANAS scale, and then to rate their productivity, distractedness, and level of focus, each on a scale of 1 to 5, and provide feedback through open-ended questions. They were also asked to rank the difficulty for the 7 accept/reject changes questions.

3.6.4 Debriefing Survey. Finally, as participants learned that all interruptions were purposeful, we asked them to rate the level of distraction and level of stress caused by those intentional interruptions. We also included open-ended questions such as "Other than the distractions we designed on purpose, was there anything else you found distracted?" to allow participants to provide feedback.

By including these post-task surveys, we aimed to gather comprehensive feedback and insights from participants regarding their perceived affective states, task perceptions, and experiences during each task in the study.

4 EVALUATION

In this section, we present a comprehensive analysis of the data collected during our study on the impact of interruptions on developers' performance and physiological measures in software engineering tasks. Our participant pool comprised a group of 20 undergraduate (n=17) and graduate (n=3) computer science students, with 11 male and 9 female, ranging in age from 19 to 23 years. To assess their social interaction anxiety levels, we employed the Social Interaction Anxiety Scale (SIAS), the scores of which are

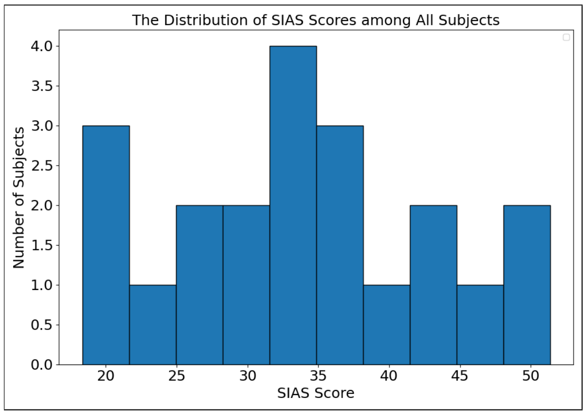


Figure 4: Distribution of SIAS among all subjects. These scores indicate the level of social interaction anxiety experienced by the participants, with higher scores indicating a higher level of anxiety. The SIAS scores of our subjects ranged from 20 to 53, with a mean of 35.35 (SD = 9.64).

distributed as depicted in Figure 4. The distribution aligns with previous studies on social anxiety among general engineering students in colleges [16, 34]. We aim to address our research questions:

RQ1: What is the effect of different interruptions on developers' performance across different software engineering tasks?

RQ2: What is the effect of different interruptions on stress measures across different software engineering tasks?

RQ3: Is there a correlation between objective measures of developers' physiological states and their self-reported feelings for different interruptions and tasks?

4.1 Performance and Productivity

In this study, we first measure time spent on each task as an indication of participant efficiency and productivity. Informally, a developer who completes tasks within a shorter time frame may be considered more productive than one who takes longer. When measuring time spent on tasks, we exclude in-person interruption time as participants are not able to work on the task during in-person interruptions. Furthermore, relying solely on one metric can lead to an incomplete understanding of productivity [23]. It is crucial to use a combination of quantitative and qualitative measures to obtain a well-rounded assessment of a developer's productivity. Therefore, we also assess their self-reported productivity through post-task surveys, although no significant result was found. Below, we discuss specific results of participant productivity with respect to the Code Comprehension and Code Review tasks (Section 3.3).

Code Comprehension. There are three problems in code comprehension task as shown in Table 1. To investigate the effect of different interruptions, we conducted a rigorous mixed-effects modeling analysis. We first investigated the relationship between types of interruptions and time spent on the problem. We specified that there is individual variability in the baseline level of the time spent among different participants. Subsequently, we employed linear mixed-effects models to predict the time spent on the problem. The model results suggest that On-screen 1 (Experiment Invitation) and On-screen 3 (Sum-up Meeting) have a significant effect on time spent. For problem with the occurrence of On-screen 1 (Experiment Invitation), the expected difference of time spent on the problem is 164.50 seconds (SE = ±72.82, $p = 0.028$) compared to the time spent on the problem without any interruption. For problem with the occurrence of On-screen 3 (Sum-up Meeting), the expected difference of time spent on the problem is 164.50 seconds (SE = ±62.90, $p = 0.0499$) compared to the time spent on the problem without any interruption. These findings conclusively demonstrate that encountering these two types of interruptions, both with high dominance of requester, led to a significant increase in the time participants spent on the problem. For in-person interruptions, we find no significant effect on time spent.

In addition to examining the impact of interruptions, we sought to explore how the difficulty of the problems influences developers' responses during the code comprehension task. To address this, we performed separate ANOVA analyses for each problem, comparing the average time spent by participants when encountering interruptions versus completing the problem without any interruption, while also accounting for individual differences. The outcomes shown in Table 3 demonstrate that interruptions had a more significant effect on simpler problems compared to their impact on more complex ones. This observation suggests that the presence of interruptions had a more pronounced influence on participants' performance and time allocation for easier problems, while the effect was relatively diminished for problems of higher difficulty.

Code Review. In the code review task, detailed in Section 3.3.3, we introduced a combination of in-person and on-screen interruptions. To assess the impact of these interruptions on the time spent during the code review, we employed a rigorous mixed-effects ANOVA analysis. This allowed us to investigate the potential significance of different types of in-person and on-screen interruptions, as well as their interaction. The results of the mixed-effects ANOVA analysis indicated a significant interaction effect ($p = 0.043$) between in-person and on-screen interruptions on the time spent during the code review task. Although the individual main effects of in-person interruptions and on-screen interruptions were not statistically significant on their own, their combined influence significantly impacted the time spent during the code review process.

To understand the interaction, we conducted a post-hoc analysis, focusing on pairwise comparisons between different combinations of on-screen interruptions and in-person interruptions. The post hoc analysis revealed compelling insights into the differences in estimated means for time spent on the task under various interruption scenarios. When On-screen 2 (ML Ads) occurred in conjunction with in-person interruptions, participants spent 1812.3 seconds less

Table 3: Differences in mean time spent by groups with minus without interruptions (seconds). The percentage of the difference is shown in parentheses.

Problem Title	Difference (s)	Standard Error	p-value
Single Number	**230.0** (142.9%)	**±72.1**	**0.0053**
Majority Element	89.5 (40.5%)	±93.5	0.3522
Subarray Sum Equals K	24.2 (9.3%)	±56.6	0.6747

time (SE = ±711, p = 0.027) on the task compared to scenarios with no in-person interruptions. Conversely, without the presence of in-person interruptions, participants spent 1575 seconds more time (SE = ±487, p = 0.0346) when encountering On-screen 2 (ML Ads) compared to On-screen 4 (Post-survey Reminder).

These findings suggest that the interaction between in-person and on-screen interruptions plays a significant role in influencing the time spent during the code review task. The specific combinations of interruptions have varying effects on the task duration, indicating the complexity and nuanced impact of interruptions on developers' code review activities. By thoroughly examining both in-person and on-screen interruptions in this mixed-effects ANOVA analysis, we gain a comprehensive understanding of how different interruptions interact to affect developers' code review performance.

Finding 1: Specific on-screen interruptions with high dominance of requester significantly increase the time spent on code comprehension problems, and interruptions in general have a more pronounced effect on time spent on simpler code comprehension problems compared to more complex ones. The combined influence of in-person and on-screen interruptions significantly impact the time spent during the code review process, and specific combinations of interruptions result in varying effects on the task duration.

4.2 Heart Rate Variability and Stress Measures

Studies have shown the connection between heart rate variability (HRV)'s time domain features and stress levels in healthy human participants [17, 60]. In particular, the short-term SDNN and RMSSD have been found to exhibit significant changes in response to stress [13, 21, 31, 36, 49, 55]. SDNN is the standard deviation of Inter-Beat Intervals (IBIs) measured in milliseconds, where NN means "normal" beats, i.e, removing abnormal or false beats. RMSSD calculates the difference between successive inter-beat-intervals (IBI) in milliseconds, squares these values, and takes the root of the mean. Punita et al. [49] suggested that SDNN and RMSSD were reduced with increased intensity of stress, and Sin et al. found that individuals with more pronounced affective reactivity to stressors had lower levels of SDNN and RMSSD [55]. We are interested in examining the effect of types of interruptions on developers' stress measures and whether it varies among different types of software engineering tasks. We thus estimated participants' objective stress measures using HRV and compared them with their self-reported stress scores in the post-survey.

The participants' physiological data was collected using an Empatica EmbracePlus wristband during our study. Specifically, we focused on extracting time domain features of Heart Rate Variability (HRV) from the collected data. To extract the time domain features, we used `hrvanalysis`, a Python module specifically designed for HRV analysis. We captured the 30-second window both before and after each interruption that occurred during the assessment. The 30-second window measurement taken before the interruption was designated as the baseline for subsequent comparisons. By establishing a baseline, we aimed to assess any changes or deviations in physiological responses resulting from the interruptions.

First, we report the differences of SDNN and RMSSD for the measurements before and after each interruption. We perform paired t-tests with different types of interruptions and software engineering tasks. The results show that, in general, after an interruption, the participant's SDNN and RMSSD increase by 12.0 ms (SE = ±9.5, p = 0.013) and 14.6 ms (SE = ±11.7, p = 0.015), indicating a decrease in stress measures. Specifically, regardless of the type of tasks being performed, In-person 1 (Student entering the room) have a significant positive effect on RMSSD, causing RMSSD to increase by 32.6 milliseconds (SE = ±28.9, p = 0.029). The increase in RMSSD indicates a decrease in stress measures.

To further investigate the impact of different interruptions, we conducted a rigorous mixed-effects modeling analysis. Initially, we performed an ANOVA analysis using random effects structures to identify the best-fitting model to account for the inherent variability in our data. We treat individual subjects as random effects. This was achieved through the inclusion of random intercepts for tasks and random slopes for subjects influenced by the types of tasks they performed. After finalizing the random effects structure, we proceeded to fit linear mixed-effects models aimed at predicting SDNN and RMSSD. We consider the following two linear mixed effects models to determine whether types of interruptions and types of tasks can predict SDNN and RMSSD. By employing these models, we aim to gain deeper insights into the complex relationship between interruptions, software engineering tasks, and their joint influence on the participants' SDNN and RMSSD measures.

Model 1: Do types of interruptions predict SDNN and RMSSD? The model results suggest that the in-person interruptions have a significant impact on SDNN and RMSSD. For SDNN, the interruption "In-person 1" shows a statistically significant effect (p = 0.0098) with an expected difference of 25.4 milliseconds (SE = ±9.7) compared to the baseline measurements taken before the interruption occurred. Similarly, the interruption "In-person 2" also has a statistically significant effect (p = 0.0107) with an expected difference of 23.9 milliseconds (SE = ±9.2) compared to the baseline measurements. For RMSSD, The interruption "In-person 1" demonstrates a statistically significant effect (p = 0.0045) with an expected difference of 34.8 milliseconds (SE = ±12.1) compared to the baseline measurements taken before the interruption occurred. Additionally, the interruption "In-person 2" has a statistically significant effect (p = 0.0432) with an expected difference of 23.4milliseconds (SE = ±11.5) compared to the baseline measurements. Remarkably, both in-person interruptions are positive predictors, indicating that when these interruptions occur, the developers' SDNN and RMSSD tend to increase, and their stress measures tend to decrease. This suggests that the presence of students or the PI in the room may have a beneficial effect on the developers' physiological indicators, potentially reducing stress levels.

Model 2: Do types of interruptions and types of software engineering activities predict SDNN and RMSSD? The model suggests that both the type of interruptions, specifically the two in-person interruptions, and and the type of task have a significant impact on RMSSD and SDNN. During interruptions, the expected difference in SDNN compared to code writing task for code comprehension task is 25.0 milliseconds (SE = ±9.6, p = 0.0027); The expected difference in SDNN compared to code writing task for

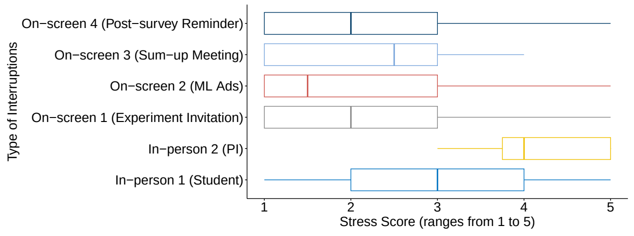


Figure 5: Self-perceived stress scores among all subjects. The graph illustrates the distribution of self-rated stress scores, ranging from 1 (least stressed) to 5 (most stressed).

code review task is 53.6 milliseconds (SE = ±7.5, $p = 4.8e^{-7}$). The expected difference in RMSSD compared to code writing task for code comprehension task is 23.9milliseconds (SE = ±9.4, $p = 0.0151$); the expected difference in RMSSD compared to code writing task for code review task is 54.8 milliseconds (SE = ±10.2, $p = 2.52e^{-5}$). These results suggest that developers experience lower stress measures during interruptions when performing code comprehension and code review tasks compared to the code writing task. The positive predictors for both tasks suggest lower stress measures, with the code review task showing a particularly pronounced effect. However, it is important to consider the sequential nature of the tasks performed. Since tasks were completed in a specific order, it is possible that developers became less stressed as they progressed to the later tasks.

Finding 2: Both the type of interruptions and the type of task significantly affect developers' objective stress measures. Specifically, interruptions in code comprehension and code review tasks are associated with lower stress measures compared to the code writing task. The presence of in-person interruptions has a significant positive impact on the physiological measures, suggesting a reduction in stress measures when these interruptions occur.

4.3 Comparing Self-Reported and Physiological Data

In the post-survey for each task, participants rate the level of stress caused by each interruption from 1 to 5, where 1 is the least stressed, and 5 is the most stressed. We noted a disparity between the objectively measured stress indicators and the self-reported stress scores. The distribution of reported stress scores is shown in Figure 5. In this subsection, we compare between these self-reported measures and the objective stress measurements reported in Section 4.2.

Using a linear mixed-effect model, we investigate the relationship between on-screen/in-person interruptions and participants' self-reported stress scores. The analysis revealed in-person interruptions have a significant effect on participants' self-reported stress scores, as participants' self-reported stress scores are 1.43 (SE = ±0.25, $p < 0.0001$) higher when they encountered in-person interruptions compared to on-screen interruptions.

Surprisingly, the findings contradict the objective physiological measurements. Participants, on average, perceived in-person interruptions as substantially more stressful than on-screen interruptions in contrast to their actual physiological data showing

Table 4: Pairwise comparisons for the self-reported stress (from 1 to 5) between different types of in-person and on-screen interruptions using Tukey's Honestly Significant Difference (HSD). Delta is calculated by subtracting the self-reported stress for on-screen interruptions from in-person interruptions. Significant results are highlighted in bold.

Delta	On-screen 1 (Exp. Inv.)	On-screen 2 (ML Ads)	On-screen 3 (Sum-up Mtg.)	On-screen 4 (Survey Rem.)
In-person 1 (Student)	1.617	1.492	1.242	0.965
In-person 2 (PI)	**3.064****	**2.939****	**2.689****	**2.412***

($^{*}p < 0.05$, $^{**}p < 0.01$)

their stress measures decrease during in-person interruptions (Section 4.2). This discrepancy prompted further investigation into potential factors influencing this contradiction.

We then performed an ANOVA analysis on the effect of specific types of interruptions on stress scores with the use of nested error term to appropriately account for the dependency between repeated measures on the same subjects: the different types of interruptions have a significant impact on participants' self-perceived stress score ($p < 0.001$). We then used Tukey's Honestly Significant Difference (HSD) Method to perform pairwise comparisons for the self-perceived stress score between groups, as shown in Table 4. The results suggest that the In-person 2 (PI) interruption type is associated with higher self-perceived stress scores compared to On-screen 1–4 interruption, but there is no significant difference in self-perceived stress scores between In-person 1 (Student) and any On-screen interruption (see Section 5.2 for limitations).

Participants' affective states for each task were assessed using the PANAS scale, which includes 10 positive and 10 negative items. Positive scores were incremented by 1 for each positive item selected, while negative scores were incremented by 1 for each negative item selected. However, we found no significant differences in positive and negative scores across the three tasks. This result suggests that participants' perceived affective states remained relatively consistent throughout the different software engineering tasks, regardless of the presence of interruptions.

The observed discrepancy between self-perceived stress scores is not limited to different types of interruptions but also extends to different tasks. Specifically, based on the results from Section 4.2, developers appear to experience a higher SDNN and RMSSD, indicating lower stress measures when engaged in code comprehension and code review tasks compared to the code writing task according to their physiological data. Interestingly, despite the lower objective stress measures during the code review task, we note that in the post-survey's open-ended question, 9 out of 20 participants mentioned some degree of disliking for the code review task. These contrasting results raise interesting questions regarding the relationship between stress, task preferences, and subjective experiences. While the quantitative data point towards lower stress during code review, the qualitative responses highlight participants' aversion or discomfort towards this particular task.

The combination of objective physiological data and self-reported measures provides a comprehensive understanding of the impact of interruptions on developers' stress measures and affective states during software engineering tasks. The observed discrepancy between subjective self-perceived stress scores and objective physiological responses underscores the potential need for considering multiple measures when evaluating stress in the software development context.

Finding 3: The self-perceived stress levels reported by participants show a discrepancy against objective physiological data. While developers reported higher stress levels during in-person interruptions compared to on-screen interruptions, their objective physiological data indicate a lower stress level. While participants' physiological responses suggest that they experienced lower stress levels during code comprehension and code review tasks compared to code writing tasks, about half of the participants reported aversion towards the code review task. This might suggest the necessity to further understand the relationship between stress, task preferences, and subjective experiences in software development environments.

5 DISCUSSION

In this section, we discuss our study's implications for the software engineering community (Section 5.1) as well as the threats to validity (Section 5.2) to identify potential limitations and weaknesses that could affect the validity of the study's findings.

5.1 Implications

Our study's results have several implications for both researchers and practitioners in the field of software engineering.

First, the study reveals that different interruptions influence developers' stress measures and productivity during various software engineering tasks. Companies may consider adopting interruption management strategies to minimize stress. For example, providing customizable notification mechanisms for on-screen interruptions and implementing interruption-free periods for developers working on critical tasks can foster a less stressful work environment.

The study further opens up new research avenues for exploring the complex effect of interruptions on stress and task performance in software development. Further investigations into productivity, stress management techniques, and the effects of different types of interruptions can lead to more tailored interventions and strategies to support developers in their work.

5.2 Threats to Validity

In this subsection, we address four threats to validity in our study. First, the nature of the interruptions used in our study, while inspired by real-life scenarios, may not fully capture the complexity and nuance of interruptions that occur in natural work settings. In a laboratory setting, certain aspects such as the intensity, frequency, and unpredictability of real-life interruptions may not be entirely replicable. While an in-situ study design could provide insights more reflective of real-life interruption dynamics, it also introduces complexities in controlling and measuring variables. This study chose a lab-based approach to balance the need for experimental control with the objective of examining interruption effects. Future research could explore in-situ methodologies to complement and extend our findings. Next, we used the Empatica EmbracePlus wristband to measure participants' physiological responses, which inherently entails several limitations and measurement errors that could affect the HRV measurements. Despite this, the Empatica EmbracePlus wristband has been widely used in previous research (Section 3.5), with multiple studies supporting its validity [51, 53, 57]. Third, the sample size of 20 participants is a potential threat to external validity. We chose 20 participants after considering the study's complexity and time requirements. Previous studies with similar sample sizes have found significant results in studying physiological measures and stress [58, 59]. This participant count allowed for in-depth analysis while managing the logistical challenges of intensive data collection and personalized attention during the lab sessions. Further, we note that we have reported statistically significant findings from our 20 participant cohort. Moreover, we note a potential limitation lies in the relatively small number of problems in the code comprehension task. We initially included an additional hard Leetcode-style problem, but excluded it based on pilot study results due to extended and unpredictable completion times. While this allowed us to examine the impact of interruptions on stress and productivity across different complexity levels, the limited number of problems may affect the generalizability of our findings. Moreover, approximating problem difficulty based on acceptance rate could be influenced by factors beyond inherent difficulty, such as problem popularity. Nonetheless, by carefully selecting problems of varying complexity levels, our findings remain relevant for understanding the intricate relationship between interruptions, task complexity, and developer experiences in software development environments. Lastly, the design of the dominance of requester for in-person interruptions might be affected due to stereotypes and implicit biases about gender and race. There could be societal expectations or stereotypes about how different genders and races communicate, which could impact how interruptions from a white male professor and an Asian female student are interpreted.

6 CONCLUSION

We conducted a controlled study on the effects of interruptions on software engineering tasks. Participants completed code writing, code comprehension, and code review tasks while experiencing six different in-person and on-screen interruptions. We collected their physiological data with the Empatica EmbracePlus wristband and gathered self-perceived evaluations through surveys. Our findings reveal that specific on-screen interruptions, especially those with a high dominance of requester, significantly increase the time spent on code comprehension tasks. The combined influence of in-person and on-screen interruptions significantly impact the time spent during the code review process, with various interruption combinations leading to different effects on task duration. Developers' stress measures are affected by the type of interruptions and tasks. Code comprehension and code review tasks are associated with higher physiological measures, and thus lower stress measures compared to the code writing task. Surprisingly, in-person interruptions positively impact physiological measures, indicating reduced stress measures. However, participants' self-perceived stress scores do not align with the objective physiological data. Developers reported higher stress scores during in-person interruptions, but physiological data suggests otherwise.

These results shed light on the nuanced impact of interruptions on developers' performance and stress measures during software engineering tasks. Understanding these complexities can inform the design of interruption management strategies, task assignments, and stress reduction interventions in software development settings. The findings also underscore the importance of considering both objective physiological data and self-perceived stress measures when evaluating developers' well-being and productivity.

7 ACKNOWLEDGEMENT

This research was supported in part by NSF grant CCF-2211429. We also thank all the participants in the study and members in our research group at Vanderbilt University for their generous help on pilot experiments.

REFERENCES

[1] S. Wilk A. Lisowska and M. Peleg. 2021. Is it a good time to survey you? Cognitive load classification from blood volume pulse. *2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS)* (2021), 137–141.

[2] Zahra Shakeri Hossein Abad, Mohammad Noaeen, Didar Zowghi, Behrouz H. Far, and Ken Barker. 2018. Task Interruption in Software Development Projects: What Makes some Interruptions More Disruptive than Others? *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering (EASE'18)*, 122–132.

[3] Zahra Shakeri Hossein Abad, Mohammad Noaeen, Didar Zowghi, Behrouz H. Far, and Ken Barker. 2018. Two Sides of the Same Coin: Software Developers' Perceptions of Task Switching and Task Interruption. *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering (EASE'18).*

[4] Zahra Shakeri Hossein Abad, Guenther Ruhe, and Mike Bauer. 2017. Task Interruptions in Requirements Engineering: Reality versus Perceptions! *Requirements Engineering Conference (RE), 2017 IEEE 25th International. IEEE*, 6–15.

[5] Zahra Shakeri Hossein Abad, Guenther Ruhe, and Mike Bauer. 2017. Understanding Task Interruptions in Service Oriented Software Development Projects: An Exploratory Study. *Proceedings of the 4th International Workshop on Software Engineering Research and Industrial Practice (SER&IP '17)*, 34–40.

[6] 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. *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, 442–447.

[7] G.M. Adel'son-Vel'skiy and Ye M Landis. 1962. An algorithm for the organization of information. *Deklady Akademii Nauk USSR* 16, 2 (1962), 263–266.

[8] B. P. Bailey, J. A. Konstan, and J. V. Carlis. 2001. The effects of interruptions on task performance, annoyance, and anxiety in the user interface. *Int. Conf. Human-Comput. Interaction*, 593–601.

[9] S. R. Barley, D. E. Meyerson, and S. Grodal. 2011. E-mail as a source and symbol of stress. *Organization Science* 22, 4 (2011), 887–906.

[10] Moritz Beller, Vince Orgovan, Spencer Buja, and Thomas Zimmermann. 2021. Mind the Gap: On the Relationship Between Automatically Measured and Self-Reported Productivity. *IEEE Software* 38, 4 (2021), 24–31.

[11] Alexander Bick, Adam Blandin, and Karel Mertens. 2020. Work from home after the COVID-19 outbreak. *Psychological Assessment* (2020).

[12] D.E. Broadbent, P.F. Cooper, P. FitzGerald, and K.R. Parkes. 1982. The Cognitive Failures Questionnaire (CFQ) and its correlates. *British Journal of Clinical Psychology* 21 (1982), 1–16.

[13] T. Chandola, A. Britton, E. Brunner, H. Hemingway, M. Malik, M. Kumari, E. Badrick, M. Kivimaki, and M Marmot. 2008. Work stress and coronary heart disease: what are the mechanisms? *European heart journal* 29, 5 (2008), 640–648.

[14] Huei-Yen Winnie Chen, Liberty Hoekstra-Atwood, and Birsen Donmez. 2018. Voluntary- and Involuntary-Distraction Engagement: An Exploratory Study of Individual Differences. *Epub* 60, 4 (2018), 533–541.

[15] P. I. Chow, K. Fua, Y. Huang, W. Bonelli, H. Xiong, L. E. Barnes, and B. A. Teachman. 2017. Using mobile sensing to test clinical models of depression, social anxiety, state affect, and social isolation among college students. *Journal of Medical Internet Research* 19, 3 (2017), 215–226.

[16] Philip I Chow, Karl Fua, Yu Huang, Wesley Bonelli, Haoyi Xiong, Laura E Barnes, and Bethany A Teachman. 2017. Using mobile sensing to test clinical models of depression, social anxiety, state affect, and social isolation among college students. *Journal of medical Internet research* 19, 3 (2017), e62.

[17] E. Clays, D. De Bacquer, V. Crasset, F. Kittel, P. de Smet, M. Kornitzer, R. Karasek, and G. G. De Backer. 2011. The perception of work stressors is related to reduced parasympathetic activity. International archives of occupational and environmental health. *International archives of occupational and environmental health* 84, 2 (2011), 185–191.

[18] S Connelly, Hasher Lisa, Zacks Lynn, and Rose T. 1991. Age and reading: The impact of distraction. *Psychology and Aging* 6, 4 (1991), 533–541.

[19] L. A. Dabbish and R. E. Kraut. 2006. Email overload at work: an analysis of factors associated with email strain. *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work.*, 431–440.

[20] Ellis. 1980. Concurrent Search and Insertion in AVL Trees. *IEEE Trans. Comput.* C-29, 9 (1980), 811–817. https://doi.org/10.1109/TC.1980.1675680

[21] C.K. Endukuru and S. Tripathi. 2016. Evaluation of cardiac responses to stress in healthy individuals- a non invasive evaluation by heart rate variability and stroop test. *Int J Sci Res* 5 (2016), 286–289.

[22] J. Feigenspan, C. Kastner, J. Liebig, S. Apel, and S. Hanenberg. 2012. Measuring programming experience. *2012 20th IEEE International Conference on Program Comprehension (ICPC'12)*, 73–82.

[23] Nicole Forsgren, Margaret-Anne Storey, Chandra Maddila, Thomas Zimmermann, Brian Houck, and Jenna Butler. 2021. The SPACE of Developer Productivity: There's more to it than you think. *Queue* 19, 1 (2021), 20–48.

[24] T. Fritz, A. Begel, S. C. Müller, S. Yigit-Elliott, and M. Züger. 2014. Using psycho-physiological measures to assess task difficulty in software development. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, 402–413.

[25] Daniela Girardi, Nicole Novielli, Davide Fucci, and Filippo Lanubile. 2020. Recognizing Developers' Emotions While Programming *(ICSE '20)*. Association for Computing Machinery. https://doi.org/10.1145/3377811.3380374

[26] R. Gokay, E. Masazade, C. Aydin, and D. Erol-Barkana. 2015. Emotional state and cognitive load analysis using features from BVP and SC sensors. *2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)* (2015), 178–183.

[27] D. Graziotin, Fabian Fagerholm, X. Wang, and P. Abrahamsson. 2018. What happens when software developers are (un)happy. *Journal of Systems and Software* 140 (2018), 32–47.

[28] D. Graziotin, X. Wang, and P. Abrahamsson. 2014. Happy software developers solve problems better: psychological measurements in empirical software engineering. *PeerJ* 2 (2014), 289.

[29] D. Graziotin, X. Wang, and P. Abrahamsson. 2015. Do feelings matter? on the correlation of affects and the self-assessed productivity in software engineering. *Journal of Software: Evolution and Process* 27, 7 (2015), 467–487.

[30] P. Hallam. 2006. What do programmers really do anyway?. In *Microsoft Developer Network (MSDN) — C# Compiler.*

[31] M. Hintsanen, M. Elovainio, S. Puttonen, M. Kivimaki, T. Koskinen, O. T. Raitakari, and L Keltikangas-Jarvinen. 2007. Effort-reward imbalance, heart rate, and heart rate variability: the Cardiovascular Risk in Young Finns Study. *International journal of behavioral medicine* 14, 4 (2007), 202–212.

[32] Yu Huang, Kevin Leach, Zohreh Sharafi, Nicholas McKay, Tyler Santander, and Westley Weimer. 2020. Biases and Differences in Code Reviews using Medical Imaging and Eye-Tracking: Genders, Humans, and Machines. In *Proceedings of Foundations of Software Engineering (ESEC/FSE) (FSE '20).*

[33] Yu Huang, Xinyu Liu, Ryan Krueger, Tyler Santander, Xiaosu Hu, Kevin Leach, and Westley Weimer. 2019. Distilling Neural Reresentations of Data Structure Manipulation using fMRI and fNIRS. In *Proceedings of the 41st ACM/IEEE International Conference on Software Engineering (ICSE '19).*

[34] Yu Huang, Haoyi Xiong, Kevin Leach, Yuyan Zhang, Philip Chow, Karl Fua, Bethany A Teachman, and Laura E Barnes. 2016. Assessing social anxiety using GPS trajectories and point-of-interest data. In *Proceedings of the 2016 ACM international joint conference on pervasive and ubiquitous computing.* 898–903.

[35] N. S. Jyothi and A. Parkavi. 2016. A study on task management system. *2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS)* (2016).

[36] M. G. Kang, S. B. Koh, B. S. Cha, J. K. Park, J. M. Woo, and S. J. Chang. 2004. Association between job stress on heart rate variability and metabolic syndrome in shipyard male workers. *Yonsei medical journal* 45, 5 (2004), 838–846.

[37] Ryan Krueger, Yu Huang, Xinyu Liu, Tyler Santander, Westley Weimer, and Kevin Leach. 2020. Neurological Divide: An fMRI Study of Prose and Code Writing. In *Proceedings of the 42nd ACM/IEEE International Conference on Software Engineering (ICSE '20).*

[38] Nilli Lavie. 2010. Attention, Distraction, and Cognitive Control Under Load. *Current Directions in Psychological Science* 19 (2010), 143–148.

[39] LeetCode. 2023. LeetCode - The World's Leading Online Programming Learning Platform.

[40] S. Leroy, A. M. Schmidt, and N. Madjar. 2021. Working from home during COVID-19: A study of the interruption landscape. *Journal of Applied Psychology* 106, 10 (2021), 1448–1465.

[41] Yi-Jen Lin and Frank W. Wicker. 2007. A comparison of the effects of thought suppression, distraction and concentration. *Behaviour Research and Therapy* 45 (2007).

[42] G. Mark, D. Gudith, and U. Klocke. 2008. The cost of interrupted work : More speed and stress. *SIGCHI Conf. Human Factors Comput*, 107–110.

[43] R. P. Mattick and J. C. Clarke. 1998. Development and validation of measures of social phobia scrutiny fear and social interaction anxiety. *Behaviour Research and Therapy* 36, 4 (1998), 450–470. https://doi.org/10.1016/S0005-7967(97)10031-6

[44] D. McFarlane. 1999. Coordinating the interruption of people in human-computer interaction. *HumanComputer Interaction - INTERACT'99,* (1999), 295–303.

[45] A. Meyer, E. T. Barr, C. Bird, and T. Zimmermann. 2019. Today was a good day: The daily life of software developers. *Transactions on Software Engineering(TSE)* (2019).

[46] A. Meyer, L. Barton, G.C. Murphy, T. Zimmermann, and T. Fritz. 2017. The Work Life of developers: Activities, switches and perceived productivity. *Transactions on Software Engineering(TSE)* (2017).

[47] Sebastian C. Müller and Thomas Fritz. 2015. Stuck and Frustrated or in Flow and Happy: Sensing Developers' Emotions and Progress. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* 1 (2015), 688–699.

[48] E. Murphy-Hill, C. Jaspan, C. Sadowski, D. Shepherd, M. Phillips, C. Winter, A. Knight, E. Smith, and M. Jorde. 2019. What Predicts Software Developers' Productivity? *Transactions on Software Engineering (TSE)* (2019).

[49] P Punita, K Saranya, and SS Kumar. 2016. Gender difference in heart rate variability in medical students and association with the level of stress. *Natl J Physiol Pharm Pharmacol* 6 (2016), 431–437.

[50] S. Raghuram, N. Sharon Hill, J. L. Gibbs, and L. M. Maruping. 2019. Virtual work: Bridging research clusters. *The Academy of Management Annals* 13, 1 (2019), 308–341.

[51] Giulia Regalia, Francesco Onorati, Matteo Lai, Chiara Caborni, and Rosalind W. Picard. 2019. Multimodal wrist-worn devices for seizure detection and advancing research: Focus on the Empatica wristbands. *Epilepsy Research* 153 (2019), 79–82.

[52] Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. 2018. Modern Code Review: A Case Study at Google. *Proceedings of 40th International Conference on Software Engineering.*

[53] Angela A. T. Schuurmans, Peter de Looff, Karin S. Nijhof, Catarina Rosada, Ron H. J. Scholte, Arne Popma, and Roy Otten. 2020. Validity of the Empatica E4 Wristband to Measure Heart Rate Variability (HRV) Parameters: a Comparison to Electrocardiography (ECG). *Journal of Medical Systems* 44, 190 (2020).

[54] Mert Sevil, Iman Hajizadeh, Sediqeh Samadi, Jianyuan Feng, Caterina Lazaro, Nicole Frantz, Xia Yu, Rachel Brandt, Zacharie Maloney, and Ali Cinar. 2017. Social and competition stress detection with wristband physiological signals. In *2017 IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. 39–42. https://doi.org/10.1109/BSN.2017.7936002

[55] N. L. Sin, R. P. Sloan, P. S. McKinley, and D. M. Almeida. 2016. Linking Daily Stress Processes and Laboratory-Based Heart Rate Variability in a National Sample of Midlife and Older Adults. *Psychosomatic medicine* 78, 5 (2016), 573–582.

[56] M J Stones and A Kozma. 1989. Age, exercise, and coding performance. *Psychol Aging* 4, 2 (1989).

[57] Hans Stuyck, Leonardo Dalla Costa, Axel Cleeremans, and Eva Van den Bussche. 2022. Validity of the Empatica E4 wristband to estimate resting-state heart rate variability in a lab-based context. *International Journal of Psychophysiology* 182 (2022), 105–118.

[58] M. Umair, N. Chalabianloo, C. Sas, and C. Ersoy. 2021. HRV and Stress: A Mixed-Methods Approach for Comparison of Wearable Heart Rate Sensors for Biofeedback. *IEEE Access* 9 (2021), 14005–14024.

[59] A Uusitalo, T Mets, K Martinmaki, S Mauno, U Kinnunen, and H Rusko. 2011. Heart rate variability related to effort at work. *Appl Ergon.* 42 (2011), 830–838.

[60] T. G. Vrijkotte, L. J. van Doornen, and E. J. de Geus. 2000. Effects of work stress on ambulatory blood pressure, heart rate, and heart rate variability. *Hypertension* 35, 4 (2000), 880–886.

[61] D. Watson, L. A. Clark, and A. Tellegen. 1988. Development and validation of brief measures of positive and negative affect: the PANAS scales. *Journal of personality and social psychology* 54, 6 (1988), 1063.

[62] Bobo Zhao, Zhu Wang, Zhiwen Yu, and Bin Guo. 2018. EmotionSense: Emotion Recognition Based on Wearable Wristband. In *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. 346–355. https://doi.org/10.1109/SmartWorld.2018.00091