T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL

EFFECT OF USER INTERRUPTION AND MITIGATION STRATEGIES ON WORK PERFORMANCE OF SOFTWARE DEVELOPERS

HATİCE DİŞLİ

A THESIS OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER ENGINEERING

ADVISOR: ASSOC. PROF. DR. MEHMET GÖKTÜRK

OCTOBER 2024

T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL

EFFECT OF USER INTERRUPTION AND MITIGATION STRATEGIES ON WORK PERFORMANCE OF SOFTWARE DEVELOPERS

HATİCE DİŞLİ

A THESIS OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER ENGINEERING

ADVISOR: ASSOC. PROF. DR. MEHMET GÖKTÜRK

OCTOBER 2024

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

KULLANICI KESMESİ VE AZALTMA STRATEJİLERİNİN YAZILIM GELİŞTİRİCİLERİN İŞ PERFORMANSI ÜZERİNDEKİ ETKİSİ

HATİCE DİŞLİ

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN: DOÇ. DR. MEHMET GÖKTÜRK

EKİM 2024

MASTER of SCIENCE JURY APPROVAL FORM

A thesis submitted by Hatice DİŞLİ, defended on 18/10/2024 before the jury formed with the 01/07/2024 date and 2024/33 numbered decision of the GTU Graduate Administration Board, has been accepted as a MASTER of SCIENCE thesis in the Department of Computer Engineering.

JURY

MEMBER (THESIS ADVISOR)	: Assoc. Prof. Dr. Mehmet GÖKTÜRK
MEMBER	: Prof. Dr. Hidayet TAKCI
MEMBER	: Assist. Prof. Dr. Burcu YILMAZ

APPROVAL

Gebze Technical University Graduate Administration Board

…..…/…..…/..…… date and ……/…….. numbered decision.

SIGNATURE/SEAL

ABSTRACT

In today's competitive software development environment, software developers need to concentrate diligently on their work to ensure their software products are delivered with few or no errors. User interruptions refer to interventions that distract or disrupt a user while they are engaged in a specific task. These interruptions are often computer-based and arise from various communication tools such as instant messaging, file sharing, and video calls. These tools play a significant role in both work and personal interactions, and their usage rates are constantly increasing. However, when these interactions are not properly coordinated, interruptions during tasks can lead to decreases in work performance and increased stress levels. Time management plays a critical role reducing the adverse impacts of these interruptions. Strategies such as effective time management, setting priorities, and task prioritization help maintain balance in professional as well as personal life.

To analyze how interruptions affect developers, a survey was conducted with 20 participants. The survey results showed that the most interruptions experienced by developers were computer-based and distracted them from their tasks. Following the survey, a two-week experimental study involving 12 participants was conducted using the Pomodoro technique to manage interruptions in the software world.

This study has been presented as both a pilot study and an exploratory experimental study, supported by literature indicating that a sample group of 12 individuals is sufficient for such research. Pilot studies aim to test the feasibility and fundamental effects of a new method or technique on a small sample, typically using small-scale groups. The small sample sizes used in pilot studies allow for initial impressions of the method's core functioning. Developers can easily establish rules tailored to themselves, such as muting computer notifications, prioritizing tasks, synchronizing different devices, and determining work and break times. The results of the study indicated a decrease in distracting interruptions, leading to a decrease in the error rate of software developers. Additionally, calculations showed that the p-value was 0.038, indicating that these findings are statistically significant.

According to this results, Pomodoro Technique enhances focus, leading to more efficient and less error-prone coding, which aids in the correct and effective implementation of algorithms and data structures. A more attentive and concentrated work environment allows software engineers to work on complex software architectures and design patterns with fewer mistakes. This increased focus is also beneficial for optimizing software performance, as it supports engineers in writing efficient code, detecting performance issues early, and resolving them effectively. A more focused and careful work routine helps reduce technical debt in software projects by enabling engineers to make fewer errors and write cleaner code, facilitating easier maintenance and development in the long term. Consequently, fewer errors lead to more effective testing processes and smoother deployment phases, ensuring the software is more reliable and robust before release.

Keywords: User Interruption, Computer Based Communication, Notification Management, Task Performance, Pomodoro Technique

ÖZET

Günümüzün rekabetçi yazılım geliştirme ortamında, bir yazılım geliştiricisinin, yazılım ürününü az hata ile veya hatasız teslim edebilmesi için işine büyük bir dikkat ve özenle yoğunlaşması gerekmektedir. Kullanıcı kesintileri, bir kullanıcının belirli bir görevle meşgulken dikkatini dağıtan veya iş akışını bozan müdahaleleri ifade eder. Bu kesintiler genellikle bilgisayar tabanlıdır ve anlık mesajlaşma, dosya paylaşımı, video aramaları gibi çeşitli iletişim araçlarından kaynaklanır. Bu araçlar, iş ve kişisel etkileşimlerde önemli rol oynar ve kullanım oranları sürekli artmaktadır. Ancak, bu etkileşimlerin doğru koordinasyonu sağlanmadığında, görev sırasında oluşan kesintiler iş performansında düşüşlere ve stres seviyelerinde artışa yol açabilir. Zaman yönetimi, bu kesintilerin olumsuz etkilerini azaltmak için kritik bir rol oynar. Zamanı etkili kullanma, öncelikleri belirleme ve görevleri sıralama gibi stratejiler, hem iş hem de kişisel yaşamda denge sağlamaya yardımcı olur.

Bu araştırmada, kesmelerin geliştiricilere etkilerini incelemek amacıyla 20 kişi ile bir anket çalışması yürütüldü. Anket sonuçları, geliştiricilerin yaşadığı kesmelerinin çoğunluğunun bilgisayar tabanlı olduğunu ve geliştiricileri görevlerinden uzaklaştırdığını gösterdi. Anket çalışmasından sonra, yazılım dünyasındaki kesmeleri yönetmek için Pomodoro tekniği kullanılarak 12 kişiyle 2 haftalık bir deney çalışması yapıldı. Bu çalışma, hem pilot çalışma hem de keşifsel deney çalışması olarak sunulmuştur, çünkü 12 kişilik bir örneklem grubunun bu tür araştırmalar için yeterli olduğu literatürle desteklenmektedir. Pilot çalışmalar, yeni bir yöntem veya tekniğin uygulanabilirliğini ve temel etkilerini küçük bir örneklemle test etmek amacı taşır ve bu bağlamda genellikle küçük ölçekli gruplar kullanılır. Pilot çalışmalarda kullanılan küçük örneklem büyüklükleri, yöntemin temel işleyişi hakkında ilk izlenimlerin elde edilmesini sağlar. Geliştirici, bilgisayar tabanlı bildirimleri sessize almak, işleri önceliklendirmek, farklı cihazları senkronize etmek, çalışma ve mola sürelerini belirlemek gibi kendine uygun kuralları kolayca oluşturabilir. Çalışmanın sonucunda, dikkat dağıtıcı kesmelerin sayısının azaldığı ve bu nedenle geliştiricinin hata yapma oranının azaldığı görüldü. Ayrıca, bu bulguların istatistiksel olarak anlamlı olduğunu göstermek için yapılan hesaplamalar sonucu p değerinin 0.038 olduğu bulundu.

Bu sonuçlara göre Pomodoro Tekniği, odaklanmayı artırarak daha verimli ve hata sayısı daha az kod yazmayı sağlar; bu da algoritmaların ve veri yapıların daha doğru ve etkili şekilde uygulanmasına yardımcı olur. Daha dikkatli ve odaklanmış bir çalışma ortamı, yazılım mühendislerinin karmaşık yazılım mimarileri ve tasarım desenleri üzerinde çalışırken daha az hata yapmasını sağlar. Sağladığı bu odaklanma, yazılım performansını optimize etmeye yönelik çalışmalarda da faydalı olabilir. Verimli kod yazma, performans sorunlarını erken tespit etme ve çözme gibi konularda mühendislerin daha etkin çalışmasını destekler. Daha odaklı ve dikkatli bir çalışma düzeni, yazılım projelerinde teknik borcun azaltılmasına yardımcı olarak mühendislerin daha az hata yapmasını ve daha temiz kod yazmasını sağlar; bu da uzun vadede projelerin bakımını ve geliştirmesini kolaylaştırır. Sonuç olarak, bu yöntemin kullanımı, daha az hata, daha etkili test süreçleri ve sorunsuz devreye alma süreçleri anlamına gelir ve yazılımın piyasaya sürülmeden önce daha güvenilir ve sağlam olmasını sağlar.

Anahtar Kelimeler: Kullanıcı Kesmesi, Bilgisayar Tabanlı İletişim, Bildirim Yönetimi, Görev Performansı, Pomodoro Teknik

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my esteemed advisor, Assoc. Prof. Dr. Mehmet Göktürk, for his guidance, motivation, suggestions, unwavering support, and assistance in the realization of this study, as well as for paving the way for this work with his knowledge. I extend my thanks to my esteemed professors who have always propelled us forward with their extensive knowledge and support, and who serve as role models with their lives.

I would like to express my gratitude to my husband, Harun Dişli, who has always been by my side during difficult times and has provided unwavering support throughout my master's journey.

Lastly, I offer my thanks to the Alkaya family for their patience, unconditional love, and support, always being by my side, showing me the right path when I was lost, and always believing in me.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

HCI	:	Human Computer Interaction
CNN	:	Convolutional Neural Network
CDF	:	Cumulative Distribution Function
PDF	:	Probability Density Function
GTD	:	Getting Things Done
Group A	:	Experimental group using the Pomodoro Application
Group B	:	Control group not using Pomodoro application
E_A	:	Number of errors in Group A
N_A	:	Total tasks in Group A
E_B	:	Number of errors in Group B
N_B	:	Total tasks in Group B
S_E	:	Standard error
e_A	:	Error rate of Group A
e_B	:	Error rate of Group B
z-score	:	Standard score
$\Phi_{(z)}$	:	Cumulative distribution function of the z-score
p-value	:	Probability value
z-distribution	:	Standard Normal Distribution
H0	:	Null hypothesis
H1	:	Alternative hypothesis

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Software developers often deal with complex problem-solving tasks that require intense concentration. Interruptions can significantly disrupt their cognitive flow and increase cognitive load, as developers frequently need to switch between tasks that demand different mental resources. The cognitive load for software developers is higher compared to other office workers because their tasks are typically non-routine, require problem-solving, and involve managing complex information. This creates a stark contrast with more routine office tasks, which generally do not demand the same level of deep focus or continuous problem-solving. Office workers might experience cognitive load from multitasking and interruptions, but their tasks are often more structured and repetitive, which can mitigate overall cognitive demands **[1][2]**.

In the context of software development, lack of proper coordination can lead to disruptions that increase error rates and stress levels. Managing these disruptions in the software development lifecycle remains a significant challenge. Effective management of developer-computer interactions has become critical as computers are now indispensable in the field of software development. Without it, developers face interrupted workflows, which can negatively affect productivity.

At this point, the concept of time management, frequently discussed in academic studies **[3][4]**, offers a useful guide. Time management is the process of using time efficiently to complete tasks, and it includes setting priorities, planning, and developing strategies to increase productivity. It also helps maintain a balance between work and personal life, reducing stress. Successful time management can allow more tasks to be completed in less time, thereby potentially creating more free time.

One popular time management technique among developers is the Pomodoro technique, which helps increase productivity by encouraging work in focused intervals followed by short breaks. This method aligns well with the demands of software development, helping developers manage their cognitive load more effectively by breaking the work process into manageable chunks. Through time management strategies like Pomodoro, developers can better handle the high cognitive demands of their non-routine tasks while mitigating the negative impact of interruptions.

2. PURPOSE OF THE STUDY

The contemporary work environment is rife with interruptions, largely due to the need for coworkers to communicate with one another. Research has shown that these interruptions can affect knowledge workers' task performance. Part of these communication disruptions stems from the inevitable byproducts of using technology to support these interactions. Examining the role of computer-based interruptions and their impact on knowledge workers' task performance has become unavoidable due to the widespread use of technology in modern workplaces **[5].**

It has been found that, when knowledge workers' primary task interrupted by peripheral tasks, users took longer to complete tasks, made more errors, and reported higher levels of annoyance and anxiety. There is a need to better understand the impact of interruptions on task performance **[6].** Consequently, it seems that with the widespread adoption of computer-mediated communication tools in the workplace, like instant messaging, there is a greater need for additional experimental case studies **[7].**

Thomas Davenport characterizes knowledge workers as individuals possessing high levels of expertise, education, or experience, with their primary role involving the creation, distribution, or application of knowledge **[8].**

Davenport also contends that by classifying software developers as knowledge workers, their development activities are inherently considered as processes of knowledge creation.

Due to this definition, in this study, the term "knowledge workers" is specifically confined to software developers.

Advancements in computer technology enable the development of systems that allow individuals to perform multiple tasks simultaneously. However, human cognitive abilities are not progressing at the same rate **[9]**. Consequently, cognitive limitations make individuals more prone to making mistakes and becoming vulnerable to delays when interrupted during a typical workday. Interruptions in work environments, especially in software development environments, where computer usage is essential, are predominantly computer-based.

Recognizing and addressing computer-based interruptions, such as WhatsApp, Instagram, email, etc., that significantly impact users in software development environments is crucial for enhancing individuals' work performance and reducing stress levels. The objective of this thesis is to examine how the apply Pomodoro technique affects the time management behavior of software developers in dealing with interruptions in software development environments and to discuss implications for the future development of tangible time management interfaces.

A deeper examination has been conducted on why and how interruptions during computer coding tasks differ from those in other office work environments. Additionally, other time management techniques in this field, besides the Pomodoro Technique, have been investigated. These techniques are presented along with explanations and comparative tables.

In the study, demographic characteristics of the participants, such as age, gender, and educational background, were surveyed to assess the impact of these factors on interruptions. Since interruptions are not limited to computer-based issues, and their effects on developers are not exclusive, this topic has been examined in detail. The contributions of the Pomodoro Technique have been concretely demonstrated, the experimental aspect of the study has been sufficiently addressed in terms of data analysis, limitations have been thoroughly examined and discussed. The adequacy of usage session durations has been evaluated, and the types of interruptions as well as whether individuals were exposed to the same interruptions have been precisely measured.

This thesis contributes to the literature by scientifically investigating the effectiveness of the Pomodoro Technique in managing interruptions within software development environments. The technique supports enhanced focus, leading to more efficient and less error-prone coding processes and facilitates the accurate and effective implementation of algorithms and data structures. Additionally, a more attentive and focused work environment helps software engineers handle complex software architectures and design patterns with fewer mistakes. This increased focus aids engineers in writing effective code, detecting performance issues early, and resolving them promptly, which is beneficial for optimizing software performance.

Consequently, by solidifying the effects of the Pomodoro Technique on work performance and stress management, this research contributes to the development of time management interfaces and promotes the adoption of more scientific and systematic approaches to time management. This effort enhances the effectiveness of software testing processes and ensures smoother deployment phases, ultimately contributing to the reliability and robustness of the software.

3. LITERATURE REVIEW

User interruption poses an increasingly significant challenge in human-computer interaction. Progress in smart and multitasking computer systems has undeniably led to a notable surge in user interruptions **[10]**.

3.1. Interruptions Occurring at Different Times

User attention is a limited resource, and people are easily overwhelmed by interruptions. Systems often fail to consider the impact of disrupting a user during a series of tasks. A study by Zijlstra et al. **[11]** assesses how interrupting a user at various points during task completion. The study show that timing of interruptions can influence the degree of this effect reading comprehension.

Timing of interruptions can also influence other task performances. Adamczyk et al. **[12]** shows that the timing of interruptions can influence the degree of this effect during task completion affects task performance, emotional state, and social perception.

Task models were constructed using event detection techniques, and these models were utilized to determine interruption times based on the user's estimated cognitive load.

In this study, the best and worst moments for interruptions are defined as follows:

Best Moments for Interruptions:

- Scheduled Breaks: Interruptions during planned breaks or low-demand phases are often less harmful.
- Low-Intensity Tasks: Tasks that require less cognitive effort are more tolerant of interruptions.

Worst Moments for Interruptions:

- High-Cognitive-Load Phases: Interruptions during complex or critical phases can severely impact performance.
- Concentration-Heavy Tasks: Tasks requiring deep focus are highly sensitive to interruptions.

For example, interrupting the user while he/she is in an intense thinking process may increase the possibility of forgetting the purpose of the task.

The results indicate that different interruption moments have varying effects on users' mood and favorable social perception. This study revealed users' limited attention and sensitivity to interruptions.

Martens et al. **[13]** emphasized the significance of the timing of interruptions in their study with university students. They found that the impact of notifications on task performance is highly dependent on when they occur. Frequent notifications reduced task completion time but increased error rates, while interruptions during reading tasks led to longer completion times. This highlights that notifications during low-demand or planned breaks are less disruptive, whereas those during high-cognitive-load phases or deep focus periods can severely impair performance. An overview of the research model is illustrated in the Figure.

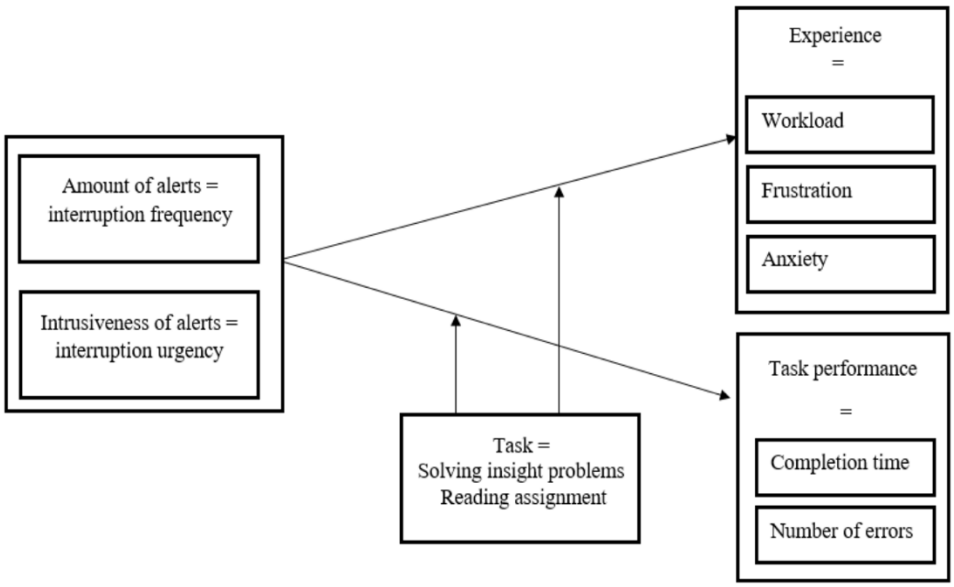

Figure 3.1: Research model overview **[13]**.

3.2. Source of the Interruptions

Grandhi et al. **[14]** argue that the origin of interruptions plays a significant role. A common form of attention distraction that is becoming more prevalent among

employees is brief disruptions from the external environment, referred to as external interruptions **[15]**.

External interruptions are a major concern in the daily work routines. Bellandi et al. **[16]** conducted a study to examine the effects of these interruptions and multitasking on surgical performance. The research, which involved monitoring the working habits of surgeons and nurses across seven hospitals in Italy, revealed that over 90% of participants reported experiencing interruptions. These disruptions mainly stem from external sources such as phone calls, messages, emergencies, and requests from other staff members.

The study found that these external interruptions significantly impact job performance. Surgeons and nurses reported that interruptions hinder their concentration, extend the duration of surgical procedures, and increase the likelihood of errors. To address these issues, the study proposes several strategies for managing interruptions more effectively:

- Surgeons and nurses should remain vigilant about interruptions and avoid unnecessary disruptions during work.
- The use of communication tools should be limited or restricted to reduce interruptions.
- Except in emergencies, interruptions caused by external factors should be scheduled at specific times.
- Designate one person to handle interruptions and assist other team members in completing their tasks.
- Prioritize tasks and plan for interruptions to occur only after high-priority work is finished.

These recommendations aim to help surgeons and nurses manage interruptions better and enhance their overall task performance.

Internal interruptions, or self-interruptions, are another significant factor in task performance.

Jin et al. **[17]** explored the effects of self-interruptions on productivity. A series of experiments were conducted to examine how self-disruption affects work productivity. The findings suggest that users' self-interruption actions only enhance work

performance in specific situations, while they may have negative effects in other cases. For instance, researchers found that users self-interrupt more frequently when faced with challenging tasks and increased task complexity, and these interruptions enhance task performance. However, when users encounter simpler tasks and task ease increases, they self-interrupt more frequently, leading to detrimental effects on work performance. Consequently, the researchers emphasized the need for users to manage their self-interruption actions according to different work situations and highlighted the necessity for further research to understand the effects of these actions on work performance.

In another study addressing the self-interruptions conducted by Dabbish et al. **[18],** it was noted that among the reasons for interruptions, besides environmental factors, individuals also interrupt themselves due to personal habits. For instance, when a person thinks while he/she is working o a spesific task, "How's the weather today, is it snowy?" they initiate a self-interruption. Such interruptions can be triggered not by external factors but by the individual's own thoughts or habits. This research highlights that interruptions affecting work performance are not solely caused by external factors but also by personal habits and self-interruption.

Abad et al. **[19]** argue that self-interruptions are more disruptive than external interruptions.

3.3. Computer-Based Interruptions

Interruptions in work environments, especially where computer usage is essential, are predominantly computer-based.

Recognizing and addressing computer-based interruptions, such as WhatsApp, Instagram, email, etc., that significantly impact users in work environments is crucial for enhancing or preserve individuals' work performance and reducing stress levels. These types of interruptions cause employees to become distracted from their current tasks. Due to the belief that interruptions caused by cell phone use in motor vehicles may lead to accidents, some countries have restricted the use of cell phones in motor vehicles, while others are considering such regulations.

A study was conducted by Redelmeier et al. **[20],** to examine the accuracy of this. The study examined more than 690 drivers involved in motor vehicle crashes that caused significant amount of property damage but did not result in personal injury.

Each driver's mobile phone communication history on the day of the accident and in the previous week was analyzed using detailed billing records. As a result of the study, it was observed that mobile phone interruptions in motor vehicles quadrupled the risk of collision, even if the call duration was short.

Notifications, usually presented in the form of visual or auditory cues, are an effective method to increase employees' access to information and awareness. However, these notifications often interrupt the workflow. Therefore, correct timing and effective management of notifications are of great importance; Otherwise, work performance may be negatively affected and focusing on tasks may become complicated.

In a study conducted by Iqbal et al. **[21]** investigating the impact of email notifications, some participants reported that turning off notifications encouraged them to check their email more often. However, other users stated that they were able to focus better on their tasks. Although most users agreed that notifications were distracting, they preferred to use notifications due to their value in maintaining awareness levels. A relevant chart is shown in **Figure 3.2**.

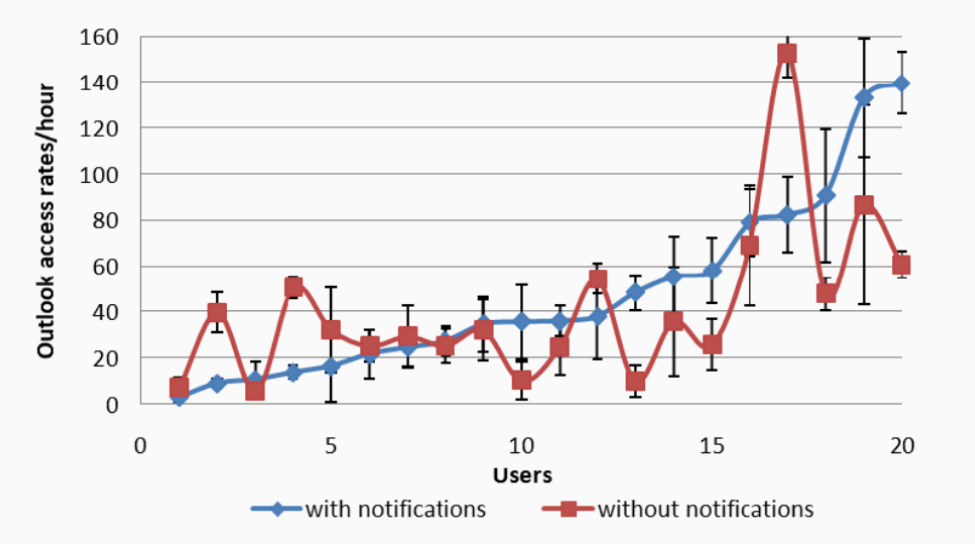

Figure 3.2: A time-normalized chart showing how email access rates change when notifications are disabled for all users. Lots of email access represents users whose job roles require them to respond to email **[21]**.

In another study conducted by Iqbal et al. **[22]** aimed to identify specific situations where talking while driving is harmful due to interruptions. Using a driving simulator, 18 participants were assessed under different driving difficulty levels and types of conversation.

The findings showed that simple paths with few or no other cars and constant speeds are safe for making and receiving phone calls. However, conversations that required recalling information from memory during driving led to driving-related issues. The results highlight the complexity of interactions between different types of conversation-focused interruptions and driving. They also emphasize the importance of understanding the nature of real world missions and their cognitive demands in realistic dual-task environments.

Managing interruption is a key factor to increase attention. A study conducted by Iqbal et al. **[23]** investigated the effectiveness of managing interruptions caused by cell phone use in motor vehicles. A driving simulator was used in the study. Methods such as putting voice calls on hold in critical road conditions were examined. These measures were found to reduce driving errors. Drivers have found this type of system valuable for maintaining driving safety in most situations. This result suggests that managing interruptions during phone calls can significantly help in keeping drivers' attention on the road. An example of the driver using the simulator is shown in **Figure 3.3.** The simulator is equipped with a dashboard containing a speed gauge, a steering wheel with indicators, and traditional brake and gas pedals.

Figure 3.3: The driver used a speakerphone to talk to the remote caller **[23]**.

Even though they may cause interruptions, another study emphasized the importance of using notifications conducted by Chang et al. **[24]** examined how mobile users manage interruptions through different ring modes.

The research found that users prefer to use different notifications to get important notifications noticed. While conducting this research, we focused on interactive communication applications such as phone calls, SMS messaging, messaging applications such as WhatsApp, Facebook Messenger, and video chat applications such as Skype. It was observed that without notification signals, users were less likely to respond immediately to mobile users' messages than when there was a signal, but despite hearing the notifications, response rates were not affected.

Another study related to notifications conducted by Akbar et al. **[25]** found that constant email notifications make it difficult for individuals to focus on their dual tasks, leading to increased stress levels. Using thermal imaging techniques, temperature changes in the facial regions of participants were observed during email notifications. Participants exposed to email interruptions showed significant increases in heart rate and skin temperature. Additionally, those frequently interrupted by emails took longer to complete their tasks. This study emphasized that email interruptions affect not only productivity but also stress levels. A relevant experimental setup is depicted in **Figure 3.4**.

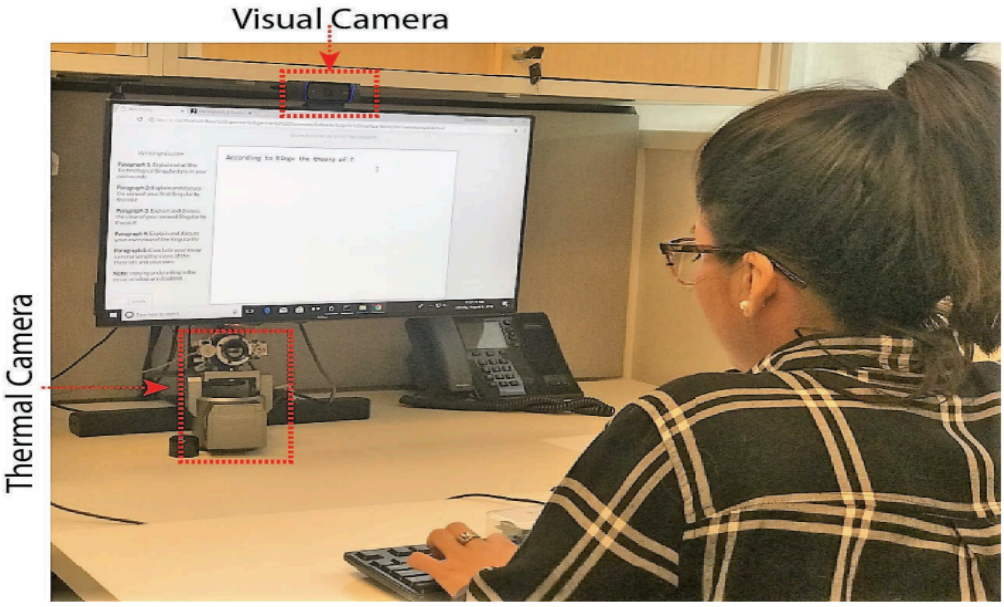

Figure 3.4: The snapshot shows a participant writing their essay **[25].**

The effect of interruptions is an issue that needs to be examined in order to offer solutions. Blank et al. **[26]** examines the impacts of email disruptions on work performance and emotional state. During the research, changes in employees' emotional states were observed following interruptions while checking their emails in an experimental study. It was observed that these changes affected their task performance. Particularly, as the frequency of checking emails increased, job stress and anxiety rose, while subjectively perceived productivity levels decreased. In the research, a convolutional neural network (CNN), one of the deep learning algorithms, was used to understand the emotional states of the participants. Relevant experiment shown in **Figure 3.5**.

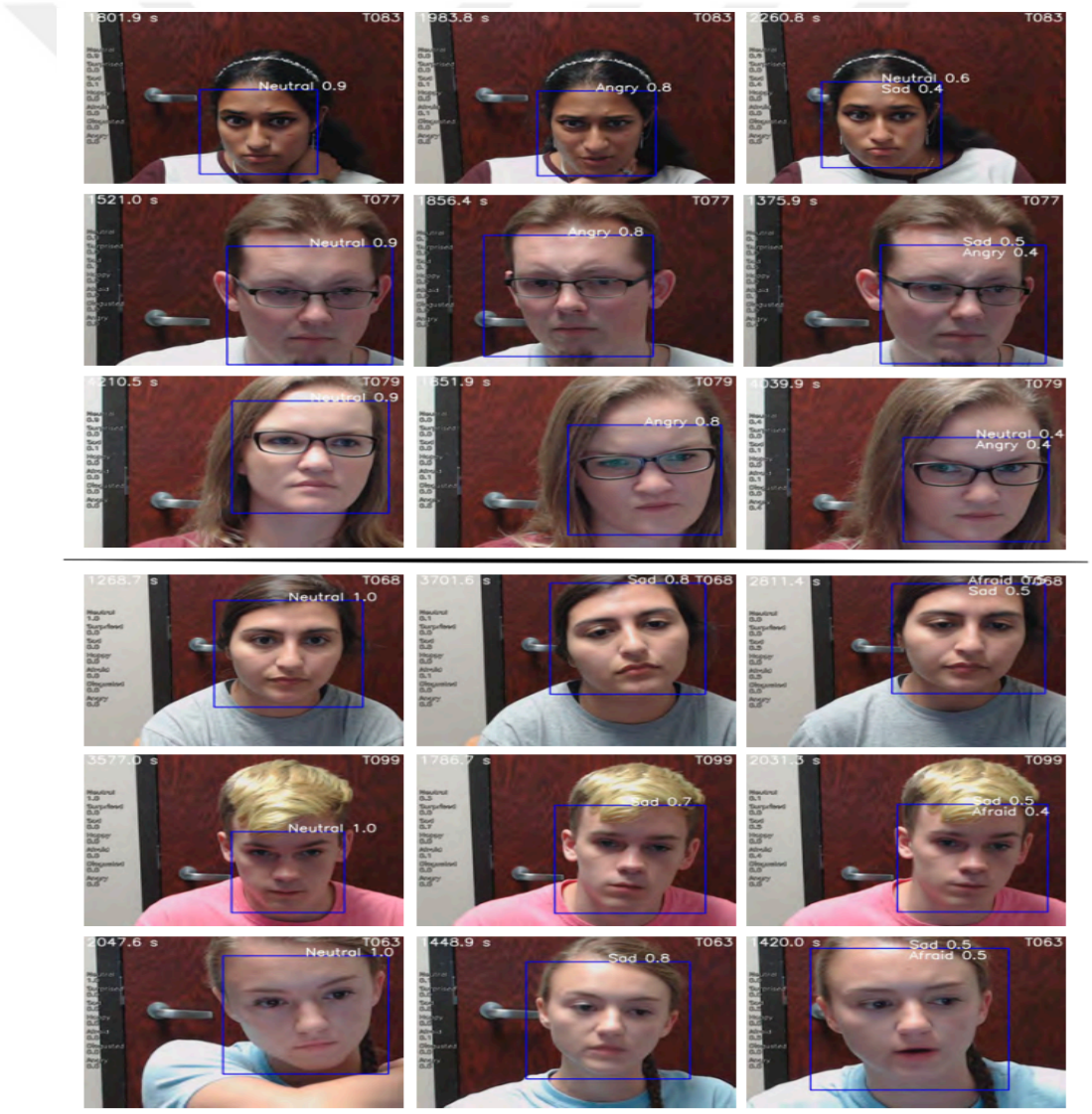

Figure 3.5: Images are annotated with the CNN output **[26]**.

There are various studies conducted to prevent digital interruptions. Tseng et al. **[27]** developed a system combining a browser extension and a chat bot. This system aims to reduce digital interruptions for employees during their transitions from breaktimes back to work by automatically blocking interrupting websites. The research indicates that this temporary blocking method can remarkably decrease digital interruptions and stress while maintaining employees' sense of control. The operating principle of the application is illustrated below **Figure 3.6**.

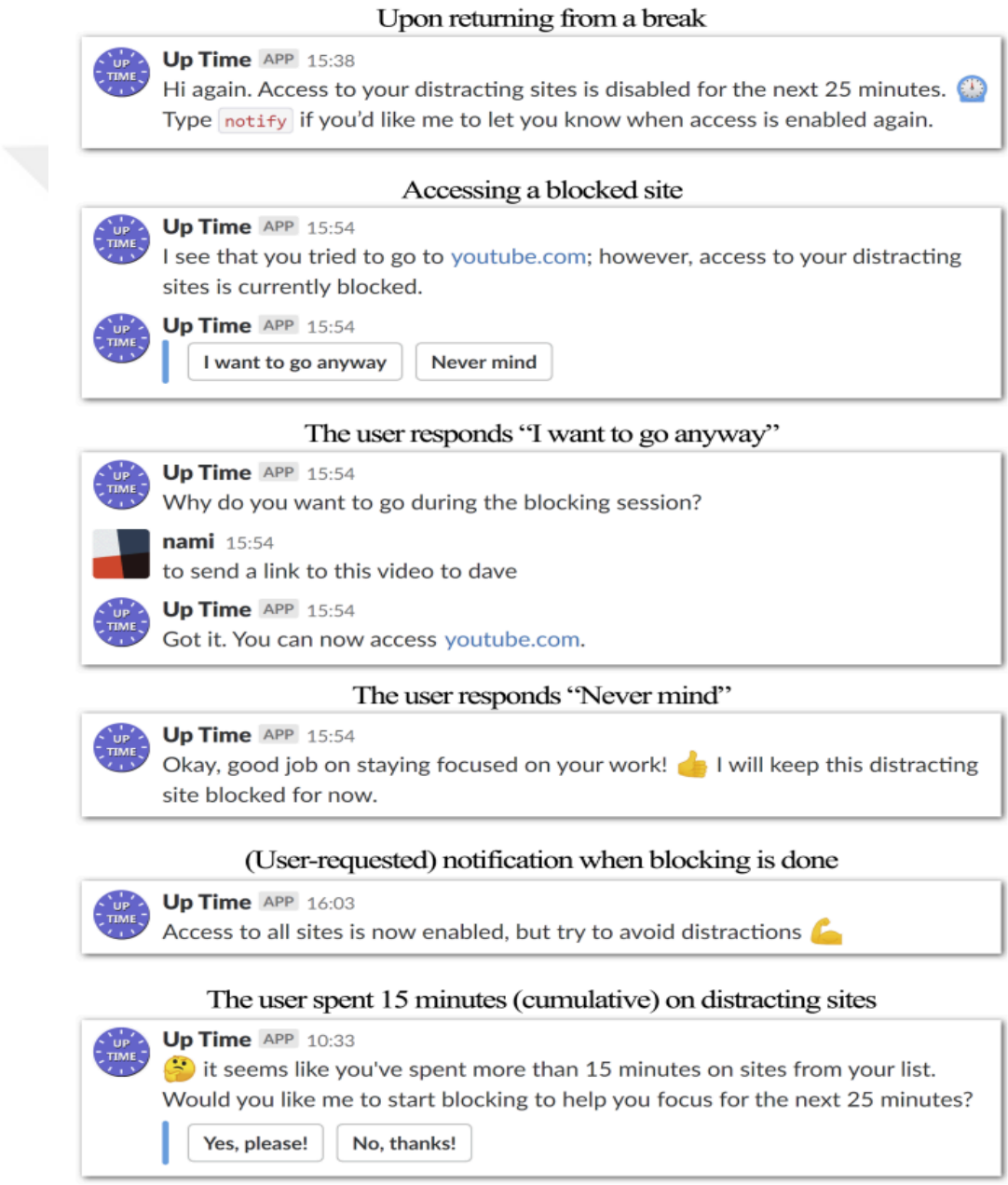

Figure 3.6: Working principle example of the application **[27]**.

Below, **Figure 3.7** shows part of the application used in the browser to select applications that interrupt the user.

Figure 3.7: Application selection screen **[27]**.

Time management is one of the beneficial process in dealing with interruptions. Time management is the process of effectively planning, prioritizing and managing tasks, activities and tasks in our lives. Time management skills are used to increase productivity, reduce stress, make it easier to achieve goals, and maintain balance. These skills allow you to use time more effectively, so we can focus more on important tasks and reduce unnecessary time wastage. Time management can be used to plan daily tasks and can also help us achieve long-term goals. It is also inevitable to benefit from time management processes in dealing with interruptions. The Pomodoro technique, one of the time management techniques, has been frequently used in academic studies. For example, Fauzan et al. **[28]** investigated the use of the Pomodoro technique to reduce the level of academic delay in students working on their thesis proposals. Findings showed that the Pomodoro technique has the ability to reduce academic delay in the process of completing the thesis proposal. In a research conducted by Kim et al. **[29]** on university students, an application called Pomodolock was developed based on the Pomodoro technique. The study focuses on the problem caused by the use of multiple devices such as computers, smartphones, tablets that are common in modern life. It examines how individuals interrupt themselves through non-work-related use of these devices, causing stress, and strategies for coping with these interruptions. Using these multiple devices for non-work purposes is seen as a

significant impediment to productivity. However, the habitual nature of this use makes non-work-related use difficult to manage.

A self-interruption management tool was developed to reduce these negative effects. Today, applications such as instant messaging and e-mail, which can be used on different device platforms such as computers and smartphones, have adopted the concept of multi-device management by making it possible to work synchronously between devices. Participants set a timer for a certain period of time, identified disruptive apps and websites, and then temporarily blocked them using the app. The results of the study showed that those who used the application had lower stress levels**.** Sample images of the application are shown in **Figure 3.8**.

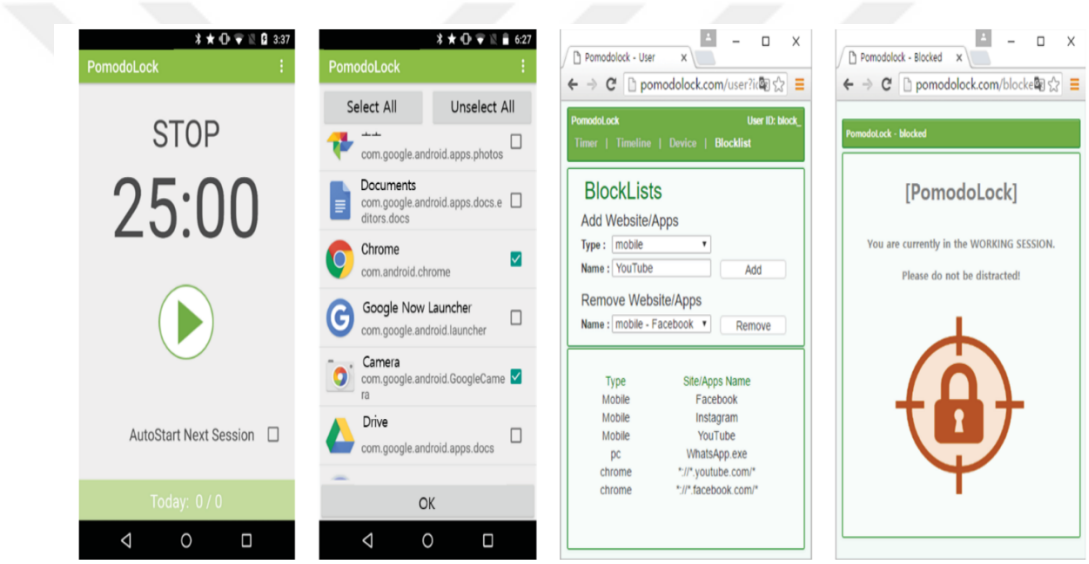

Figure 3.8: Application screen of PomodoLock **[29]**.

3.4. Interruptions in Software Development

A software developer's workday can be affected by numerous factors, including the tasks at hand, meetings, interruptions from colleagues, the technical setup, or the office environment **[30,31].** Certain factors lead to frequent activity and context changes, which can fragment work and negatively affect a developer's perceived productivity, task progress, and the quality of their work **[32]**.

Early research examining the overall way software developers allocate their time at work was conducted by Perry, Singer and colleagues, through two experiments, a diary

study, and observations, discovered that the majority of time was dedicated to coding, there was considerable spontaneous interaction with coworkers, and tasks typically took place in two-hour blocks **[31]**.

Singer and his team, employing a blend of questionnaires, interviews, observations, and tool usage metrics, determined that software developers mostly focus their time on finding information, studying documentation, and inspecting source code **[33]**. Subsequent practical studies have largely focused on distinct aspects of software development, such as developer teamwork and communication **[34],** common coding activities and tools **[35],** developers' requirements for information **[36],** and their understanding of software **[37].** For example, Goncalves and his team discovered through monitoring and conversations that developers spend 44 percent of their time collaborating, 33 percent searching for information, and only 10 percent using software management tools **[38].** More recent research by Minelli et al. **[39]** and Amann et al. **[40]** has examined how developers allocate their efforts within the integrated development environment.

In one study conducted by Meyer **[41],** researchers examined how programmers distribute their time at work and how this influences their views on productivity. To address this, a tracking tool was installed on 21 computers utilized by experienced software engineers from four different corporations, recording their activities over an average of 12 complete workdays in their natural work environment. When we look into the interruptions experienced by software developers, the term "flow" frequently comes up **[42].**

Flow is a psychological state first described by American psychologist Mihaly Csikszentmihalyi **[43].** It is characterized by a deep level of concentration where an individual becomes fully immersed in an activity, often losing track of time. Flow typically occurs during activities that are both enjoyable and motivating for the individual.

These flow experiences are often linked to tasks that require advanced skills and present increasing challenges. Key features of the flow state include complete focus on the assignment, a combination of action and consciousness, a feeling of mastery over one’s responsibilities, and the perception of task as intrinsically rewarding **[43,44].**

Since software development demands high levels of analytical problem-solving and creativity, it is expected that developers experience flow in their work. Flow is an essential component of "high-intensity tasks" like software development **[45].**

Interruptions and the need to switch tasks or contexts create a significant cognitive load, which can lead to reduced performance for software developers **[32].**

Such disruptions incorporate daily interruptions like "being diverted by other assignments; experiencing frustration or boredom with complicated or monotonous tasks; receiving urgent requests for changes from management; or even something as minor as a colleague's inquiry" **[45].**

The analytical nature of development work renders programmers especially susceptible to both external distractions and intentional self-disruptions, due to the limited capacity of human short-term memory. Studies indicate that programmers perceive themselves as most efficient when they can finish several or substantial tasks without interruptions or context shifts during their workday **[45,46,47].** Previous studies have highlighted that flow is an essential element of effective workdays for software engineers. Naturally, disruptions and context switching have been proven to reduce developers' efficiency, resulting in fewer flow states throughout the workday. **[41, 47, 48 49].**

In Ritonummi' study **[50],** a qualitative a questionnaire was carried out to comprehend the flow experiences of individuals working in software development and the factors that hinder these experiences. The research employed the Critical Incident Technique (CIT), a qualitative method introduced by Flanagan, to gather observations from participants about critical moments in their work. The survey aimed to gain insight into flow states in software development and the barriers that prevent these experiences.

The data collection process was carried out in two stages from December 2021 to April 2022. In the initial stage (pilot study), individuals working at major software development companies in Finland and globally were contacted via email, and in the second stage, data was gathered utilizing the Prolific online panel. Participants were asked to describe a specific flow experience they encountered in their work in detail and to provide information about the factors that influenced this experience. The participants held various job titles, including backend developer, frontend developer,

full-stack developer, game developer, IT manager, mobile developer, software engineer, software developer, and technical lead.

The participants' ages varied from 19 to 72, with an average age of 31. Most of them were male (75%) and possessed a achelor's degree (51%), whereas approximately a third held a master's degree.

The data, analyzed through the coding process, revealed that the factors hindering flow experiences could be categorized into three main groups: internal elements, perceived challenge-skill equilibrium, and external influences. Interruptions were identified as the most significant barrier to flow in software developers' work. The most frequently encountered flow barriers included interruptions (66%), tasks that were overly simple, tedious, or monotonous. (48%), and insufficient requirements (28%). Additionally, 43% of participants reported experiencing flow sometimes, while 30% reported experiencing it frequently. This study highlights the factors that obstruct flow in software development and the frequency with which these factors are encountered.

Another study conducted by Parnin et al. **[51]** investigated how interrupts affect performance during programming tasks performed by software developers. Interrupted developers attempted to make up for lost time before resuming their tasks.

Parnin and Rugaber discovered that merely one in ten programming tasks that are interrupted is resumed within a minute after the interruption.

The study identified three different strategies that developers use to reduce performance loss due to outages:

- Pick up where you left off: This strategy allows developers to pick up where they left off. This way, they can make up for lost time without having to redo the task.
- Task reconfiguration: This strategy allows developers to restructure the task instead of completing the task as it was before the interruption. In this way, less time is spent after downtime and performance loss is reduced.
- Completely restart the task: This strategy requires developers to completely restart the task. This is the strategy with the highest performance loss and is often used when interruptions take too long.

This study shows that disruptions have a significant effect on programming achievement, developers may need to use different strategies to restart tasks. These strategies can help mitigate the impact of interruptions.

In today's software development environment, developers must work intensively to complete software products within specific time frames. At this point, the concept of time management becomes important.

Even though various distractions are encountered during the work, developers are still required to deliver the software product within the specified time. In one study conducted by Ruensuk **[52]** explored methods to increase developers' focus and productivity. In this study, the Pomodoro technique was introduced to developers. As seen in this study results demonstrated in the **Figure 3.9 and Figure 3.10**, with the reduction of distractions, developers were able to complete their tasks in a shorter time.


[-405] Show line chart when mouseover ray bars Created: 13/Aug/14 Updated: 20/Oct/14 Resolved: 14/Aug/14

Status: Closed
Project:

Type: Improvement
Priority: Major
Reporter: Napassaporn Likhitsajjakul
Assignee: Mintra Ruensuk
Resolution: Fixed
Labels: None
Remaining Estimate: 0 minutes
Time Spent: 3 hours, 5 minutes
Original Estimate: Not Specified

Attachments: Country page.pdf
Story Points: 1
Developer: Mintra Ruensuk

Figure 3.9: Example of time spent before using the Pomodoro Technique **[52]**.


[PS-59] As a Scheduler, I would like to manage groups
Created: 29/Sep/14 Updated: 16/Dec/14 Resolved: 29/Sep/14

Status: Closed
Project:

Type: Task
Priority: Minor
Reporter: Mintra Ruensuk
Assignee: Mintra Ruensuk
Resolution: Fixed
Labels: None
Remaining Estimate: 0 minutes
Time Spent: 2 hours, 5 minutes
Original Estimate: Not Specified

Story Points: 1
Developer: Mintra Ruensuk

Figure 3.10: Example of time spent after using the Pomodoro Technique **[52]**.

As previously mentioned, Davenport **[8],** describes knowledge workers as software developers. Due to this definition, in the Vanessa et. al's study **[53],** the term "knowledge workers" is specifically confined to software developers.

Vanessa's research aimed to gather first-hand to comprehend the impacts of various notification types. A survey was designed to collect software developers' perceptions and experiences regarding digital interruptions. The survey included qualitative questions to explore how interruptions affect their flow states and how they cope with them, supplemented by some quantitative data. Google Forms was used as a free platform for the study. Participants received a concise summary of the study's scope. Personal identifying information was not collected, but demographic questions related to expertise in the software development sector were included.

A sum of 86 individuals responded to the survey. The participants have varied software development backgrounds: 14 individuals (16%) have less than one year of experience, while 21 (24%) have more than seven years of experience. The remaining participants have experience ranging from 1-3 years (26%) and 4-6 years (34%).

Most participants (80%) use Windows as their development platform. Linux (30%) and macOS (15.3%) are also commonly used platforms. The most frequent notifications received by programmers includes new email alerts (86%) and instant messaging apps (84%). Other types of alerts encompass system updates (76%), task modifications from productivity platforms like Jira (48%), non-work-related notifications (38.3%). The most commonly used platforms for communication are email and Microsoft Teams (see **Figure 3.11**).

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%

Figure 3.11: Most commonly used platforms for communications **[54].**

Participants were inquired if they utilize the "Quiet Mode" function About a third of the programmers show that they utilize this function occasionally (36%). Meanwhile, 34% of developers reported using it regularly, and 30% stated that they never use it. When asked in which situations this feature is activated, approximately a quarter of the respondents noted its use during meetings and screen sharing sessions (25.6%). The feature is also used when there is a sudden increase in notification frequency. Other scenarios mentioned include tasks such as deploying code to production, debugging, and code reviews. All of these tasks are interconnected, and errors can lead to significant consequences. These findings support observations that interruptions disrupt software developers' flow states, increasing stress and cognitive load **[54].** Such interruptions require additional resources and effort, negatively impacting developers' performance. The stacked bar chart (**Figure 3.12**) illustrates how programmers respond to digital disruptions while performins various key tasks.

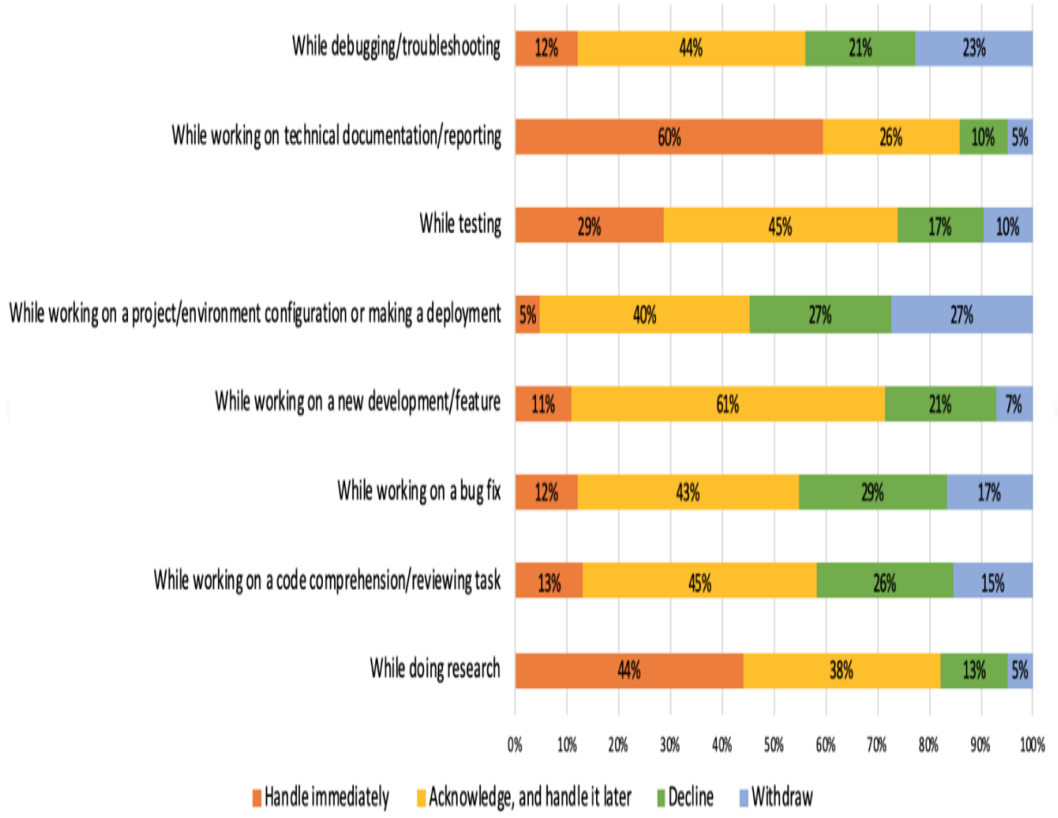

Figure 3.12: How programmers respond to digital distractions while performing key tasks **[54].**

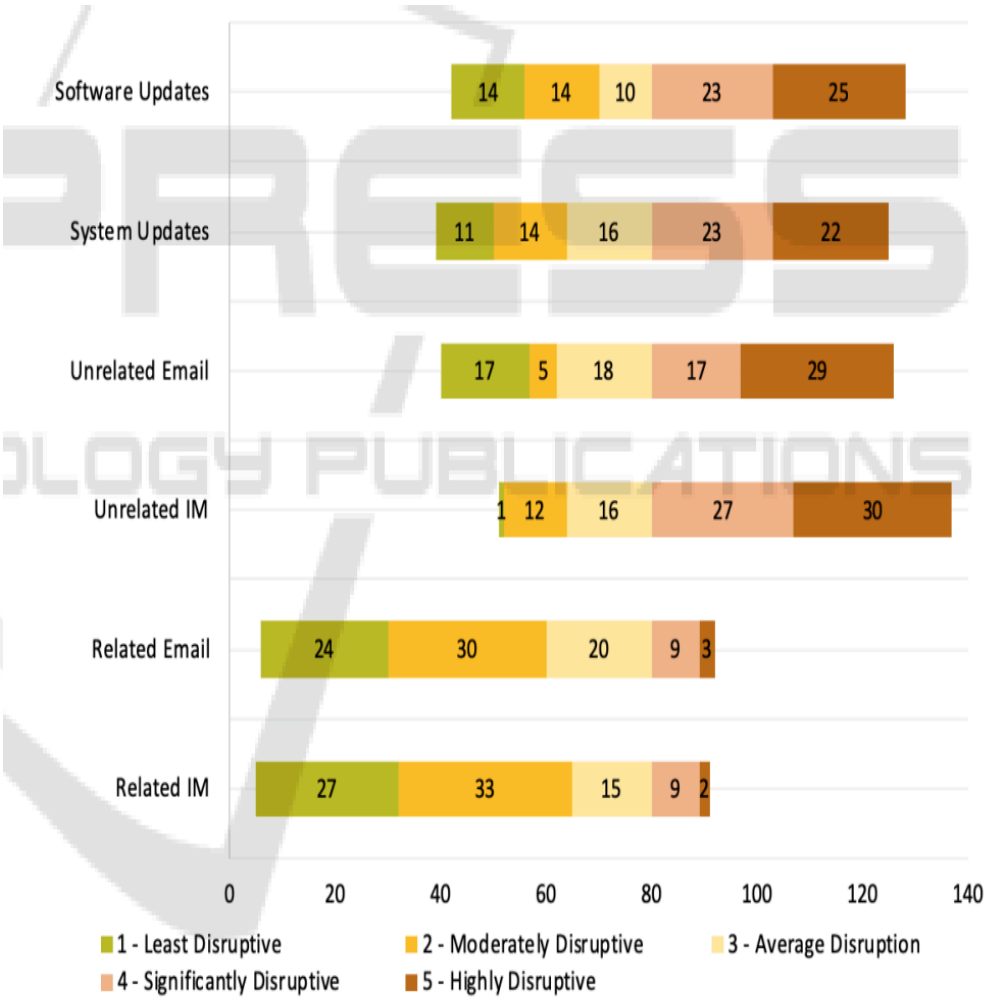

Figure 3.13: Bar chart displaying how software developers view different types of notifications, with 5 indicating the most disruptive and 1 the least **[54]**.

Digital disruptions in the work environment create a variety of challenges to work quality, stress levels, and cognitive demands. This study provides insights into how software developers perceive digital interruptions and when notifications are perceived as disruptive. Notifications that are not related to the main task are viewed as bothersome and have a greater effect on the programmers' state of flow. Therefore, emergency tactics such as documentation and code management are essential for helping programmers continue with their key tasks.

3.5. Mitigation Strategies

When discussing mitigation strategies for interruptions that affect software developers' work performance, the Pomodoro Technique is one of the most well-

known. However, several other methods have been explored in the literature, each addressing different aspects of productivity and work interruptions.

3.5.1 Time Blocking

Time blocking includes allocating designated time slots during the day for focused work. Developers dedicate blocks of time to particular tasks without interruptions. Research suggests that having pre-defined work periods can minimize interruptions and enhance focus, reducing recovery time after interruptions. Time blocking provides flexibility for handling interruptions by scheduling catch-up times **[55].**

Software Example: A developer is restructuring a microservices architecture. This can be a complex task requiring several hours of complete focus.

Interruption Example: The developer allocates specific time blocks (e.g., 09:00-12:00) solely for this task, protecting themselves from external interruptions such as meetings, emails, or messages. No other work is accepted during this time.

Advantage: This strategy ensures uninterrupted work on large projects, such as developing new features or system integration.

Disadvantage: However, if the project has a flexible schedule or requires sudden changes, time blocking may not be flexible enough and can be restrictive.

3.5.2 Task Switching and Resumption Lag Mitigation

Several studies **[56,57,58]** show that interruptions often result in a "resumption lag" — the time it takes to get back into a task after an interruption. Cognitive models have been developed to address this, suggesting strategies such as forced delays before task resumption to ensure higher accuracy. For instance, implementing an interface lockout after a break has been found to decrease error rates by preventing premature resumption of tasks.

Software Example: A software developer might be working on several projects simultaneously. For instance, they could be busy fixing bugs in a mobile application while also adding a new feature to a web application.

Interruption Example: When the developer receives a request from the project manager to review a new bug, they have to temporarily step away from their primary

task. When returning to their main task, a certain amount of time is needed to refocus. During this process, Resumption Lag, or the delay in resuming work, comes into play.

Advantage: Transition times between tasks after an interruption are reduced, and lost time is minimized through reminder notes and digital tools (e.g., Jira).

Disadvantage: However, a developer who frequently switches tasks may experience constant loss of focus, which can lead to an increase in error rates over time.

3.5.3 Mindfulness and Flow Induction

Mindfulness techniques encourage maintaining awareness of distractions and refocusing. These techniques help developers regain concentration more effectively after an interruption. Similarly, cultivating a state of "flow" — an intense focus on the task — has been seen as a method to increase resistance to distractions [56,59].

Software Example: A developer is engaged in deep algorithm optimization or performance improvement work and enters a state of flow. In this state, their attention is completely focused on one task and they are isolated from the outside world.

Interruption Example: An unexpected phone call or an urgent meeting request can disrupt this state of flow. The developer might use mindfulness techniques to regain focus and reduce stress after the interruption, such as practicing breathing exercises.

Advantage: Achieving a state of flow allows for deep focus on a task and complete isolation from external stimuli, leading to increased productivity and creativity.

Disadvantage: This state is very fragile; once disrupted, it can be quite difficult and time-consuming to return to the flow.

3.5.4 Attention-Aware Systems

Technology that dynamically manages attention, such as attention-aware systems, can reduce disruptions. These systems might pause notifications during critical work phases or delay non-essential communications until a break point. Studies indicate that integrating such systems can lower distraction rates and improve the quality of work **[56, 60].**

Software Example: A developer uses a system that provides attention awareness to track how much time is spent on various tasks throughout the day and analyzes which tasks are most frequently interrupted.

Interruption Example: The developer notices distracting factors through notifications from the system. For instance, if email notifications or social media alerts are causing interruptions, the developer stops these alerts and focuses more on their work. *Advantage:* These systems analyze the developer's work habits, showing which time periods experience the most interruptions and how to work more efficiently.

Disadvantage: However, the constant feedback from these systems can sometimes be distracting and create a form of "meta-interruption" while dealing with the feedback.

3.5.5 Pomodoro Technique

It is a technique that encourages working with intense focus at certain intervals (usually 25 minutes but adjustable per person) and short breaks (5 minutes) **[61].** It aims to increase efficiency, especially in short-term jobs or jobs that experience constant interruptions. The effectiveness of the method comes from its resistance to interruptions and its ability to optimize focus time.

Software Example: A software developer might be working on small and repetitive tasks, such as writing unit tests or cleaning up code. These tasks require short bursts of focus and can yield quick results.

Interruption Example: The developer takes a short break after every 25 minutes of work. If an email or instant messaging notification arrives, they respond to it after a Pomodoro cycle, thereby preventing the disruption of their work.

Advantage: This technique enhances productivity on small tasks and minimizes distractions by taking short breaks.

Disadvantage: However, for tasks requiring long periods of focus, such as in-depth optimization of an algorithm, taking a break every 25 minutes may disrupt the flow of work.

Apps based on the pomodoro technique usually have these features:

- It has a simple and user-friendly interface.
- It works in the background as well.

- The work and break durations can be adjusted.
- It allows synchronization across different devices (computer, phone, tablet).
- It enables restricting application notifications.
- Daily, weekly, and monthly reports can be generated for productivity tracking.
- Notification sound and vibration features are available.

3.5.6 Comparison of Pomodoro Technique with Other Strategies

The Pomodoro technique stands out compared to other strategies like Time Blocking, Task Switching, Timeboxing, and Getting Things Done (GTD) due to its flexibility and personalization features. For example, while Time Blocking uses fixed time intervals that cannot be adjusted during the work process, the Pomodoro technique offers adjustable work and break durations. This is especially useful for software developers, who may need different time intervals based on the complexity of tasks like debugging or coding. This flexibility aligns with findings by Yadav et al. **[62],** who noted that flexible time management strategies tend to be more effective in dynamic work environments.

Pomodoro also excels in minimizing distractions compared to Task Switching by allowing background operation and notification restrictions. Task Switching often leads to interruptions, as developers switch between tasks and applications, potentially reducing focus. Pomodoro's ability to limit notifications directly addresses the issue of attention fragmentation, which Bailey and Konstan **[6]** showed negatively impacts developers' productivity.

Additionally, the synchronization feature of Pomodoro, which allows seamless use across devices (like phones and tablets), offers more mobility compared to Timeboxing, where synchronization across devices is generally lacking.

This makes Pomodoro particularly useful for developers who need to transition between different work environments. Mark et al. **[18]** found that flexibility in workspaces improves software development performance, making this feature highly valuable.

Lastly, Pomodoro's built-in productivity tracking through reports (daily, weekly, monthly) offers a quantitative measure of progress, unlike Getting Things Done, which focuses more on task management without providing performance data.

This data-driven approach helps developers identify areas for improvement, aligning with research by Linderbaum and Levy **[63],** which found that continuous productivity tracking enhances task optimization and performance.

3.6. Literature Evaluation

The ultimate goal of studies in the field of user interruption is to understand their effects on individuals' work performance and emotional states, and to develop coping strategies for interruptions. In line with this objective, initial research in the domain of human-computer interaction has focused on examining the effects of disruptions at different times.

Subsequently, attention has been directed towards the impact of interruption types, and the influence of task variations on interruptions has been investigated. In addition to experiments conducted in laboratory settings, studies conducted in real office environments have been included to observe the real-life impact of these interruptions. Particularly, significant emphasis has been placed on computer-based interruptions, influenced by the digital age, and these interruptions have been evaluated alongside time management tools.

The inclusion of time management tools in academic studies has led to curiosity about their applicability in professional settings.

Consequently, the literature examines the use of time management techniques in coping with interruptions in the workplace and interprets their effects.

4. METHOD

4.1. Pomodoro Technique

The Pomodoro technique is one of the methods used to increase time management and productivity, developed by Francesco Cirillo in the 1980s **[61]**. This technique involves working in cycles with specific intervals between work and rest periods. Since Cirillo used a tomato-shaped kitchen timer when developing this technique, it was named 'Pomodoro', which means 'tomato' in Italian. Below, example of pomodoro timer shown in **Figure 4.1**.

Figure 4.1: Example of Pomodoro timer **[61].**

The Pomodoro technique usually includes these steps **[64]**:

- A task is determined.
- A 25-minute study period is started, this period is called a "Pomodoro".
- Work on the task with uninterrupted focus for 25 minutes.
- When the Pomodoro period ends, take a short break (usually 5 minutes).
- After every four Pomodoros, take a longer break (usually 15-30 minutes).
- Completing 8-10 Pomodoros per day is considered ideal.

4.2. Tools

4.2.1 Pomodoro Application

First, A Pomodoro application that suited our needs was initially chosen. The selected Pomodoro app is named Focus To-Do: Pomodoro. The following features are offered by it:

- It has a simple and user-friendly interface.
- It works in the background as well.
- The work and break durations can be adjusted.
- It allows synchronization across different devices (computer, phone, tablet).
- It enables restricting application notifications.
- Optional background sound selection is available.
- Daily, weekly, and monthly reports can be generated for productivity tracking.
- Notification sound and vibration features are available.

4.2.1 Jira Application

Jira application **[65]**, which provides task tracking, project management, calendar integration, reporting, and workflow management. In Jira, different priority levels are used to determine the urgency and importance of each issue. These levels indicate how quickly and crucially the issue needs to be resolved. The common priority levels in Jira and their descriptions are as follows **[66]**:

Highest: This level is used for critical errors or problems. Such issues can stop or significantly hinder the project's or application's operation.

Decision Criteria:

- Errors that cause the system to crash or become unusable in a production environment.
- Security vulnerabilities or severe data breaches.
- Urgent customer needs or situations requiring legal compliance.

Major: This level is for significant issues that are not as urgent as the "highest" level but can seriously affect workflows or operations.

Decision Criteria:

- Major functions not working correctly or malfunctioning.
- Significant performance issues or problems that slow down workflows.
- Major customer complaints or dissatisfaction.

Medium: The medium priority level is for issues that affect system operation but can typically be worked around temporarily.

Decision Criteria:

- Partial failure of specific features or functions.
- Errors that negatively impact user experience but do not completely halt work.
- Moderate performance issues.

Minor: The minor level is for small bugs and issues that are annoying but do not prevent work from being completed.

Decision Criteria:

- Issues classified under the minor level often include minor user interface errors, such as small misalignments, color mismatches, or typographical errors that do not interfere with functionality but may affect the visual appeal or polish of the product.
- Small functional problems that are unlikely to disrupt the user's primary workflow or objectives. These might include rare bugs that occur only under specific conditions and do not affect the system's overall stability or usability.
- Problems that have minimal impact on the user experience, meaning they do not prevent the user from completing tasks or meeting objectives, nor do they significantly reduce the efficiency of the workflow. Such issues may be mildly irritating but do not hinder the core functionality of the product.

Determining these priority levels typically involves a combination of team meetings, customer feedback, test results, and an assessment of overall business goals. Assigning the correct priority to each issue is essential, as it guides the team in deciding which problems to address first and how to allocate resources most effectively. This process not only ensures that team members work in alignment but also enables a focused approach toward solutions that enhance customer satisfaction, ultimately playing a key role in achieving broader business objectives.

5. EXPERIMENTS

5.1. Survey

This section will provide details regarding the methods implemented within the scope of the thesis. Firstly, a survey was conducted with 20 individuals selected from colleagues in different job positions in the workplace (front-end developer, back-end developer, mobile app developer, user experience/interface engineer, tester). This study aimed to identify the types and durations of disruptions frequently encountered in the workplace and to investigate their impact on task performance.

The following questions were asked to the participants:

1. Please indicate how many interruptions you experienced during work.
2. What types of interruptions do you mostly experience?
3. Which types of interruptions (E-mail, WhatsApp, Instagram, meetings, calls, phone etc.) do you think have the most impact on your work performance?
4. When an interruption occurs, how long do you think you are affected by it?
5. How much do you think interruptions distract you from your other tasks at the workplace?

5.1.2 Survey Results

Most participants experienced interruptions several times during the study period. As seen in the **Figure 5.1**, the most common type of interruption is caused by digital communication mediator such as Instagram, WhatsApp and E-mail.

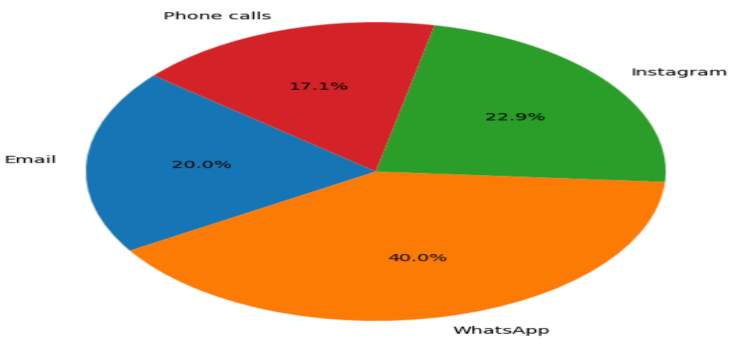

Figure 5.1: This is the most common types of interruptions.

Table 5.1: Survey results.

Participant	Q1 answer	Q2 answer	Q3 answer	Q4 answer	Q5 answer
1	3	Whatsapp	Email	15 minutes	15 minutes
2	5	Instagram	Phone calls	20 minutes	10 minutes
3	2	Email	Whatsapp	10 minutes	15 minutes
4	4	Whatsapp	Instagram	18 minutes	10 minutes
5	1	Email	Email	5 minutes	10 minutes
6	3	Instagram	Whatsapp	12 minutes	10 minutes
7	6	Whatsapp	Phone calls	25 minutes	12 minutes
8	2	Instagram	Whatsapp	8 minutes	15 minutes
9	3	Email	Email	13 minutes	20 minutes
10	4	Whatsapp	Instagram	17 minutes	15 minutes
11	2	Instagram	Whatsapp	9 minutes	15 minutes
12	5	Whatsapp	Phone calls	22 minutes	10 minutes
13	1	Email	Email	6 minutes	10 minutes
14	3	Instagram	Whatsapp	11 minutes	15 minutes
15	4	Whatsapp	Instagram	16 minutes	10 minutes
16	2	Instagram	Email	10 minutes	5 minutes
17	6	Whatsapp	Phone calls	24 minutes	10 minutes
18	3	Email	Whatsapp	14 minutes	7 minutes
19	2	Instagram	Email	9 minutes	6 minutes
20	4	Whatsapp	Phone calls	19 minutes	9 minutes

The survey results are as shown in **Table 5.1** above.

The survey results have shown that in the workplace, interruptions are mostly caused by computer and mobile device notifications such as WhatsApp, Instagram, E-mail. It has been observed that after interruptions, there is an increase in the time it takes to return to works, leading users to experience a reduction in productivity and resulting in stress. This situation has made it inevitable to conduct a study on managing interruptions.

5.2. Pomodoro Experiment

Recently, the Pomodoro technique, which is frequently used in academic studies to manage interruptions, has shown promising results, sparking interest in examining its effects on interruptions in the workplace specifically in software development environments **[28][67]**. Additionally, some research indicates that prolonged work sessions lead to a decrease in dopamine release, thereby reducing motivation and productivity **[68]**. Therefore, the effects of the intermittent working method, which forms the basis of the Pomodoro Technique, have been investigated.

To achieve this goal, we conducted an experiment over a two-week period with a group of 12 colleagues selected from various job positions such as front-end developer, back-end developer, mobile application developer, user experience/interface engineer. The experiment aimed to observe the impact of the Pomodoro technique on software developers due to interruptions caused by computer-based notifications.

Then, Participants were divided into two groups of six: an experimental group (Group A) and a control group (Group B). Both groups were assigned through the Jira application, 40 tasks of equal difficulty were assigned, 10 of which were major and 30 were medium, as shown in the **Table 5.2** and **Table 5.3**.

Before beginning their tasks, Group A was instructed to download and actively use a designated Pomodoro application. This specific application was chosen to assist them in managing their work intervals and breaks in a systematic and structured way. Participants in Group A were required to strictly adhere to the Pomodoro technique, working in focused time blocks followed by short, scheduled breaks to enhance productivity and maintain focus. Group B, in contrast, started their tasks under their usual working conditions without the aid of the Pomodoro application. They were

permitted to manage their work periods and breaks in their regular, unstructured manner, without any formal guidance or intervention regarding time management.

Table 5.2: This is the beginning of Sprint of Group A.

Priority	Closed	Testing	In Progress	Assignee	Total
Major	0	0	0	10	10
Medium	0	0	0	30	30
Total Unique Issues	0	0	0	40	40

Table 5.3: This is the beginning of Sprint of Group B.

Priority	Closed	Testing	In Progress	Assignee	Total
Major	0	0	0	10	10
Medium	0	0	0	30	30
Total Unique Issues	0	0	0	40	40

Participants in Group A, after downloading the Pomodoro application, activated its multi-device synchronization feature, allowing them to block interruptions from notifications across all their devices. This feature was intended to minimize distractions and help them maintain focus during work sessions. Additionally, Group A participants made use of the application's reporting feature, which enabled them to track and measure their productivity over time, providing valuable insights into their work patterns and efficiency.

Within Group A, there were varying preferences regarding the work environment; some participants opted to play background music while they worked to create a more motivating atmosphere, while others chose to work in complete silence for maximum concentration. In contrast, participants in Group B, who did not use the Pomodoro application and decided to keep notifications active, faced occasional interruptions during their tasks, which potentially disrupted their focus and workflow.

6. RESULT

As a result of the study, as shown in the **Table 5.4** and **Table 5.5**, it was observed that Group A failed to complete 7 out of 40 tasks, whereas Group B failed to complete 9 out of 40 tasks.

Table 5.4: This is end of the Sprint of Group A.

Priority	Closed	Testing	In Progress	Assignee	Total
Major	8	0	2	0	10
Medium	25	0	5	0	30
Total Unique Issues	33	0	7	0	40

Table 5.5: This is end of the Sprint of Group B.

Priority	Closed	Testing	In Progress	Assignee	Total
Major	7	0	3	0	10
Medium	24	0	6	0	30
Total Unique Issues	31	0	9	0	40

According to this result, it was found that using the Pomodoro technique does not lead to a significant change in the number of tasks completed or the speed of performing the given tasks. However, as an interesting finding, when comparing the errors reported after the 2-week study, it was observed that 11 error records were opened related to the tasks performed by Group A, while 17 error records were opened for Group B. These results shown in **Table 5.6** and **Table 5.7**.

Table 5.6: Number of Bugs opened at the end of the Sprint for Group A.

Priority	Closed	In Progress	Assignee	Total
Highest	0	0	0	0
Major	0	0	3	3
Medium	0	0	5	5
Minor	0	0	3	3
Total Unique Issues	0	0	11	11

Table 5.7: Number of Bugs opened at the end of the Sprint for Group B.

Priority	Closed	In Progress	Assignee	Total
Highest	0	0	3	3
Major	0	0	4	4
Medium	0	0	6	6
Minor	0	0	4	4
Total Unique Issues	0	0	17	17

These results indicate that the use of the Pomodoro technique in Group A leads to an increase in attention levels, a decrease in error rates, and consequently, an improvement in overall software quality. For Group B, however, it was observed that interruptions caused by notifications led to a decline in attention levels and an increase in the number of errors. Furthermore, when comparing the severity levels of the reported errors, it was found that Group B, which did not use the application, made errors of higher criticality, with a greater total number of errors overall, as shown in **Table 5.6** and **Table 5.7**.

6.1 Correlation Findings

Participants:

- Age Range: 22-39
- Average Age: 25.83
- 58% of the participants are women and 42% are men. Those who are 30 years old and above are 2 women and 2 men.
- Age Groups:
    - Under 30: Present in both the Pomodoro-using and non-Pomodoro-using groups.
    - 30 and Over: Present in both the Pomodoro-using and non-Pomodoro-using groups.
- Job Positions: Mobile, Front-end, and Back-end developers are equally distributed.
- Education Level: All participants are graduates with a bachelor's degree in Computer Engineering.
- Types of Notifications Received: Instant messaging chats (WhatsApp), email alerts, system updates, task updates from productivity systems like Jira (via email), phone calls, and non-work-related notifications (Instagram).
- Experience Duration: Mobile developers have 3-4 years of experience, backend developers have 5-15 years, and frontend developers have 4-15 years of experience.

Performance Data:

- Non-Pomodoro Group:
    - Under 30:
        - Highest Severity Errors: 3
        - Major Errors: 4
        - Medium Severity Errors: 4 (out of 6 total)
        - Total Errors: 3 (highest) + 4 (major) + 4 (medium) = 11 errors

    - 30 and Over:
        - Highest Severity Errors: 0
        - Major Errors: 0
        - Medium Severity Errors: 2 (out of 6 total)
        - Minor Errors: 4
        - Total Errors: 0 (highest) + 0 (major) + 2 (medium) + 4 (minor) = 6 errors
- Pomodoro Group:
    - Under 30:
        - Highest Severity Errors: 0
        - Major Errors: 3
        - Medium Severity Errors: 4 (out of 5 total)
        - Minor Errors: 2 (out of 4 total)
        - Total Errors: 0 (highest) + 3 (major) + 4 (medium) + 2 (minor) = 9 errors
    - 30 and Over:
        - Highest Severity Errors: 0
        - Major Errors: 0
        - Medium Severity Errors: 1 (out of 5 total)
        - Minor Errors: 1 (out of 4 total)
        - Total Errors: 0 (highest) + 0 (major) + 1 (medium) + 1 (minor) = 2 errors

People under 30 were responsible for the highest severity errors. The Pomodoro group (people under 30) demonstrated a significant advantage, particularly in reducing these errors. While the Non-Pomodoro group (people under 30) had 3 highest severity errors, the Pomodoro group had 0. This shows that the Pomodoro technique plays a crucial role in reducing the highest severity errors among software developers under 30, and it contributes to improving overall performance.

The Pomodoro technique has had a more pronounced effect on individuals aged 30 and over; the Pomodoro-using group made only 2 errors in total, while the non-Pomodoro group (individuals aged 30 and over) made 6 errors.

This demonstrates the Pomodoro technique's significant impact on error reduction, especially for critical and minor errors, and its overall effectiveness in improving performance across different age groups.

No significant performance differences were observed between different job positions, suggesting that the Pomodoro technique can enhance performance and reduce errors regardless of job role.

There is also no significant performance differences were observed between different genders.

6.2 Performance Criteria

The statistical significance of our results was measured to determine whether the findings are reliable. The concept of statistical significance implies that a statistical test result is not due to random chance but indicates a real effect or relationship within the data set. Statistical significance is typically evaluated using a probability value (p-value) or a confidence interval associated with a specific confidence level (alpha level). The alpha level is usually set to values such as 5% or 1%. In numerical research, data is examined using null hypothesis significance testing, commonly referred to as hypothesis testing **[69]**.

This is an organized approach for determining if a correlation between variables or a distinction between groups is statistically significant.

The null hypothesis (H0) suggests that there is no significant impact, no correlation among variables, or no difference among groups. In contrast, the substitute hypothesis (Ha or H1) proposes that there is a significant effect, an association among variables, or a disparity between groups. Hypothesis testing starts with the premise that the null hypothesis is valid. This approach allows you to assess the probability of your results occurring under this premise. Depending on the test's outcome, you can either reject or accept the hypothesis. The statistical significance of a result is related to the ability to reject the hypothesis. The hypothesis typically represents a default assumption,

suggesting that there is no effect or connection being tested. When conducting a statistical test, the resulting p-value provides a criterion for deciding whether to reject the null hypothesis. If the obtained p-value is smaller than the alpha level 5%, this shows that the hypothesis can be rejected, and the alternative hypothesis can be acknowledged. In this instance, the result is statistically significant and not due to chance. Statistical significance provides strong evidence for the presence of a real impact or connection in the data set **[69]**. However, statistical significance does not imply that the result is practically significant. Therefore, it is essential to take into account the practical significance of the results, additionally their statistical significance. The formulas for calculating the p-value are shown below.

First, the error rates for Group A are calculated in equations (6.1).

$$e_A = \frac{E_A}{N_A} \tag{6.1}$$

Then, the error rates for Group B are calculated in equations (6.2).

$$e_B = \frac{E_B}{N_B} \tag{6.2}$$

Where:

E_A: Number of errors in Group A

N_A: Total tasks in Group A

E_B: Number of errors in Group B

N_B: Total tasks in Group B

Then, the standard error is calculated in equations (6.3).

$$SE = \sqrt{\left(\frac{e_A(1-e_A)}{N_A}\right) + \left(\frac{e_B(1-e_B)}{N_B}\right)} \tag{6.3}$$

Where:

e_A: Error rate of Group A

e_B: Error rate of Group B

N_A: Total tasks in Group A

N_B: Total tasks in Group B

Then, the z-score is calculated in equations (6.4).

$$z = \frac{e_B - e_A}{SE} \tag{6.4}$$

Where:

e_A: Error rate for Group A

e_B: Error rate for Group B

S_E: Standard error

Finally, the p-value calculated in equations (6.5).

$$p = 1 - \Phi(z) \tag{6.5}$$

Where:

$\Phi_{(z)}$: Cumulative distribution function (CDF) of the z-score.

Table 6.1: Table of p-value interpretation **[70]**.

P-value	**Explanation**
$0.01 \leq p < 0.05$	Statistical significance
$0.001 \leq p < 0.01$	High level of statistical significance
$p < 0.01$	Very high statistical significance
$0.05 \leq p < 0.10$	Trend towards significance (borderline significance)
$p > 0.10$	The difference was due to chance (statistically significant difference not detected)

The p-value interpretation is shown in the **Table 6.1** above. The first hypothesis was tried to be verified using the formulas (6.1), (6.2), (6.3), (6.4) and (6.5).

First Hypothesis is (for total error count):

Null Hypothesis (H0): There is no difference in the number of errors made by those who use the Pomodoro technique and those who do not use it.

Alternative Hypothesis (H1): The number of errors made by those who do not use the Pomodoro technique is greater than the number of errors made by those who use it.

The code in **Figure 6.1** below was used to calculate the p-value to measure the accuracy of this first hypothesis.

```python
import scipy.stats as stats
import math
from scipy.optimize import minimize_scalar

# Number of tasks completed (Group A)
tasks_A = 33
# Number of tasks completed (Group A)
tasks_B = 31

# Total error count for Pomodoro users (Group A)
errors_A = 11
# Total error count for non-Pomodoro users (Group B)
errors_B = 17

# Calculate error counts and p-value
errors_B = round(errors_B)
error_rate_A = errors_A / tasks_A
error_rate_B = errors_B / tasks_B
standard_error = math.sqrt((error_rate_A * (1 - error_rate_A) / tasks_A) + (error_rate_B * (1 - error_rate_B) / tasks_B))
z_score = (error_rate_B - error_rate_A) / standard_error
p_value = stats.norm.sf(z_score)

print(f"Error count for Pomodoro users: {errors_A}")
print(f"Error count for non-Pomodoro users: {errors_B}")
print(f"Z-Score: {z_score}")
print(f"P-Value: {p_value}")

if p_value < 0.05:
    print("The null hypothesis rejected.\n"
      "Results is statistically significant.\n"
      "Number of errors of users who do not use Pomodoro is higher than\n"
      "number of errors of users who use Pomodoro.")
else:
    print("Null hypothesis accepted. There is no difference between the number of mistakes made by Pomodoro users and the number of mistakes made by non -Pomodoro users.")

```

Figure 6.1: Piece of python code that calculates p-value.

The output of the code snippet of **Figure 6.1** shown in **Figure 6.2.**

```
Error count for Pomodoro users: 11
Error count for non-Pomodoro users: 17
Z-Score: 1.7723477562034653
P-Value: 0.03816842419660913
Standard Error: 0.12133835624986232
The null hypothesis rejected.
Results is statistically significant.
Number of errors of users who do not use Pomodoro is higher than
number of errors of users who use Pomodoro.
```

Figure 6.2: The output of the python code snippet in **Figure 6.1**

Then, a graph showing a standard normal distribution (z-distribution) with the critical region and the observed z-score was plotted using the code snippet in **Figure 6.3** below.

```python
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats

# Parameters
alpha = 0.05
p_value = 0.038

# Calculate z-scores
z_alpha = stats.norm.ppf(1 - alpha)
z_value = stats.norm.ppf(1 - p_value)

# Z distribution for x-axis
x = np.linspace(-4, 4, 1000)
y = stats.norm.pdf(x)

# Plotting the graph
plt.figure(figsize=(10, 6))
plt.plot(x, y, label='Z Distribution', color='blue')

# Shading the critical region
plt.fill_between(x, 0, y, where=(x > z_alpha), color='red', alpha=0.5, label='Critical Region (α=0.05)')
plt.fill_between(x, 0, y, where=(x > z_value), color='green', alpha=0.5, label='Observed Z-Score (p=0.038)')

# Z-score lines
plt.axvline(z_alpha, color='red', linestyle='--', label=f'Z-Score (α=0.05): {z_alpha:.2f}')
plt.axvline(z_value, color='green', linestyle='--', label=f'Z-Score (p=0.038): {z_value:.2f}')

# Graph decorations
plt.title('Z Distribution and Statistical Significance')
plt.xlabel('Z Values')
plt.ylabel('Density')
plt.legend()
plt.grid(True)

# Show the plot
plt.show()
```

Figure 6.3: Piece of python code that draws a z-distribution graph.

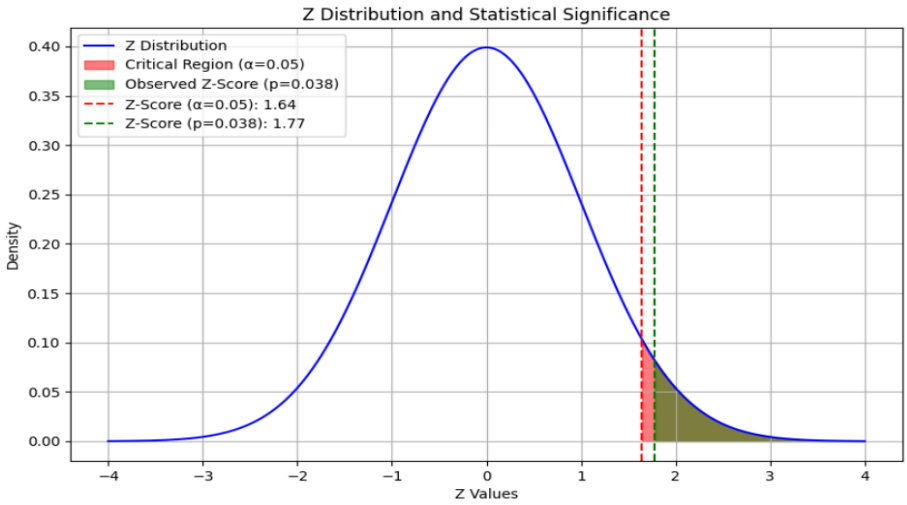

Figure 6.4: The output graph of the code snippet in **Figure 6.3**

The probability density function (PDF) for different z-values is shown in **Figure 6.4**

The following is the interpretation of the first hypothesis graph in **Figure 6.4**.

Z Distribution Curve (Blue Line): This represents the standard normal distribution curve with a mean of 0 and a standard deviation of 1. It shows the probability density function (PDF) for different Z values.

Critical Region (Red Area)**:** The shaded red area represents the critical region, which is the area under the curve where the probability is less than the significance level (α). In this case, α=0.05, indicating a 5% significance level. Values in this region would lead to the rejection of the null hypothesis if the observed z-score falls within it.

Observed Z-Score (Green Area)**:** The shaded green area represents the observed z-score, which corresponds to a given p-value of 0.038. The z-score is the number of standard deviations a data point is from the mean. In hypothesis testing, it helps determine the significance of the observed difference. This area indicates the probability of observing a z-score as extreme as the observed one under the null hypothesis.

Vertical Lines: The red dashed line marks the critical z-score (Z_{α}) at the 5% significance level (α=0.05). Any z-score beyond this line falls into the critical region.The green dashed line marks the observed z-score (Z_{value}) corresponding to the given p-value of 0.038.

Overall, the graph visually represents how the observed Z-score compares to the critical region and the significance level (α). In this case, the observed Z-score falls within the critical region, suggesting statistical significance at the 0.05 level, indicating that the null hypothesis would be rejected.

Then, second hypothesis was tried to be verified using the formulas (6.1), (6.2), (6.3), (6.4) and (6.5).

Second Hypothesis (for highest criticality level error count):

Null Hypothesis (H0): There is no difference between the error count at the highest criticality level of those who use Pomodoro and the error count at the highest criticality level of those who do not use Pomodoro.

Alternative Hypothesis (H1): The error count at the highest criticality level of those who do not use Pomodoro is greater than the error count at the highest criticality level of those who use Pomodoro.

The output of the python code snippet in **Figure 6.1**, adapted for the highest criticality level, shown in **Figure 6.5** below.

```
Number of errors at the HIGHEST criticality level for Pomodoro users: 0
Number of errors at the HIGHEST criticality level for non-Pomodoro users: 3
Z-Score: 1.9086270308410553
Z-Score: 1.9086270308410553
P-Value: 0.02815511117475652
The null hypothesis rejected.
Results is statistically significant.
Number of errors at the HIGHEST criticality level of users who do not use
Pomodoro is higher thannumber of errors at the HIGHEST criticality level
of users who use Pomodoro.
```

Figure 6.5: The output of the python code snippet in **Figure 6.1**, adapted for the highest criticality level.

Below, output of the probability density function for different z-values python code snippet in **Figure 6.5,** adapted for the highest criticality level in shown in **Figure 6.6**.

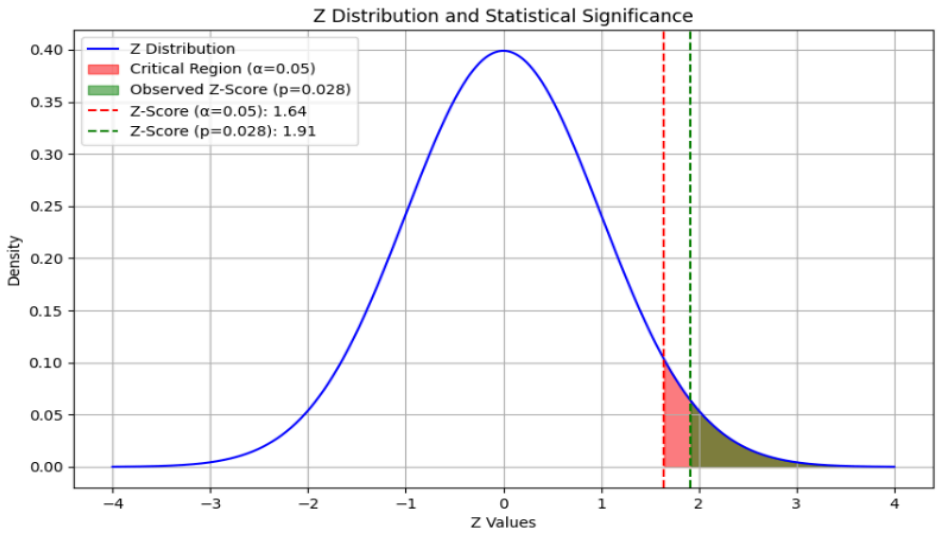

Figure 6.6. Standard normal distribution graph for the highest criticality level error counts.

The following is the interpretation of the second hypothesis graph in **Figure 6.6**.

Critical Region (Red Area): This region is where we obtain statistically significant results. If the Z-score falls into this region, the null hypothesis is rejected. The critical Z-score (1.645), indicated by the red line, corresponds to α=0.05. This is the critical value at a significance level of 5%.

Observed Z-Score (Green Area): The observed Z-score (1.91) is above the critical value (1.645). This indicates that the p-value (0.028) is less than 0.05. Therefore, the null hypothesis is rejected, and the result is statistically significant.

In the summary of Graph, it is shown that the observed Z-score obtained from the statistical test falls within the critical region, and the p-value is less than the alpha value. This indicates that the test result is statistically significant and supports that the error count of those not using Pomodoro is higher than those using Pomodoro. Then, third hypothesis was tried to be verified using the formulas (6.1), (6.2), (6.3), (6.4) and (6.5).

Third Hypothesis (for major, medium, minor criticality level error count):

Null Hypothesis (H0): There is no difference between the error count at the major, medium, minor criticality levels of those who use Pomodoro and the error count at the major, medium, minor criticality levels of those who do not use Pomodoro.

Alternative Hypothesis (H1): The error count at the major, medium, minor criticality levels of those who do not use Pomodoro is greater than the error count at the major, medium, minor criticality level of those who use Pomodoro. Below, output of the probability density function for different z-values code snippet in **Figure 6.1**, adapted for the major, medium, minor criticality levels in shown in **Figure 6.7**, **Figure 6.8**, **Figure 6.9**.

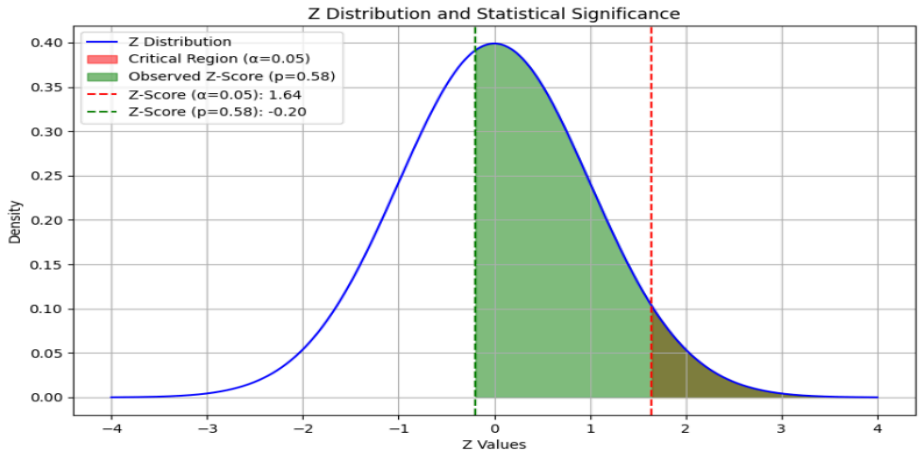

Figure 6.7: Standard normal distribution graph for the major criticality level error counts.

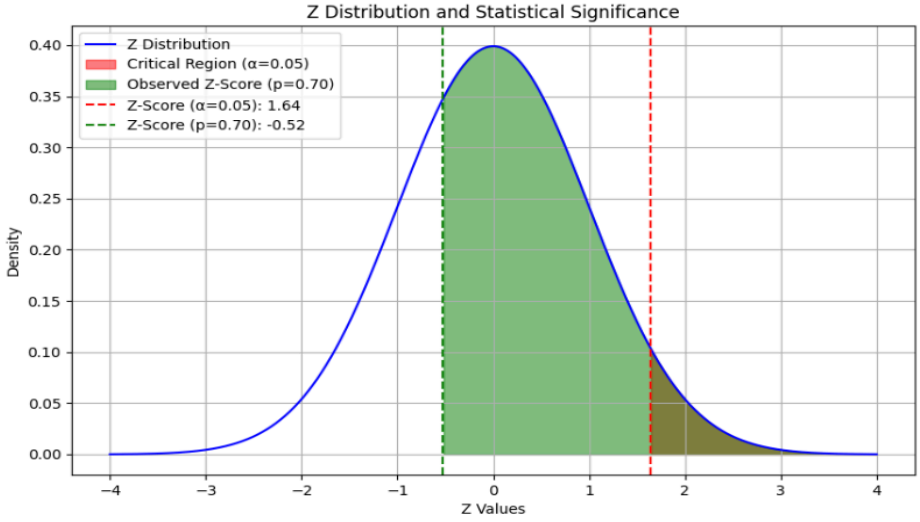

Figure 6.8: Standard normal distribution graph for the medium criticality level error counts.

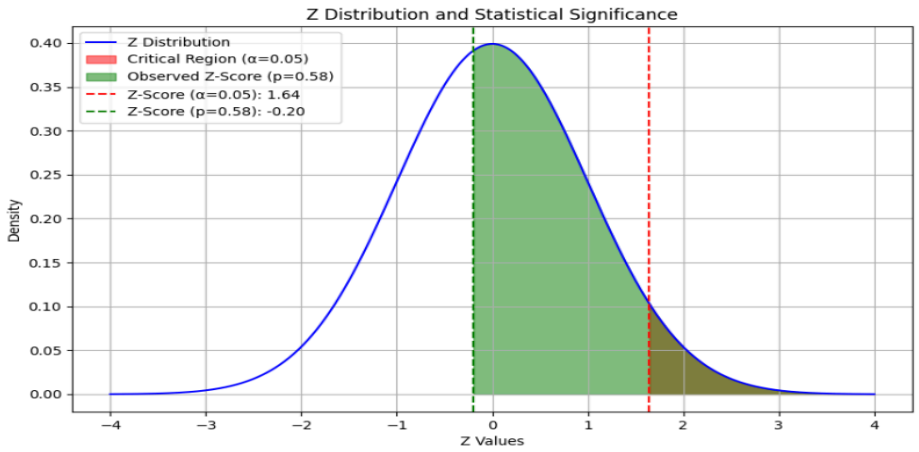

Figure 6.9: Standard normal distribution graph for the minor criticality level error counts.

When the major, middle, and minor graphs are examined, the observed z-score is found to fall outside the critical region, suggesting that the observed result is not statistically significant at the (0.05) level, indicating that the null hypothesis would not be rejected.

6.3. Limitations

This study has some limitations. Firstly, the sample size of 12 individuals is small, which may restrict the generalizability of the findings. Small sample sizes can restrict the applicability of the results to a broader population and may not adequately reflect the effects of individual differences. Furthermore, the study only included participants selected from specific job positions (frontend developers, backend developers, mobile application developers, user experience/interface engineers), which could lead to variations across different job roles and experience levels, potentially affecting the overall validity of the results. This exploratory experiment assessing the effects of the Pomodoro technique was conducted over a short period (two weeks), which may not supply sufficient information about the long-term effects of the technique. Furthermore, the study was carried out in a specific software development area, leaving it uncertain whether similar results would be obtained in different sectors or work environments. Lastly, the effects of participants' personal preferences (such as listening to background music) and work habits could not be fully controlled, indicating that the results may have been influenced by some external factors.

7. CONCLUSIONS

In the current landscape, where communication tools are increasingly integral to both professional and personal interactions, the interdependence between software and computers remains undeniable. Within the field of software engineering, characterized by widespread computer usage, the efficient coordination of developer-computer interactions is paramount. The absence of such coordination can lead to interruptions for developers, disrupting workflow and potentially increasing error rates and stress levels.

Effectively managing these disruptions within the software development lifecycle remains a pressing challenge. The Pomodoro technique has been observed as an effective strategy for reducing interruptions and decreasing error rates among software developers. Research indicates that developers who implement the Pomodoro technique make fewer errors overall, including those of critical importance, compared to those who do not.

Statistical analysis confirms the significance of these findings, with a calculated p-value of 0.038. Based on these results, the Pomodoro Technique improves concentration, resulting in more efficient and less error-prone code, which facilitates the accurate and effective implementation of algorithms and data structures.

A more focused and attentive work environment enables software developers to tackle complex software architectures and design patterns with fewer errors. This enhanced focus is also advantageous for optimizing software performance, as it helps developers write efficient code, identify performance issues early, and address them promptly.

A more concentrated and meticulous work routine contributes to reducing technical debt in software projects by allowing engineers to make fewer mistakes and produce cleaner code, making long-term maintenance and development easier.

As a result, fewer errors lead to more efficient testing processes and smoother deployment phases, ensuring that the software is more reliable and robust before release.

This study has been presented as both a pilot study and an exploratory experimental study, supported by literature indicating that a sample group of 12 individuals is sufficient for such research.

Pilot studies aim to test the feasibility and fundamental effects of a new method or technique on a small sample, typically using small-scale groups. The small sample sizes used in pilot studies allow for initial impressions of the method's core functioning **[71][72].**

Furthermore, exploratory experiments are conducted to gather early-stage information on a new topic and formulate hypotheses for larger-scale studies, and these studies can also be performed with small groups, as the goal at this stage is to collect initial data about the nature and effects of the phenomenon rather than generalizable results **[73].**

A group of 12 is considered an appropriate sample size for such pilot and exploratory studies.

Small-scale studies can often provide sufficient data to understand the fundamental effects of a technique or method and can guide larger-scale research Specifically, data obtained from small groups can inform general trends of methods or techniques and identify potential issues or improvements **[74].** Therefore, the group of 12 used in this study is a suitable size for collecting and evaluating the necessary data in both pilot and exploratory phases.

This thesis contributes to the literature by scientifically examining the effect of the Pomodoro technique on computer interruptions in software development environments.

It highlights the positive effects of this technique on work performance and stress management and advances discussions on effective time management practices tailored to the needs of software developers. In conclusion, these findings advocate for the adoption of systematic approaches to time management and productivity in the field of software development.

This research represents an important step towards designing more effective strategies for managing interruptions, improving user experiences with time management tools, and improving overall developer performance and software quality. However, to generalize these findings, broader-scale and long-term research is required. Future studies should also evaluate the effects of these techniques in various development settings and among different participant profiles.

Such comprehensive studies will deepen the understanding of the impacts of these techniques on work, leading to the development of more effective strategies, particularly within the discipline of software engineering.

REFERENCES

[1] A. N. Meyer, T. Fritz, G. C. Murphy, T. Zimmermann, Software developers' perceptions of productivity, in: Proceedings of the 22nd ACM SIGSOFT Symposium on the Foundations of Software Engineering, SIGSOFT/FSE '14, ACM Press, New York, NY, 2014, pp. 19–29. doi:10.1145/2635868.2635892

[2] A. N. Meyer, E. L. Barton, G. C. Murphy, T. Zimmermann, T. Fritz, The Work Life of Developers: Activities, Switches and Perceived Productivity, in: IEEE Transactions on Software Engineering, 43 (2017) 1178–1193. doi:10.1109/TSE.2017.2656886

[3] Dabbish, Laura, Gloria Mark, and Víctor M. González. "Why do I keep interrupting myself? Environment, habit and self-interruption." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2011.

[4] Martens, Mieke Josephina Elisabeth. "Is the smartphone a performance killer? : The effect of the frequency and intrusiveness of smartphone alerts on the task performance of University students." MS thesis. University of Twente, 2017.

[5] Mansi, Gary, and Yair Levy. "Do instant messaging interruptions help or hinder knowledge workers' task performance?." International Journal of Information Management 33.3 (2013): 591-596.

[6] Bailey, Brian P., Joseph A. Konstan, and John V. Carlis. "The Effects of Interruptions on Task Performance, Annoyance, and Anxiety in the User Interface." Interact. Vol. 1. 2001.

[7] McFarlane, D. C., & Latorella, K. A. (2002). The scope and importance of human interruption in human-computer interaction design. Human-Computer Interaction, 17, 1–61.

[8] Davenport, Thomas H. *Thinking for a living: how to get better performances and results from knowledge workers*. Harvard Business Press, 2005.

[9] McFarlane, Daniel Craig. "Interruption of people in human-computer interaction." The George Washington University, 1998.

[10] McFarlane, Daniel C. "Interruption of people in human-computer interaction: A general unifying definition of human interruption and taxonomy." Naval Research Laboratory, Washington, DC, USA, formal report NRL/FR/5510–97-9870 (1997).

[11] Zijlstra, F. R. H., & Van Doorn, L. (1985), The construction of a scale to measure perceived effort. University of Technology.

[12] Adamczyk, Piotr D., and Brian P. Bailey. "If not now, when? The effects of interruption at different moments within task execution." Proceedings of the SIGCHI conference on Human factors in computing systems. 2004.

[13] Martens, Mieke Josephina Elisabeth. "Is the smartphone a performance killer? : The effect of the frequency and intrusiveness of smartphone alerts on the task performance of University students." MS thesis. University of Twente, 2017.

[14] Grandhi, Sukeshini, and Quentin Jones. "Technology-mediated interruption management." International Journal of Human-Computer Studies 68.5 (2010): 288-306.

[15] Fisher, Cynthia D. "Effects of external and internal interruptions on boredom at work: Two studies." *Journal of Organizational Behavior: The International Journal of Industrial, Occupational and Organizational Psychology and Behavior* 19.5 (1998): 503-522.

[16] Bellandi, Tommaso, et al. "Interruptions and multitasking in surgery: a multicentre observational study of the daily work patterns of doctors and nurses." Ergonomics, 61(1), 40-47, 2018.

[17] Jin, Jing, and Laura A. Dabbish. "Self-interruption on the computer: a typology of discretionary task interleaving." Proceedings of the SIGCHI conference on Human Factors in Computing Systems. 2009.

[18] Dabbish, Laura, Gloria Mark, and Víctor M. González. "Why do I keep interrupting myself? Environment, habit and self-interruption." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2011.

[19] Z. S. H. Abad, M. Noaeen, D. Zowghi, B. H. Far, K. Barker, Two sides of the same coin: software developers' perceptions of task switching and task interruption, in: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering, EASE '18, ACM Press, New York, NY, USA, 2018, part F1377. doi:10.1145/3210459.3214170

[20] Redelmeier, Donald A., and Robert J. Tibshirani. "Association between cellular-telephone calls and motor vehicle collisions." New England Journal of Medicine 336.7 (1997): 453-458.

[21] Iqbal, Shamsi T., and Eric Horvitz. "Notifications and awareness: a field study of alert usage and preferences." Proceedings of the 2010 ACM conference on Computer supported cooperative work. 2010.

[22] Iqbal, Shamsi T., Yun-Cheng Ju, and Eric Horvitz. "Cars, calls, and cognition: Investigating driving and divided attention." Proceedings of the SIGCHI conference on human factors in computing systems. 2010.

[23] Iqbal, Shamsi T., et al. "Hang on a sec! Effects of proactive mediation of phone conversations while driving." Proceedings of the SIGCHI conference on Human factors in computing systems. 2011.

[24] Chang, Yung-Ju, and John C. Tang. "Investigating mobile users' ringer mode usage and attentiveness and responsiveness to communication." Proceedings of the

17th International Conference on Human-Computer Interaction with Mobile Devices and Services. 2015.

[25] Akbar, Fatema, et al. "Email makes you sweat: Examining email interruptions and stress using thermal imaging." Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. 2019.

[26] Blank, Christopher, et al. "Emotional footprints of email interruptions." Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. 2020.

[27] Tseng, Vincent W-S., et al. "Overcoming distractions during transitions from break to work using a conversational website-blocking system." Proceedings of the 2019 CHI conference on human factors in computing systems. 2019.

[28] Fauzan, Abduh. "Literature Review: The Application Of The Pomodoro Technique To Reduce Academic Procrastination Levels Among Students In Completing Thesis." International Conference of Bunga Bangsa. Vol. 2. No. 1. 2024.

[29] Kim, Jaejeung, Chiwoo Cho, and Uichin Lee. "Technology supported behavior restriction for mitigating self-interruptions in multi-device environments." Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 1(3), 1-21, 2017.

[30] T. DeMarco and T. Lister, “Programmer performance and the effects of the workplace,” in Proc. 8th Int. Conf. Softw. Eng., 1985, pp. 268–272.

[31] D. E. Perry, N. A. Staudenmayer, and L. G. Votta, “People, organizations, and process improvement,” IEEE Softw., vol. 11, no. 4, pp. 36–45, Jul. 1994.

[32] B. W. Boehm, “Improving software productivity,” IEEE Comput., vol. 20, no. 9, pp. 43–57, Sep. 1987.

[33] J. Singer, T. Lethbridge, N. Vinson, and N. Anquetil, “An examination of software engineering work practices,” in Proc. CASCON 1st Decade High Impact Papers, 2010, pp. 174–188.

[34] M. K. Gonc¸alves, L. R. de Souza, and V. M. Gonzalez, “Collaboration, information seeking and communication: An observational study of software developers’ work practices,” J. Universal Comput. Sci., vol. 17, no. 14, pp. 1913–1930, 2011.

[35] T. D. LaToza, G. Venolia, and R. DeLine, “Maintaining mental models: A study of developer work habits,” in Proc. 28th Int. Conf. Softw. Eng., 2006, pp. 492–501.

[36] A. J. Ko, R. DeLine, and G. Venolia, “Information needs in collocated software development teams,” in Proc. 29th Int. Conf. Softw. Eng., 2007, pp. 344–353.

[37] T. Roehm, R. Tiarks, R. Koschke, and W. Maalej, “How do professional developers comprehend software?” in Proc. 34th Int. Conf. Softw. Eng., 2012, pp. 255–265.

[38] M. K. Goncalves, L. R. de Souza, and V. M. Gonzalez, “Collaboration, information seeking and communication: An observational study of software developers’ work practices,” J. Universal Comput. Sci., vol. 17, no. 14, pp. 1913–1930, 2011.

[39] R. Minelli, A. Mocci, and M. Lanza, “I know what you did last summer—an investigation of how developers spend their time,” Proc. 23rd IEEE Int. Conf. Program Comprehension, 2015, pp. 25–35.

[40] S. Amann, S. Proksch, S. Nadi, and M. Mezini, “A study of visual studio usage in practice,” in Proc. 23rd IEEE Int. Conf. Softw. Anal. Evol. Reengineering, 2016, pp. 124–134.

[41] Meyer, André N., et al. "The work life of developers: Activities, switches and perceived productivity." *IEEE Transactions on Software Engineering* 43.12 (2017): 1178-1193.

[42] Murphy, Gail C. "Getting to flow in software development." *Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software*. 2014.

[43] J. Nakamura, M. Csikszentmihalyi, The concept of flow, in: Handbook of Positive Psychology (2002). doi:10.1002/9780470172698.ch19

[44] S. Abuhamdeh, Investigating the “Flow” Experience: Key Conceptual and Operational Issues, Frontiers in Psychology 11 (2020). doi:10.3389/fpsyg.2020.00158

[45] Z. S. H. Abad, O. Karras, K. Schneider, K. Barker, M. Bauer, Task interruption in software development projects: what makes some interruptions more disruptive than others?, in: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering, EASE ’18, ACM Press, New York, NY, USA, 2018, pp. 122–132. doi:10.1145/3210459.3210471

[46] Z. S. H. Abad, M. Noaeen, D. Zowghi, B. H. Far, K. Barker, Two sides of the same coin: software developers’ perceptions of task switching and task interruption, in: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering, EASE ’18, ACM Press, New York, NY, USA, 2018, part F1377. doi:10.1145/3210459.3214170

[47] A. N. Meyer, T. Fritz, G. C. Murphy, T. Zimmermann, Software developers' perceptions of productivity, in: Proceedings of the 22nd ACM SIGSOFT Symposium on the Foundations of Software Engineering, SIGSOFT/FSE ‘14, ACM Press, New York, NY, 2014, pp. 19–29. doi:10.1145/2635868.2635892

[48] S. Müller, T. Fritz, Stuck and frustrated or in flow and happy: Sensing developers' emotions and progress, in: Proceedings of the 37th IEEE International Conference on Software Engineering, IEEE/ACM ‘15, 2015, pp. 688-699, doi:10.1109/ICSE.2015.

[49] T. Fritz, S. Müller, Leveraging biometric data to boost software developer productivity, in: Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering, SANER ’16, 2016, pp. 66-77. doi:10.1109/SANER.2016.107

[50] Ritonummi, Saima, et al. "Flow barriers: What prevents software developers from experiencing flow in their work." *CEUR Workshop Proceedings*. RWTH Aachen, 2022.

[51] Parnin, Chris, and Spencer Rugaber. "Resumption strategies for interrupted programming tasks." Software Quality Journal, 19, 5-34, 2011.

[52] Ruensuk, Mintra. "An implementation to reduce internal/external interruptions in Agile software development using pomodoro technique." 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS). IEEE, 2016.

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

[54] Baethge, A. and Rigotti, T. (2013). Interruptions to workflow: Their relationship with irritation and satisfaction with performance, and the mediating roles of time pressure and mental demands. Work & Stress, 27(1):43–63.

[55] Mangalji, Ezaan, et al. "Blockit: Time Blocking Made Easy.

[56] Alkahtani, Mohammed, et al. "Human Interruption Management in Workplace Environments: An Overview." *Engineering, Technology & Applied Science Research* 10.2 (2020).

[57] Khoshbakht, Maryam, Eziaku O. Rasheed, and George Baird. "Office distractions and the productivity of building users: The effect of workgroup sizes and demographic characteristics." Buildings 11.2 (2021): 55.

[58] Farivar, Farveh, Osveh Esmaeelinezhad, and Julia Richardson. "Digital intrusions or distraction at work and work-Life conflict." New Technology, Work and Employment 37.3 (2022): 363-380.

[59] Kuutila, Miikka, et al. "Individual differences limit predicting well-being and productivity using software repositories: a longitudinal industrial study." Empirical Software Engineering 26.5 (2021): 88.

[60] Brumby, D.P., Janssen, C.P., Mark, G. (2019). How Do Interruptions Affect Productivity?. In: Sadowski, C., Zimmermann, T. (eds) Rethinking Productivity in Software Engineering. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4221-6_9.

[61]] Noteberg, S. (2009), "Pomodoro technique illustrated: The easy way to do more in less time."

[62] Yadav, Vanita. "A flexible management approach for globally distributed software projects." Global Journal of Flexible Systems Management 17 (2016): 29-40.

[63] Linderbaum, Beth A., and Paul E. Levy. "The development and validation of the Feedback Orientation Scale (FOS)." Journal of management 36.6 (2010): 1372-1405.
[64] Söğüt, Selin. "Pomodoro Tekniği ile Zamandan ve Dikkatten Tasarruf Etme Sanatı." 2015.
[65] Atlassian. Jira Software, from https://www.atlassian.com/software/jira.

[66] Pressman, Roger S. "Software Engineering: A Practitioner's Approach." (7th Edition)

[67] Dizon, R. J., et al. "The effects of pomodoro technique on academic-related tasks, procrastination behavior, and academic motivation among college students in a mixed online learning environment." Globus Journal of Progressive Education, 11, 58-63, 2021.

[68] Browne, Robert, Luke Raeside, and Geraldine Gray. "Gamification in education: productivity and motivation through gamified time management software." European Conference on Games Based Learning. Academic Conferences International Limited, 2018.

[69] Bhandari, P. (2023, June 22). An Easy Introduction to Statistical Significance (With Examples). Scribbr. Retrieved June 3, 2024, from https://www.scribbr.com/statistics/statistical-significance/.

[70] Kul, Seval. "İSTATİSTİK SONUÇLARININ YORUMU: P DEĞERİ VE GÜVEN ARALIĞI NEDİR? / INTERPRETATION OF STATISTICAL RESULTS: WHAT IS P VALUE AND CONFIDENCE INTERVAL?." Plevra Bülteni, 8(1), 11, 2014.

[71] Thabane, L., Ma, J., Chu, R., et al. (2010). "A tutorial on pilot studies: the what, why and how." BMC Medical Research Methodology, 10(1), 1-10. doi:10.1186/1471-2288-10-1.

[72] Maxwell, J. A. (2012). Qualitative Research Design: An Interactive Approach. Sage Publications.

[73] Button, K. S., Ioannidis, J. P., Mokrysz, C., et al. (2013). "Power failure: why small sample size undermines the reliability of neuroscience." Nature Reviews Neuroscience, 14(5), 365-376. doi:10.1038/nrn3455.

[74] Statistical Power Analysis for the Behavioral Sciences" (1988). Cohen, J. Erlbaum Associates.

BIOGRAPHY

Hatice DIŞLI completed his secondary education at Akyurt Anatolian High School. She graduated from Ankara Yıldırım Beyazıt University, Department of Computer Engineering in 2017. She started her master's degree at Gebze Technical University in 2022. She currently continues her work as a senior expert researcher at The Scientific and Technological Research Council of Turkey (TÜBİTAK).

PUBLICATIONS AND PRESENTATIONS FROM THE THESIS

Dişli H., Göktürk M., (2024), “Bilgisayar Tabanlı Uygulama Bildirimlerinden Kaynaklanan Kullanıcı Kesmelerinin Görev Performansı Üzerindeki Etkin Yönetimi”, GTÜ 8. Lisansüstü Araştırmalar Sempozyumu

Dişli H., Göktürk M., (2024), “Strategies for Mitigating Computer-Based Interruptions for Software Developers”, 3rd Eurasian Conference of Human-Computer Interaction HCI-E. (accepted, conference date 22-23 November 2024)