Managing Developer Interruption

by

Gabrielle Cristina Perez Dias

A thesis submitted in partial fulfillment
of the requirements for the degree of

Masters of Science

in

Computer Science

University of Ontario Institute of Technology

Supervisors: Dr. Christopher Collins and Dr. Jeremy Bradbury

November 2018

Abstract

Society in general has several distracting elements on their daily routine. With so many devices and notifications showing in your screen or making sounds around you, it is not difficult to think that focus is something really hard to keep mainly when talking about cognitively-intense activities.

Interruption occurs to individuals anytime someone or something takes their attention during focused activities. The high frequency of interruptions during cognitively-intense activities can be annoying and detrimental to deadline-driven work, such as software development.

Some researchers focused in how to avoid interruptions, the main focus of this thesis is in how to recover from them though. The main challenges in managing interruptions are:

1) Interruptions will occur even if all the notifications, phones and buzzers are turned off.

2) Interruptions are not just accidental but they can also be deliberate such as lunch, bathroom breaks or meetings.

In this thesis, we propose a solution that addresses these challenges by focusing on the recovery of momentum based on the understanding that interruption recovery involves knowledge about the interrupted activity, the developer, as well as the context of the work.

Acknowledgements

First, I would like to thank God for this and all the other accomplishments in my life.

Secondly, I am very thankful for this opportunity and all the support and advice received from my supervisors Prof. Dr. Christopher Collins and Prof. Dr. Jeremy Bradbury.

I would also like to thank my family, in special my parents for all the guidance, love, support and infinite patience with me through all my life in special through all this thesis development process and my boyfriend for all the patience, care and love mainly throughout this period of my life. Without you, I could not have done this or any other thing.

I cannot forget to thank all my friends in both Software Quality Research Lab (SQRLAB) and Visualization for Information Analysis Lab (VIALAB) for all the advice, support and help.

Lastly, the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC) is gratefully acknowledged.

Contents

List of Figures

List of Tables

Abbreviations

CPU Central Processing Unit.

GPU Graphics Processing Unit.

GUI Graphical User Interface.

IDE Integrated Development Environment.

NLTK Python Natural Language Processing Toolkit.

OpenCV Open Source Computer Vision Library.

Chapter 1

Introduction

Throughout this work, both terms programmer and developer are used, in this scenario they have the same meaning and therefore are interchangeable.

1.1 Problem Definition

Researches show that developers have a routine with several interruptions [4, 10, 31]. According to Parnin and Rugaber [33] programmers are likely to get only 2 hours of uninterrupted work session in a day.

An interruption can be defined as anything that changes someone's focus from the task being executed. Interruptions can be long (1-2 hours) or short (15-30 minutes) [33], they can be self-initiated [7], externally-initiated, a distraction, an intrusion or a break [4].

The nature of the interruption, its length or duration, its complexity and context may change how one might recover from it and the time spent to do so. Another factor that directly affects the recovery time is the amount of time used to collect one's thoughts either before interruption or during it [20]. When the task is self-initiated

the programmer's brain has time to "prepare" - rehearse - for a later resumption. This time is called interruption lag, the longer the interruption lag, the easier and faster the recovery. It is possible to find a formal definition on Altmann and Trafton [2] "Interruption lag is a brief transitional interval immediately preceding an interruption, during which the operator knows the pending interruption but is not yet engaged by it". This preparation might be done by encoding goals to accomplish later for example. If programmers have time to do so, they usually make use of something more physical, such as, making notes on Post-it's, inserting intentional execution errors, inserting comments and TODO markings[1] [33].

Research shows that a programmer often spend between 15 30 minutes recovering from long interruptions before getting back to work [33, 40]. It is not uncommon developers to forget everything about the task they were working on [30].

The high frequency of interruptions can cause annoyance, anxiety, frustration, errors, stress, context-switching costs and time pressure due to the increase of a task completion time [1, 5, 6, 20, 26]. However, some interruptions are rather beneficial to the developer, such as bathroom or lunch breaks. According to Chong and Siino [4], some interruptions are also "essential for swapping or gaining information required for high quality work".

Several researches focus on avoiding interruptions or decreasing their frequency, these approaches address the annoyance and stress but they do not address the time spent in the recovery neither the effort spent in a recovery task in the inevitable interruptions that will happen - either more or less times.

[1] A TODO note is a marking designed to list points that requires further attention, these markings can be done differently according to the used application but are usually done by writing \\TODO in the code. They can be seen in a TODO list and they are also signalized when viewing the file.

1.2 Thesis Statement

Thesis statement: A curated video of a developer's previous work section can help with the recovery process in terms of velocity and confidence.

This thesis presents a tool that gathers data, processes a screen captured video and presents it to developers. There is an assumption that with this curated video the developer will be able to rebuilt the context of the tasks, therefore helping with the recovery process.

1.3 Thesis Contribution

The main contribution of this work are:

1) A framework of captured, processed and further shown data in a form of recorded screen and edited video to address interruption recovery.

2) An algorithm that describes developer's activities and scores them in order to detect importance of moments in one's work section.

A tool was developed and tested in a user study to prove its efficacy. This tool was built with several modules. Some of these modules can also be cited as a sub-contribution. The module that classifies a text as being code, formal text or personal text, using some libraries and a set of rules, has an accuracy of 95%, further discussion can be found on Chapter 3. The module that checks for processes - applications - changes and stores interactions through mouse and keyboard can also be mentioned as a contribution. This module does not store physically the typed text, it only processes everything in memory. That can be easily changed to do such a thing depending on the scenario to be analyzed.

This thesis is the first draft of how this tool would work and look like. The study presented here shows if gathered data of simple hardware (normally available to all

programmers - such as webcam, keyboard and mouse/touch pad) along with a full curated video would be beneficial to a developer when recovering from an interruption. The tool can be later edited to improve the detection of important moments using other techniques such as machine learning without having to be entirely rewritten.

The main assumption of this thesis work is that almost none of the available tools have considered all programs - instead of only Integrated Development Environment (IDE) - and therefore lacked on a fully context retrieving task. This was also pointed by Rule et al. [35]. Since the beginning, the availability of the used hardware in a normal development environment was taken in consideration, therefore there is no use of eye tracker or any research exclusive and/or not commonly used external device to gather data because we wanted to make a tool that could be used by all developers.

1.4 Thesis Organization

This thesis is organized as follows:

- Chapter 1 provides a brief overview to the interruption problem, it presents the contribution of this thesis and the background required to understand some of the main assumptions made to create the approach.
- Chapter 2 presents some of the related work used as reference to the initial decisions.
- Chapter 3 describes our approach to help developers recovery from interruption.
- The user study made to prove the tool efficacy is described and its results are presented on Chapter 4.
- Lastly, Chapter 5 contains the conclusion and future work.

Chapter 2

Background

Advances in technology and the fact that communication devices have become much more affordable and indispensable in people's lives have not only increased our ability to communicate but have also increased our chances to get interrupted while working [11]. Developers and knowledge-intense workers face blocked tasks and interruptions in their daily routines, [29,32] and in these cases, the interruption can be a significant issue due to the complexity of their tasks [4].

Throughout the academic literature related with interruptions and developer's routine, there are several definitions and types of interruptions. Jett and George [17] define interruption as "incidents or occurrences that impede or delay organizational members as they attempt to make progress on work tasks". They describe four types of interruptions: intrusions, breaks, distractions and discrepancies.

Studies show that interruptions are inevitable [11] due to the multitask environment that cognitive-intense work companies have and the likelihood of interruption increases if we take in consideration that most of the software development is done in teams, and therefore can be considered collaborative work [4]. This work can involve scheduling meetings, requests from co-workers, or blocked tasks due to issues

in communication between colleagues.

Parnin and Rugaber [32] conducted an experiment to understand how developers currently deal with interruptions. They mention three states of handling an interruption: suspension, resolution and resumption. Suspension is the period of time before attending to the interruption, where interrupted developers can preserve their working state for faster recovery. This preservation might be done internally or externally where there is something physical or electronic to help recovery. The concept of suspension is also mentioned in Altmann & Trafton [2] and Hodgetts & Jones [13], as a period of time with a crucial role in managing the recovery process. The importance of suspension time is also confirmed by Labonté et al. [21], who showed that the use of a pre-interruption warning would give time to the participants to perform the following tasks mentioned in Trafton et al. [39]: retrospective rehearsal - collect information about what they were doing - and prospective goal encoding - collect information on what they were about to do - before attending to the interruption itself improving then, the decision speeding. This means that rehearse and the amount of time available to do it plays an important role in how people will recover from an interruption due to the amount of information available to do it.

Resolution is the period of time where one is attending to the interruption itself. Resumption is the recovery process to "get back to the zone" [31]. The main issue for developers is that the time one might have available to rehearse varies according to the type of interruption. When developers have time to prepare for an interruption, researchers found that they make use of several strategies to generate external cues.

In Czerwinski et al. [6]'s study, they mentioned the use of self-emailing reminders and the use of web pages with a set of task reminders. Parnin [31] also mentioned the use of memory aids along with possible tools to create these task reminders in an electronic or better way. He mentions the use of TODO comments, and proposes

Smart Reminders - a reminder triggered by specific conditions such as a code review done by a colleague or proximity to a reminder - to call one's attention to specific comments. He also addresses the fact that there is no impetus to address or review the TODOs. He mentions the fact that sometimes developers need to change code in several different locations, and that it can be difficult to remember all of these locations. Therefore, developers tend to leave their last open windows visible, or to keep highlighted lines and code error messages. Parnin proposes then, the use of *Touch Points* - a tool that allows developers to track status across different locations.

Parnin also suggests *Code Narrative*, an application that helps developers to recall contextual details, and *Sketchlets*, a tool that contains a visual plane with a set of annotations or an alternative representation projected on top of programming elements, this would help programmers to form concepts and support abstractions. The majority of these suggestions have one thing in common: the need for user interaction. These tools and applications will not work without developers clicking somewhere, so for the tool to know that a specific point is important, the developer needs to write something to indicate that. Another interesting point mentioned by Parnin is that during an interruption, different types of memories will be affected, and each of the proposed applications was designed to aid specific types of memory.

In our approach, we tried to incorporate some of the ideas behind the suggestions in order to help with the different types of interruption affecting different types of memory, however we attempted to avoid the need for any developer action except to start the tool before working and to watch the video to recover.

The use of a video log with a recorded screen has proven to be beneficial as a memory aid [6, 35]. It was also proven to be a worse recovery aid than doing nothing by John et al. [18] in a study where the main task was related with monitoring complex situations and detecting significant changes. In their experiment, changes

would occur while the participant attended to an interruption, and when returning, participants would have to acknowledge all of the changes. Although some developers split tasks with a team, and code would still be developed while one is attending to an interruption, the nature of a developer's work is different than this kind of situation.

One proven disadvantage of using video logs, is the high use of resources that can affect performance, and the fact that people consider video logs tedious or annoying to watch [35]. We try to address this situation by using a curated, shortened video. In our study, we compare results and determine if the nature of a developer's work enables the use of a video log to provide real benefit.

In Rule et al. [35], it is shown that the use of a screen recorder can help workers to rebuild mental context about their previous tasks. The authors discuss how annoying an interruption recovery can be and how much more difficult it is to recover from intense tasks. Their approach was to take a screenshot every time a click, type, or pause is made for a few seconds. One of the claims of the study is that thumbnails were more effective than videos due to the fact people tend to only remember what the video shows directly, rather than forcing the memory to process the video and recreate context. This thesis tries to overcome that claim with the use of a curated video, which is quicker than real time in order to jog the memory. Other notable differences include the important moments detection and the time-limitation. The tool in this thesis do not show a full video, because it is curated according to important and non-important moments in a different way than the study presented by Rule et al. [35]. The video is supposed to be less annoying to watch due to its length, and the amount of work represented by the video is lower than Rule et al.'s study where participants would review the video at the end of the day instead of the end of a session.

In order to define what is important during a developer's session, it is necessary

to understand what a developer does, what tasks are included in their routine, and how those tasks are prioritized. In Meyer et al. [29], Singer et al. [37], and LaToza et al. [22] it is possible to see the tasks presented in a developer's routine. To better analyze those studies, we created a color coding and tried to abstract the presented tasks in seven task abstractions:

- Coding;
- Reading code;
- Search/Reading;
- Testing;
- Typing work-related text;
- Typing personal texts (not important for the our recovery process);
- Important tasks not covered or supported by our approach;

For the last abstraction, "not supported" describes activities done in Remote Desktop that currently, are not fully supported by the tool. It is important to note that some of the abstractions/classifications presented here were already presented in the original papers, we then, used the same classification only adapting it to our own color-code or fitting the abstractions in our own. We used the group of tasks presented in Meyer et al. [29]'s original table, but with color-coded abstractions in order to match the other tables presented to facilitate comparison and analysis.

Task abstractions:	
Code/Task	Color/Task
C	Coding;
RC	Reading code;
SR	Search/Reading [>> Selecting >> Copying >> Pasting];
T	Testing
WRT	Typing work-related text;
PT	Typing personal texts (not important for the tool);
NS	Important tasks but not covered or supported by the tool

Singer et al.	
SR	Read documentation
RC	Look at source
WRT	Write documentation
C	Write code
NS	Attend meetings
SR	Research/identify alternatives
NS-WRT	Ask others questions
NS	Configure hardware
NS-WRT	Answer questions
C	Fix bug
C	Design
T	Testing
RC	Review other's work
C-RC-SR-WRT	Learn
T	Replicate problem
C-RC	Library maintenance

LaToza et al.	
C-RC-SR	Analyzing a new problem and mapping out the broad flow of code which will be used to solve the problem. […]
C	Creating a new method, source file, or script and getting it to a compilable state
RC	Determining information about code including the inputs and outputs to a method, what the call stack looks like, why the code is doing what it is doing, or the rationale behind a design decision. […]
C	Editing existing code and returning it to a compilable state.
T	Ensuring that code is behaving as expected. […]
NS-WRT	Any computer mediated or face-to-face communication about information relevant to a coding task […]
WRT	Any other code related activities including building, synchronizing code, or checking in changes.
C	Other code [No description provided]
PT-WRT-NS	Non code

Meyer et al.	
C	Reading/Editing/Navigating code (and other code related activities)
C	Use the debugger inside the IDE
RC	Performing code reviews
RC-WRT-T-C	(Version Control) Reading/Accepting/Submitting changes
WRT-PT	Reading/Writing emails
C	Editing work items/tasks/todos; Creating/changing calendar entries
RC-WRT-SR	Read/write documents reading/editing documents and other artifacts
NS	Scheduled meeting/Call
NS-WRT	ad-hoc, informal communication; e.g., unscheduled phone call / IM
SR	Internet browsing related to code/work/task
NS	Internet browsing work unrelated
NS	Anything else (changing music, updating software, using the file explorer or having a break)
NS	Remotedesktop use which could not be mapped to another category

Figure 2.1: Task abstractions from: Singer et al. [37], Meyer et al. [29] and LaToza et al. [22]

As shown in Figure 2.1, the presented tasks were abstracted in high level tasks. The gray color represents tasks that are not considered as important to the tool as coding, but are part of a developer's routine and are a type of an interruption. The majority of the time in a developer's day is spent on code [29]; this can also be seen in the figure when taking a look at the amount of green activities. In order to code, a developer might need to have a meeting to define parameters or requirements, but in this scenario, "important" tasks to the tool are those where the developers are alone or in a pair at their computers working on the already defined task. Anything that removes the developer's main focus from the task, is an interruption.

Although we acknowledge the interruptions caused by non-work related websites access [27] or interruptions and distractions caused by background noise [**?**, 36], this thesis will focus on recovery from interruptions or self-interruptions outside the computer that tend to be longer than distractions or computer-based interruptions. We believe that distractions using the computer can be easier to recover from, as they would not require a full review of a work section due to their short duration. As can be seen in Sykes [38], there are some easy options to reduce computer-based interruptions such as:

- Only using instant messaging when necessary;
- Self-imposing task switches;
- Customize email settings about notifications;
- Encouraging the use of short email messages instead of long ones.

Sykes also mentions changes in the physical workplace environment including people's behavior or even office culture. We believe that these changes should be applied as another measure to reduce the effects of interruptions by the company to help

their employees but at the same time, we understand that those measures are outside of a developers control. This work will focus in helping individuals to recover from interruptions that would occur despite other precautions.

Email-checking frequency is a well-known measure. In Gupta et al. [12] it is said that the frequency of changing focus to a secondary task such as emails can decrease performance for the first task and that this frequency should be only two to four times a day according to their study. The same reduction is suggested by Jack et al. [15] as a measure to reduce interruption effect. However, attempted email-checking frequency reduction alone is not solely sufficient, as reducing or canceling new emails notifications might reduce interruptions, but Iqbal and Horvitz [14], showed that for some of the people actually increased when notifications were disabled. One explanation for this phenomenon is that with a canceled or reduced amount of notifications, participants were checking their emails manually several times to ensure that emails were answered on time. In a post study survey, they also got results that from the 18 participants, 17 would revert back to using notifications.

Several studies were conducted in this field, some of them trying to understand the effects of an interruption [16], or to understand the recovery process [31], while others attempt to avoid interruptions [11] or even study how the interface/style of a notification can affect the interruption [34]. The observations provided by all of these studies motivated Kersten et al. [19] to develop an extension for Eclipse IDE. With this extension, a developer could start a new task and the extension would record the files changed for that specific task to create a filter for the next time the developer worked at the same task. This would allow developers to not only recover from interruptions, but also to switch between several tasks. One of the problems we saw with this approach was the restriction for only one application, while a developer's task might use several applications simultaneously.

Chapter 3

Approach

The approach to help with interruption recovery was to come up with a tool that would record data, process it and show what one might need in order to rebuild context to get back to work.

3.1 The tool

FastRecovery is a tool developed to be an automatic tool that, collects all the data that it needs, detects when one has been interrupted, starts the processing and prompt the video/information one might require to recovery from an interruption. We define the tool as automatic because after being started, it passes through all processes without requiring an extra action from the user. The tool could also be integrated with services that starts along with the operational system and the developer would not even need to start the tool.

In order to facilitate understanding, the tool can be split in four parts: gathering data, interruption detection, processing data and Graphical User Interface (GUI).

3.1.1 Gathering data

Our approach relies on detection of important moments, in order to do so, we need to gather data from the work section. Furthermore, to guarantee data being collected "at the same time" from different sources the tool runs through threads.

In order to facilitate implementation, comprehension, and further analysis of data, we have three concepts: User section, App Section and Ten second windows. See Figure 3.1

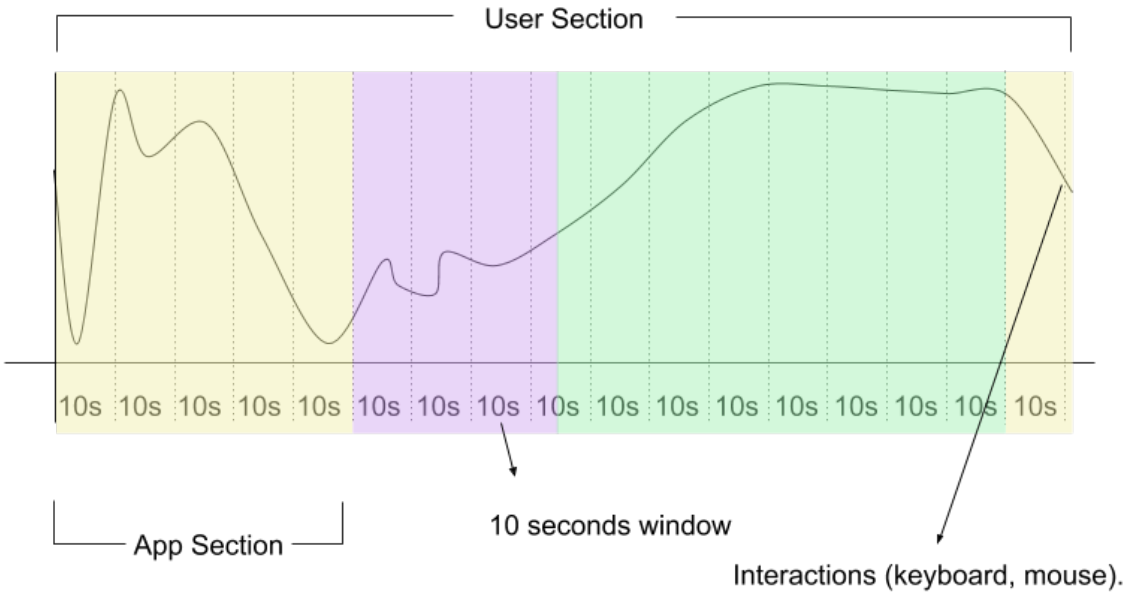

Figure 3.1: User section, App section and Ten Second Windows representation

An User Section is the period of time that starts after the last recovery process until the next interruption time. The App Section is the period of time that starts when one opens a new application (program) - changing the active window - or when the tool is initialized, whatever comes first until one changes the active window or an interruption is detected, whatever comes first. The Ten Second Windows are the smallest part of an app section, they exist to address long app sections and facilitate choosing the best moments instead of one hour long best moment. Even though they

were created to address long sections, they are also used for small ones in order to maintain consistent in the system through all the collected data. To gather data we have the following six modules running in parallel through the use of threads:

- One that records the screen;
- One that keeps checking for active application changes;
- One that organizes the data and keep track of ten second windows;
- One that records keyboard actions;
- One that records mouse actions;
- One that processes the features present in the typed text.

The collected data involves: Mouse/TrackPad clicks, Mouse/TrackPad movements, Mouse/TrackPad right clicks, Mouse/TrackPad click location, pressed keys, actions made (copying, pasting, saving, deleting), used application and typed text (text is processed but not stored).

Recording screen

The process of recording one's screen has proven to be successful when talking about recovery processes, our approach then, uses this method along with the important moments detection to reduce the amount of seen content and/or the time spent seeing it.

A lot of research was done in order to find the best cross-platform way to record screen. As presented in Chapter 2, some studies used screenshots but our goal was to use video to show the entire thinking flow. One of our assumptions was that a curated video might be beneficial to rebuild context.

The best tool we could find that worked on Mac, Windows and Linux is called FFmpeg [1] and it records the screen through the command line and can be triggered from Python within a thread which covers all our scenarios. The thread to record the screen is started as soon as the tool is started, and it keeps running until an interruption is detected.

One drawback of recording a screen is Central Processing Unit (CPU) usage that can directly affect the velocity of the computer, and therefore can affect the task completion which causes anxiety [25]. In order to address this performance issue, we tried different parameters combination to obtain the best balance between video quality and performance.

As an additional measure to improve performance, we included the use of Graphics Processing Unit (GPU), when available, to reduce CPU usage. While running some tests, for user sections longer than two hours, we were getting a CPU usage of more than 200% on a Mac environment. After the use of GPU, CPU usage dropped to a maximum of only 25%. The same concept was applied to Windows versions. The only issue is availability and compatibility since we depend on GPU integration with FFmpeg library. Currently, the only supported GPUs are h264_videotoolbox for Mac environments and h264_nvenc for Windows with NVidia. Other than that, the tool will be relying exclusively on CPU processing.

Checking application changes

Developers' work nature requires the use of different applications at the same time. Sometimes, a browser to test, sometimes more than one browser to test, IDEs to code, email application, terminal to compile, calendar, music player and others that

[1] FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much anything that humans and machines have created. It supports the most obscure ancient formats up to the cutting edge. - https://ffmpeg.org/about.html - Last accessed on: October 9 2018

can vary according to the used technology and project to be developed. Some of these applications might be directly related to work and some of those might be almost like a distraction that makes the work more comfortable such as music player, others might be used for both situations such as a browser or an email applications. It is important for our approach to know which application is being used and associate the gathered data with the correct application because this connection might increase or decrease the importance of a section.

Our tool checks every second for active window changes, this is done differently according to the operational system being used. Once the thread detects that there is a change in the active window, it triggers the thread that process the text - in memory 3.1.1 -, stops the current thread responsible to track ten second windows and starts a new one for the new active window 3.1.1.

Managing Ten second windows

As mentioned in the previous section, to detect the application being used is important because this might change the importance of what is being done inside of it. At the same time, having only the application to delimit small blocks of work, might not be enough since one can spend several hours in the same application mainly when we talk about IDEs that might have console and emulators integrated with it. We need a more granular separation in order to detect important moments that would be more specific than a 20, 10 or even 5 minutes period.

It is important to manage small windows in order to keep the required granularity of changes, to get small important moments instead of long app sections where there can be highs and lows. Inside the management of small windows there is the capture of keys, clicks, keys combinations, keywords done by triggering keyboard and mouse listeners 3.1.1. There is the constant check for any kind of input from the user, if

there is none, we trigger the process to detect an interruption 3.1.2.

Keyboard/Mouse actions

As explained in previous sections, our approach relies on importance of moments and it uses gathered data to detect them. It is important to know the amount of interaction coming from the developer because this will also indicate interruptions. The most common inputs in a developer's setup are mouse - or touchpad - and keyboards.

A lot of research was done to find the best way to capture data cross-platform. We found a library developed by user moses-palmer called Pynput [2]. It uses a combination of native libraries provided by each operational system to provide listeners that allow monitoring key and mouse events.

Processing text features

In order to define importance of moments inside a work section, one of the data we consider is the typed text, it is important to identify if it is code or not, as stated in Chapter 2, this tool considers code text to be more important than plain text.

The text features are extracted for each application in order to improve the score of that specific app section. Due to privacy and security concerns the tool does not record and store immediately the text that the user typed - which can be passwords or other sensitive information - during the app or user section, all text is kept in memory. Therefore, to increase accuracy without sacrificing privacy and security, the text is processed in memory which ensures enough context after the developer changes the active window.

[2]Pynput is an open source library developed in Python available on Github - https://github.com/moses-palmer/pynput - Used version 1.4 - Last accessed on: October 9 2018

In order to differentiate code of plain text, it is necessary to check if there are any existent symbols among the characters to be analyzed, it is not common to have several of them in a normal conversation. Symbols such as "(, (, _, {, }, [,], <, >, +, =, |, \" are not common in our daily communication, thus, a high percentage of text formed by symbols might imply that the text is a code or it might be due to the use of emoticons [3].

Furthermore, we considered the inclusion of emojis [4] and emoticons [5] in the text. To incorporate this new way of communication, the tool makes use of a python module called emoji that contains unicode emojis [6], their aliases [7] and representations, the tool also uses a list of western emoticons that was compiled from Wikipedia [8] - See Appendix A.

The library and the compiled list are used to check the amount of symbols that are in fact emojis/emoticons to handle misidentifying the text or state as an early prediction of 'CODE'. However, the way code is done, some symbols may be arranged in a way that they would be among the emoticons list, that is why there is a threshold of emoticons inside a code text and other verification steps are necessary in order to improve detection and distinction between code and text.

One of these steps is to check the typed words against the stop word[9] list from

[3] "A representation of a facial expression such as a smile or frown, formed by various combinations of keyboard characters and used to convey the writer's feelings or intended tone." - https://en.oxforddictionaries.com

[4] "A small digital image or icon used to express an idea or emotion." - https://en.oxforddictionaries.com

[5] "A representation of a facial expression such as a smile or frown, formed by various combinations of keyboard characters and used to convey the writer's feelings or intended tone." - https://en.oxforddictionaries.com

[6] The following emojis are supported by the library: http://www.unicode.org/emoji/charts/full-emoji-list.html - Version 11

[7] The following aliases are supported by the library: https://www.webpagefx.com/tools/emoji-cheat-sheet

[8] https://en.wikipedia.org/wiki/List_of_emoticons - Last accessed on: June 2, 2018

[9] "A word that is automatically omitted from a computer-generated concordance or index." - https://en.oxforddictionaries.com

Python Natural Language Processing Toolkit (NLTK) [3][10] and also NLTK Words and Wordnet [11].

Currently, the tool only supports English for natural language but several programming languages. The process could be reversed, but in this scenario, mainly considering that there is an user study, it is more important for this moment, to support more programming languages than natural languages.

To help increase accuracy, we made a compilation of programming language words - See Appendix C. The programming languages are: Javascript, Python, Java, Ruby, PHP, C++, CSS, C#, GO, C, Typescript, Shell, Swift, Scala, Objective-C and they were chosen from the list of fifteen most popular languages on GitHub [12]. As previously stated, the approach developed in this tool was intended to be as inclusive as possible with the developer community in regards to support different programming languages. And, although there are only fifteen languages here, these keywords only aggregates information, which means that any developer using the tool with another language will not have an issue. The accuracy of text detection will slightly decrease, and if the chosen programming language has words in common with the languages presented here, the accuracy might not even be affected.

As can be seen in Lo et. al. [23], stop words such as pronouns, conjunctions, prepositions can be found in almost all natural language documents. Typically representing 20% to 30% of the words in them. Which make them very valuable in our

[10] "NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum." - https://https://www.nltk.org/ - Last acessed on: September 2018

[11] George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41. Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

[12] Github created the website https://octoverse.github.com/ - Last acessed September 2018 - in order to show some numbers from 2017. The list used in this thesis is a list of the fifteen most popular languages on GitHub by opened pull request

approach because a high percentage of use combined with a low percentage of symbols can indicate natural language. Since there might be some comments and instructions inside a code, there are two other NLTK modules used to increase the percentage of "English words", Words and Wordnet packages inside NLTK. The Wordnet package is a lexical database from Princeton University with more than 140 thousand words and the Words package is a public domain package with more than 230 thousands English words. Combined these two modules offer more than 330 thousand different English words.

After detecting text as being non-code, our tool makes use of a NLTK module called VADER [9] used to detect sentiment. VADER not only uses matches words in the text with its own lexicon to detect sentiment but also takes in consideration capitalization, exclamation points, words such as "but" that changes the sentiment and intensifier adjectives such as "extremely". The module then, provides four numbers used in this tool calculations:

- Pos that corresponds to positive sentiment;
- Neg that corresponds to negative sentiment;
- Neu that corresponds to neutral sentiment;
- Compound that corresponds to the sum of the lexicon ratings standardized between -1 and 1 where -1 is negative, 1 is positive and 0 is neutral.

Mostly, formal content tends to be more neutral according to tests. The parameters used in some of the rules were tuned to ensure accuracy. After tests with more than 100 code files (considering the code for this project, some code of the libraries used and also some personal projects - among the programming languages were: CSS, HTML, Python, Javascript and C), more than 20 paragraphs from random Wikipedia

articles and personal messages/tweets, the tool were able to correctly identify 96% of the code files and 100% of the random Wikipedia paragraphs as 'FORMAL TEXT'. The limitation for this detection consists of 21% of false positives for personal messages, that has 56% accuracy. This is due to two factors: (1) Messages that are more formal and could be labeled as 'FORMAL TEXT'; (2) Tweets that are mainly formed by Twitter hash tags that cannot be read as English, in this scenario they are mistaken as code. In the tool scenario, the False Positive means that there will be some sections that might be wrongly tagged as important, but, the high accuracy for code detection also means that no important moment will be missed mainly considering that the wrong results are as 'FORMAL TEXT' which is also considered "important" in the important sections detection.

To get this result, there are some cleaning to be done in the text. First, it is necessary to remove urls patterns from the text. Urls patterns are formed by non-English words or English words without any delimiter character this structure might cause misidentifications for code. There is also a need to ensure all unicode emojis are separated by the delimiter character, otherwise the library and the list will not match with the ones shown in the text. After the cleaning process, it is necessary to split the gathered text using space as delimiter and get all words from the text. Words in this scenario are defined by a sequence between delimiters that contains only A-Z or a-z characters. Once these steps are finished, the following validations are performed, see Algorithm 3.1:

3.1.2 Interruption detection

As stated in Chapter 2, our tool was designed in a way to not require user actions to detect interruptions and/or important moments. Therefore, we developed a method to check if the developer is working or attending to an interruption. In order to do

Algorithm 3.1 Validations used for text classification

Calculate percentage of emojis and emoticons in the text;
Calculate percentage of programming keywords in the text;
Calculate percentage of stop words in the text;
Calculate percentage of English words in the text;
Calculate percentage of unusual symbols in the text;
if percentage of unusual symbols is higher or equal to 50% **and** percentage of emoticons is less than 10% **then**
    Text can be directly classified as CODE since the percentage of unusual symbols is too high and this is not due to the use of emoticons
end if
if (percentage of stopwords is higher or equal to 28% **or** (percentage of English words is higher or equal to 65% **and** percentage of unusual symbols is lower than 2%) **or** percentage of emoticons is higher than 10%) **and** percentage of programming keywords is less than 30%) **then**
    Text is non-code;
else
    Text is classified as CODE;
end if
if text is non-code **then**
    Get sentiment provided by NLTK - VADER:
    if (neutrality of sentiment is higher than 0.8 OR (neutrality of sentiment is higher than 0.5 **and** sentiment compound is less than 0.6 **and** sentiment compound is higher than -0.6)) **and** percentage of emojis and emoticons is less than 1%) **then**
        The text is classified as FORMAL TEXT;
    else
        Text is classified as INFORMAL TEXT;
    end if
end if

so, we take in consideration two factors:

1. The absence of interactions with the computer;
2. The absence of faces looking directly at the screen.

Since our main focus is code tasks, we believe most important moments to be the ones with most interactions. But in a scenario where one might be engaged in a coding task of reading and/or watching a tutorial video, we use a face detection as a second checker.

After initial testing, we devised the following time-based rules:

1. The system always keeps track of user interactions;
2. After two and a half minutes with no interactions, the system tries to turn the webcam on;
3. It checks for a face looking at the screen for thirty seconds;

Meyer et al. [29] defines an inactive period as a lapse of interactions for two minutes or longer, therefore, we tried to focus on interruptions longer than three minutes. We thought that short interruptions - less than three minutes - would not required a recovery help.

During the face check phase, the system tries to detect a face using Haar Cascade and OpenCv. Viola and Jones [41] proposed a machine learning approach where a cascade function is trained from positive and negative images for object detection. This was later used to generate many pre-trained classifiers for face, eyes, smiles and so on and are currently shipped with OpenCv. In our tool, we use the haarcascade_frontalface_default classifier to detect faces in real time looking at the screen. We also tried to apply the haarcascade_eye classifier to ensure the face would be looking

at the screen, and therefore, both eyes would be detect, and even take developers who use glasses in consideration - which includes the research behind this development - with the use of haarcascade_eye_tree_eyeglasses classifier but the solution made the accuracy to drop.

To improve the detection accuracy, we had to take in consideration the fact that people do not spend an entire day in the same position without moving. The same happens with head. To address that, we rotate the webcam image between -40° and 40°, that according to researchers [24] is the range of movements of a head bending, and check every image rotation against the classifier. After checking it, we add a flag in an array, 'F' if a face was found, 'NF' if no face was found and 'E' if there is any error such as no webcam available. The array is filled during thirty seconds, and every one second a new image is analyzed. This approach is due to two factors, first, the system is not 100% accurate, which means that one might be looking at the screen, and still no face is found. At the same time, the opposite might occur, mainly if we consider the fact that in a company environment it is common to have several developers, some of them might pass in front of the camera and be mistaken for the developer.

To work around possible misidentifications, we try to find a face within thirty seconds, then, we analyze the array looking for the percentage of face found or not found. If there are errors, then, the interruption detection time will jump from three minutes to five minutes disregarding webcam, and only considering all the time with no interactions. If there are no errors, a set of rules shown in Algorithm 3.2 will be applied. It is important to mention that while the system checks the webcam for faces, it keeps checking for interactions. If any interaction is made within the 30 seconds that the webcam is on, the webcam is immediately turned off even if the 30 seconds period is not over.

Algorithm 3.2 Validations used for text classification

```
1: if Face is found with a percentage of higher or equal to 85 then
2:     Interruption process stops;
3:     Tool continues to gather data;
4:     Tool continues to check for interactions;
5: else
6:     if Face is found with a percentage of less than 10% then
7:         An interruption is detected;
8:         The entire tool stops to gather data and starts to process it;
9:     else
10:        An uncertainty is detected;
11:    end if
12: end if
```

Uncertainty is a state that will stop the interruption detection process but it will also store the amount of uncertainties, after three consecutive uncertainty results, an interruption is detected. This state of uncertainty might be due to people crossing, or the design of the office that may have developers right behind others.

We can point that this is not an error-prone approach, one might not have a webcam, and be watching a video and not interacting with the computer and still be detected as interrupted, but, this was the best scenario we could achieve.

3.1.3 Processing data

After all the gathering data and interruption detection processes, the next step to have the tool to present a recovery to the developer, is to process everything that was collected. This process involves the calculation of importance for each moment and the video adjustment.

In order to detect important moments, we came with a score-based approach. It is known as can be seen in Chapter 2 that a task might involve different activities. One single activity might also involve different actions. We believe that scoring actions is a better approach than just a set of rules because we can describe different tasks, with

different actions taking in consideration that each developer has a different behavior.

The scores

Using a score approach, it is possible to consider actions isolated or combined. The score depends on the type of action and how much it is important in our scale of actions in the developer's routine described on Chapter 2.

There are three types of scores:

1. One that varies according to the amount;
2. One that is important to look at the change in behaviour;
3. One that is important to look at the change in behaviour;

For instance, there are developers that tend to save their work after every new line and developers that save only when they need to test it or turn off the computer, thus, if the score for "saving" was only about amount, every coding moment - where normally there is save action - would be more important than all the rest independent of the score value. In order to address this issue, some scores are based in "changes of behaviour" where the score will be processed based in the difference from the developer's average. This average is calculated for each user section, which means, that different tasks can fit in this calculation.

In order to understand the score approach, we first present two formulas to calculate the types of score previously mentioned. The formula for the average-based score is as it follows:

$$average\_score = score\_value * \frac{amount}{averages[feature]}$$

Where the score value is fixed as LOW, MEDIUM or HIGH importance with predefined values 5, 10, 20, respectively, adjusted after tests and user study. The amount is the quantity used, for example, a score for saving, considers the amount of save commands used whithin the section. Averages is an array of calculated averages. We calculate the averages of each score average-based at the beginning of each recovery process. The average considers a sum of all amounts split by all ten seconds windows, which means, that it also considers ten seconds windows with no amounts because this is also an indicator because if there are several ten seconds windows with no amount, any amount will show a change in behavior.

The formula for the amount-based score can be represented as:

$$amount\_based\_score = score\_value * amount$$

Where the score value varies from LOW, MEDIUM and HIGH, having the same predefined values as stated earlier.

These formulas are applied according to the algorithm in Figure 3.2 for each and every app section. All score distribution is based on tests and the idea that code tasks are more important than any other task as explained in Chapter 2. Considering this, a code task might involve typed code, applications used to type code and specific actions such as saving, copying, pasting, searching and so on. All these actions are taken in consideration in the algorithm. Important tasks are then separated in three classes with their own additional scores: Code important: +100; Text important: +40; Non important: -100.

The algorithms

In order to define important moments of a section, it is mandatory to first describe most of the code-related activities. The algorithm we developed tries to include as many activities as possible because different people might have a slightly different approach. For that, we have formulas considering averages that tries to detect change in the normal personal developer behavior. On top of that, our approach considers more interactions as more important scenarios, thus, the scores for each action would accumulate according to the task, creating higher or lower scores.

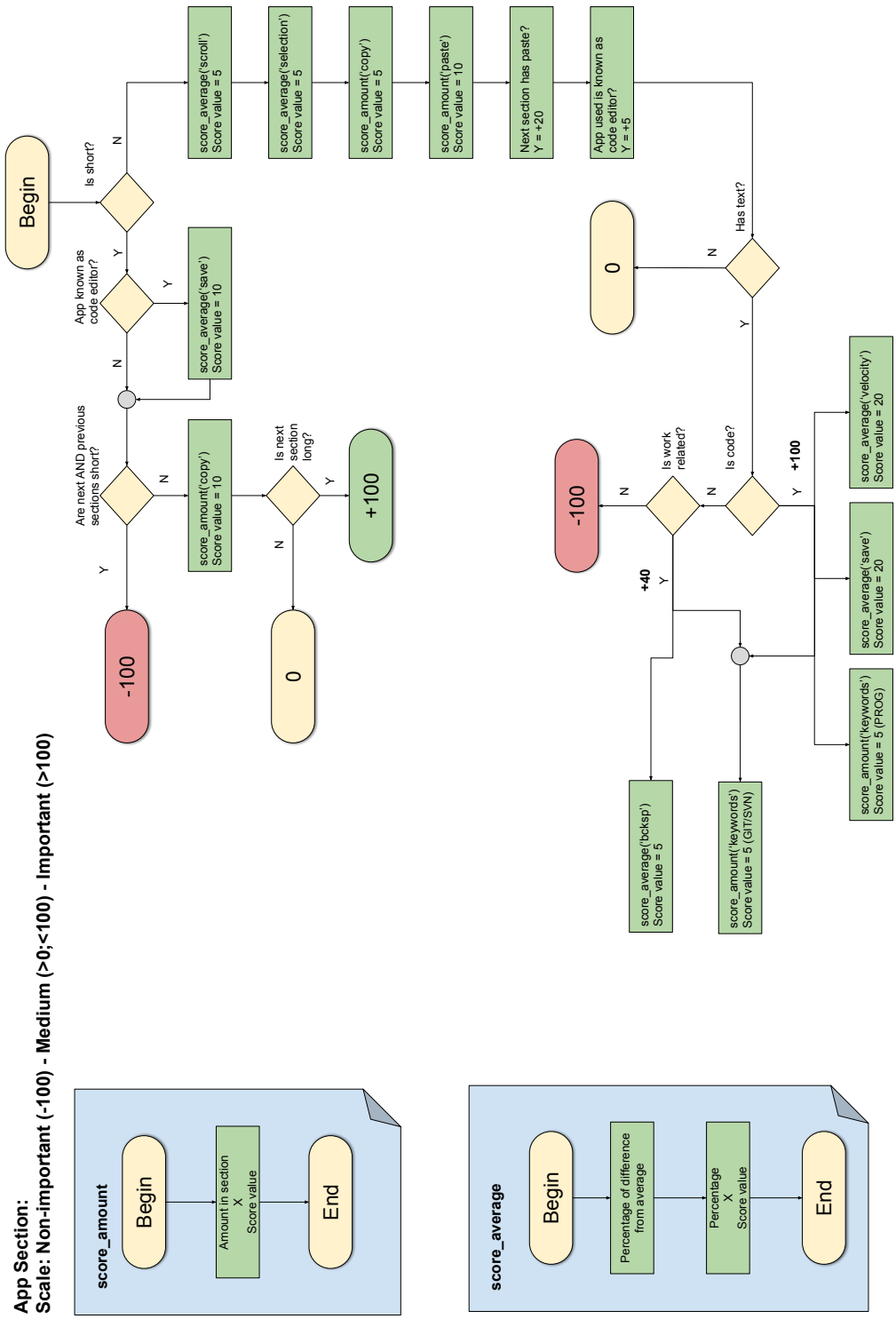

Figure 3.2: Score algorithm for app sections

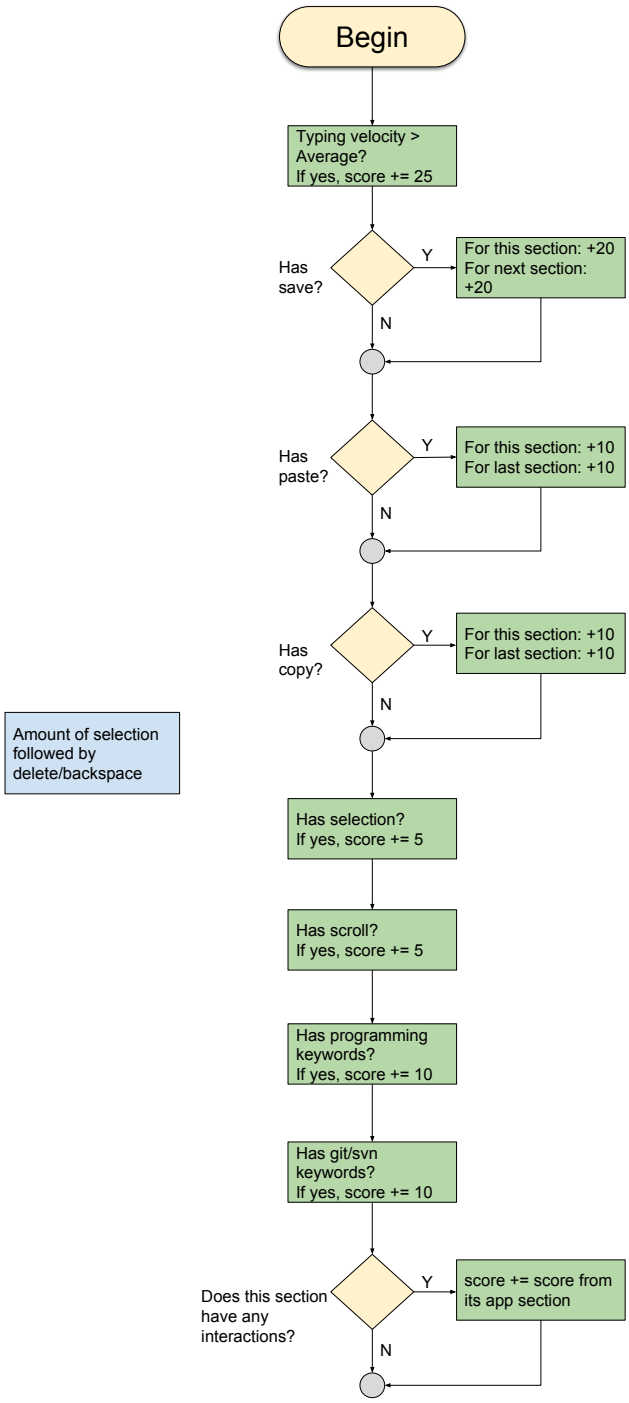

Figure 3.3: Score algorithm for ten seconds windows

After setting a score for the app section, we run algorithm 3.3 to calculate the score for ten second windows. These scores will later be used to defined the playback

rate of each portion of the video. We believe that most recent actions should be more important to recreate context, thus, to ensure that most recent actions will have the correct score, we apply a time function on top of the score for both, app section and ten second windows. This time function is defined as follows:

$$boosted\_score = calculated\_score + (\frac{time\_of\_section}{duration\_of\_user\_section}) \times calculated\_score$$

Where the calculated score is the score before the time boost, time of section is the time where the referred section starts and total duration of user section is the total seconds of the section length. This will ensure that sections closer to the end to the section will have their score boosted. But this also creates an issue, because there might some high scored ten seconds windows far from the end of section, but that, without the boost would be even more important than recent sections. In order to address this, we get the most scored ten second window for each app section, and apply another function that will apply the time boost in reverse, meaning that, at least one ten second window for each application will be able to match, in score, with recent ten second window. These most important ten second windows we call *key moments*.

Our assumption is that, for one to be able to rebuild context, it is necessary not only to see the beginning and ending of a section, but all the path that led to the final section work. To show the entire path, we ensure to show the entire video, and *key moments* where the playback rate will be lower than the other moments in the video.

The video

As mentioned in Chapter 2 videos are proven to have good result in remembering tasks but it can be tedious or annoying to watch a full video of one's working. It can also disturb memory triggering once one might focus only in what the video shows. In order to address these issues, we came with a solution where the video will have only three minutes length - or what the user choose to be, check Subsection 3.1.4 for more details. We believe that three minutes is a fair amount one might be willing to watch without making this task tedious and annoying.

The user section can have hours in length, the showed video can have a minimal length of twenty minutes - or what the user choose to be, check Subsection 3.1.4 for more details - therefore, in order to fit all this amount in three minutes, the tool make some process. The main idea is to change the playback rate of the video according to each section score. But this means that the longer the video the faster the video will be played. There are some technical issues to adjust more than fifty minutes in only three minutes mainly related with the video choppiness that cannot be solved without increasing the three minutes value. We also, assume that developers would not need information from longer time than that - assumption that was evaluated with the user study. Therefore, instead of accelerate a four hours video into three minutes, we first need to trim it the best as possible.

The video is cut at two positions: (1) The part where the system started to receive no interactions from the user - where one might probably be already interrupted. This can correspond up to four minutes and a half. This amount of time is related with the longest function to detect interruption as explained in Subsection 3.1.2. And/Or (2) The video is longer than the maximum value of the range chosen to the length of the video.

This first cut is done in the end of the video. We first find the initial position of

the longest sequence of ten seconds windows scored zero, that means, no interaction. Here, we take in consideration score and not only if there is an interaction, because there are scenarios where a ten second window score might be affected by the previous ten second windows even though, this specific ten seconds has no interactions. We get the first position to make sure to cut off the video the largest part of video.

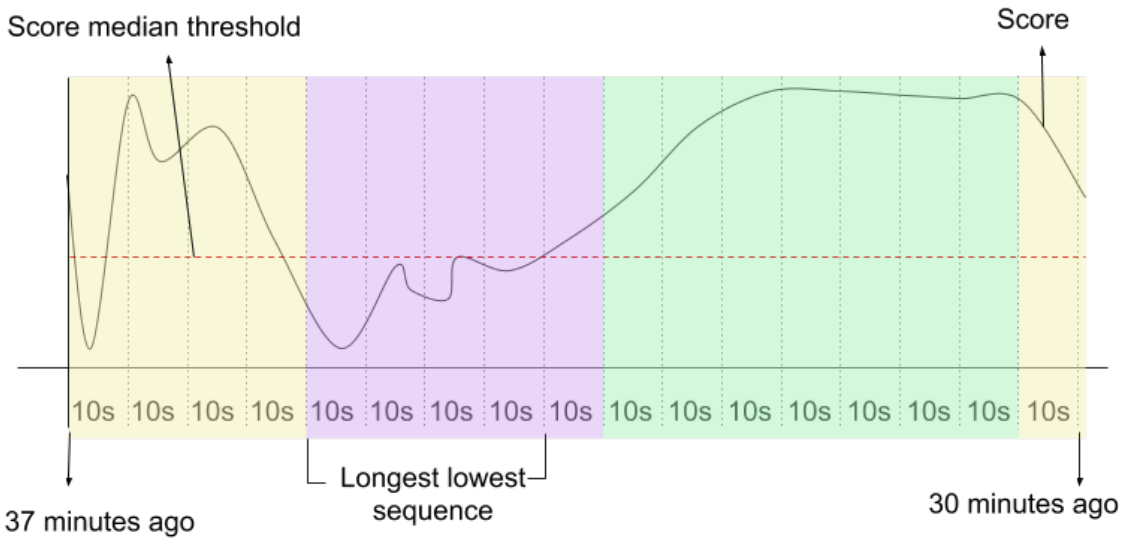

Figure 3.4: Video cut

The second cut is only done in cases where the length of the video, after the first chop, is higher than the maximum value of the range chosen to the length of the video. The default length range for the videos is between thirty and thirty-seven minutes. In this case, we search for the longest lowest score sequence to ensure, that the first seconds of video will have important information or very important information. To determine this, we look to the video length, in reverse, and get all ten seconds window between the length range. We then, use the median of the scores among those ten seconds windows. As can be seen in Figure 3.4, we will find the longest sequence of score below the threshold, one difference in this cut though, is that, since the idea

is to have the shortest video possible inside the length range, we will use the last position of the longest lowest sequence, and get rid of the part of the video - and correspondent ten seconds windows - finished before this position.

3.1.4 Visualization

The visualization we developed is an attempt of showing all the information one might require to go back to work. The idea is to provide all information to enable the developers to rebuild their context.

This screen was developed using wxPython [13] due to its ability to play video without relying on system clocks which precision varies according to platforms and are not high-precise which means that can vary according to the amount of process being executed "at the same time".

Looking at image 3.5, it is possible to observe six main areas: there is (1) the menu where features such as Restart; Exit and Settings can be accessed; (2) the area where the files modified between start and end of the current user section are shown; (3) the are where all the apps used in the current user section along with the real time hour they were used are shown; (4) the area where the video is; (5) the area where the buttons used to control the video - Play, Pause, Stop and Turn speed On/Off - are shown; and (6) the area that contains the timeline;

Menu Area

The first main area varies according to the operational system due to a library limitation - on a Mac, there is a toolbar, on a Windows/Ubuntu Linux there is a File

[13] "wxPython is a cross-platform GUI toolkit for the Python programming language. It allows Python programmers to create programs with a robust, highly functional graphical user interface, simply and easily. It is implemented as a set of Python extension modules that wrap the GUI components of the popular wxWidgets cross platform library, which is written in C++." - https://wxpython.org/pages/overview/ - Last accessed on October 4 2018

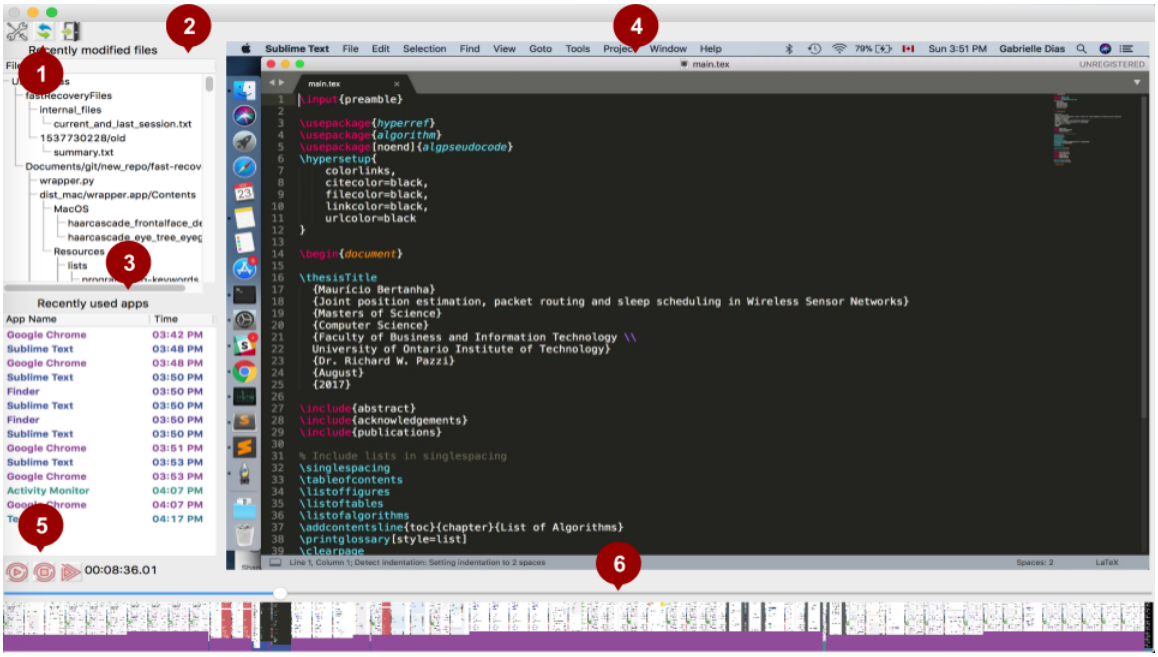

Figure 3.5: Screenshot of the tool on a Mac environment

dropdown menu. The actions though, are exactly the same.

As can be seen in Figure 3.6, the first option is to adjust settings, second option is to restart the tool and the last is used to completely exit the tool. When clicking on "Settings", a dialog opens allowing the developer to adjust three parameters. The first one is about quality level, users can choose between four quality levels - High, Medium-High, Medium-Low and Low. They vary the parameters used with ffmpeg, the High one prioritizes quality, allowing the tool to use more memory; The Medium-High and Medium-Low are similar but the High one still allows a little more memory than the Low one; the Low restricts a lot of memory and CPU consuming, and the final video has a poor quality. To record a screen, even more with all the other data being collected at the same time, is a very memory and CPU consuming task. Since the tool is supposed to run in different computer configurations, the tool allows the user to select what should be a priority, performance or quality. The higher the quality, the more the tool will require from computer's processing, if the user has a

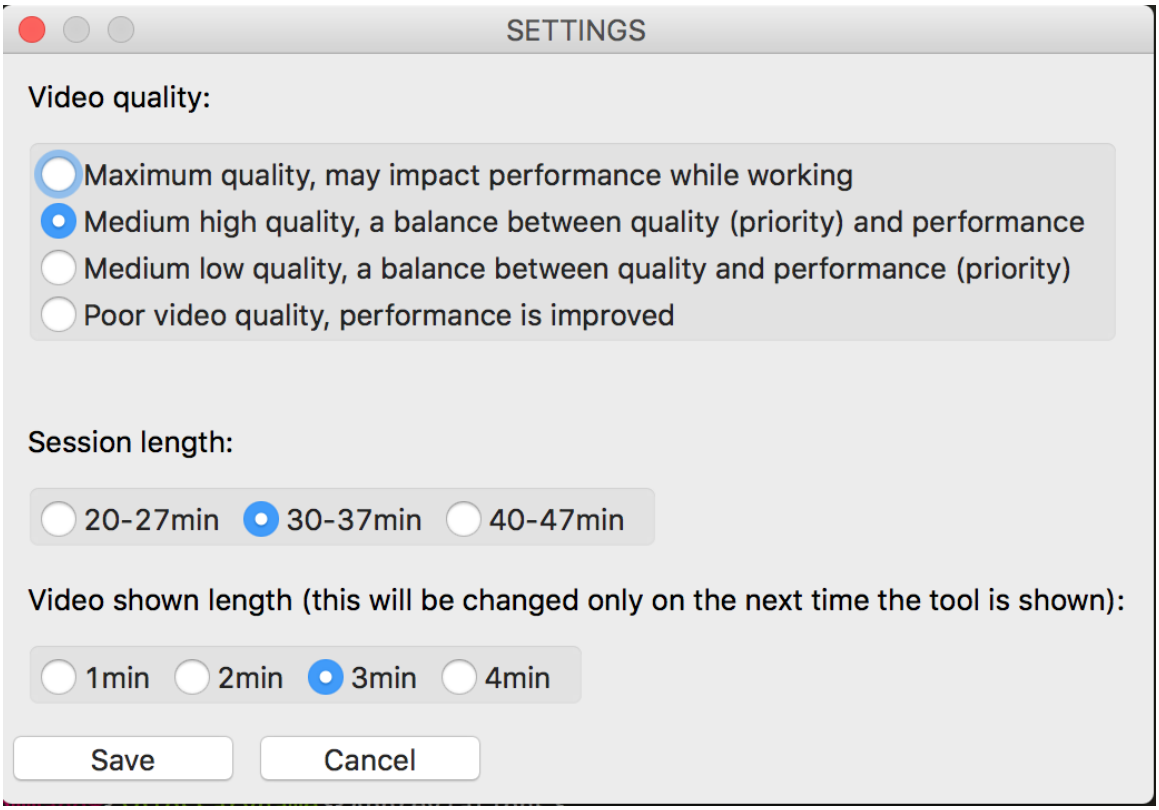

Figure 3.6: Settings screen on a Mac environment

lot of computer power and/or does not mind to have a computer a little slow, he/she can select a better video quality, if otherwise, he/she can also select a poorer one. The default option is medium-high.

The second parameter is about the session length. Users can choose between three options, 20-27, 30-37 and 40-47. This range will affect the amount of content available, the range has a distance of 7 minutes to account for the time for interruption detection. The tool analyses the video and the generated score in order to find the best place to cut the video - and all the information that will not be necessary after the trim. The best moment to cut the video is defined by the last ten seconds window of the longest lowest scored sequence.

The third parameter sets the length of the video displayed. The default length is

3 minutes, but users can choose between 1 and 4 minutes. This number indicates how fast the video will be displayed, the velocity itself if always defined by the score, so, each section will have a different velocity, but the overall velocity will be defined by how much of video - defined by the second parameter - will have to fit in the length defined by this parameter.

Modified Files

The second main area shows all files owned by current user that were modified since the user section beginning. In order to restrict the files shown the tool checks the owner of the file and also its extensions. The extensions allowed are in the compilation made B. The tool also does not considers folders initiated with "." because these folders are commonly modified by the system even though owned by the current user. There is also a list of restricted folders such as"Library" for Mac users and "AppData" for Windows users.

Applications Used

The third main area is a list made with all the applications used in chronological order along with hour and minute that they were used. If an application was used more than once, it will appear several times. The idea behind this area is to help memory triggering with the hour/minute, the order of applications used and a combination of all these factors with the bottom panel timeline. For each application the tool assigns a color that will be also used in the timeline below.

The color used for the text in main area three and for the bars in main area six is obtained through a range of colors starting with a tonality of purple and ending with a tonality of green. Purple and green were chosen to accommodate colorblind [14]

[14] "As many as 8 percent of men and 0.5 percent of women with North-

developers but at the same time have a visible difference between the items in the palette.

Video

Main thread four is where the video is displayed. This video is controlled by the buttons on main area five. There are five possible buttons with only three of them being show. The first button is a "Play/Pause", where "Play" is shown when video is paused and/or stopped and "Pause" while video is playing. Second button is a "Stop" available at any point because when stopping a video the starting point will be put in the beginning, and this could be done with video playing or paused. The last button is an "On/Off" of the important moments feature. When turned on, the video will have different playback rates and therefore different velocity according to the score for each ten seconds windows, when turned off video will be played at normal pace. This feature can be turned on and off at any point with video playing or paused/stopped. Another information displayed in this section is the video position time.

As previously stated the video is not a full replay of the last X minutes of a user section. It is curated according to ten seconds windows scores, this allows the user to spend less time watching the video but still get the context necessary to recover from the interruption. Due to a technical limitation, it is not possible to control precisely the time of each frame exhibition. To address this we make use of the library function of adjust playback rate.

To make use of the score for such things as playback rates or width/height we had to normalize the scores. Note that it is possible to have negative scores. And that when talking about playback rate the logic is reverse. The higher the score the small

ern European ancestry have the common form of red-green color blindness." - https://nei.nih.gov/health/color$_b lindness/facts_about - LastaccessedinOctober2nd$.

the playback rate once high scored sections should be played slower than lower scored ones. The scores are normalized by just finding the minimum and maximum scores throughout all the sections - after the video cropped - and applying the following formula:

$$normalized\_score = (\frac{(ten\_seconds\_window\_score - minimum\_score)}{(maximum\_score - minimum\_score)}) + 1$$

This formula ensures that all scores will be positive and higher than zero. After this, we sum all the new normalized scores to apply to the formula that will calculate the playback rate itself. The playback rate will be applied in a way that all the video will be shown in the amount of time chosen by the user or pre-defined as three minutes. In order to do so, we use the following formulas to first determine the accelerated duration of each ten seconds window and then, use this value to get the playback required to achieve this.

$$accelerate\_duration = \frac{(ten\_seconds\_window\_score \times final\_video\_duration)}{sum\_of\_normalized\_scores}$$

$$playback\_rate = \lfloor \frac{original\_duration}{accelerate\_duration} \rceil$$

We use a round function in the second step to decrease the choppiness of the video. During tests we realized that having playback rates every ten seconds without too much of a difference would not improve the tool but rather make the video hard to watch. We tried to apply a moving average algorithm to reduce choppiness smoothing the scores but it did not help and made the process longer. The easiest way to handle

this issue was to simply round it, this way, less playback rates would be applied to the video and we would still have differences in velocity.

Slider and Thumbnails

Main area six in Figure 3.7 has two components, timeline is a blue bar/slider as any timeline in a media player screen, it shows the position of the video, the width corresponds to the video length, users can drag and drop the cursor at any point to advance or back the video.

Below the slider there is a thumbnails stack. These humbnails are generated by extracting frames from the video using Open Source Computer Vision Library (OpenCV) [15], the frames are extracted only when the screen is built to not overload the machine storing several images.

Figure 3.7: Timeline bar detail

There is some background knowledge used into this thumbnail bar, as said in Chapter 2 there is a research showing that thumbnails helps triggering memory, another research by Deline et. al. [8] shows that it is possible to take advantage of human spatial memory for locating methods. Therefore, human spatial memory might be used to possible help triggering memory mainly due to code syntax colors, text shape or even error colors. There is a correspondence between where the video is, the application being used - represented by the color with match in main area three - and the

[15] "OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code." - https://opencv.org/about.html - Last accessed on October 4 2018

thumbnails allowing users to use the previous mentioned slider to position the video exactly where the thumbnails are shown in the video.

Using this thumbnails bar we wanted to have four different features: (1) Ability to detect important moments; (2) Ability to select a segment; (3) Ability to identify applications used in a specific segment; and (4) Identify aggregated application score.

To address the first feature, we wanted to add a bar chart on the main area six' bottom, but wxPython does not allow overlaying components, which means that the thumbnails could not be overlayed by a bar chart to show importance - score - of each ten seconds window. To overcome this limitation we used size and borders.

To address the second feature, it is possible to click directly in the thumbnail to position the video and timeline on the selected point. In order to do so and also address third feature, it is required to have a correspondence between the timeline that indicates video position and the thumbnail bar, so, some calculations were used.

Considering the features, their need of correspondence between components and taking technical limitations in consideration, the slider will have a minimum and maximum size equal to the available width in the screen. The same way, the slider maximum value will correspond to the video length. For the thumbnails, we will have to simulate a bar chart, to do so, first, we create a Panel - a type of box - which width will correspond to the duration length of the app section using the following formula:

$$\forall_{app} \in_{app\_sections} \rightarrow \frac{(width\_available * (app\_section\_duration + time\_diff))}{duration}$$

Where width_available is the screen width, app_section_duration is the duration of each app section calculated on the gathering information step, time_diff is the difference between the calculated video length and the real video length equally distributed

between the app sections. This difference can occur due to possible delays in processing. The time_diff is used to mitigate discrepancy between the slider - and video - position and thumbnails. Duration is the video length. This formula guarantees that video, slider and app section' boxes will match.

Once we have the box, they are colored according to the correspondent color in the palette. With this step, we defined the "bar chart" bar color. Next step is to fill the panels with the thumbnails. Each thumbnail has three features, height, width and the video position it represents. Height is used to increase or decrease the amount of background - panel - shown, this will create the idea of "bar chart", this means that the most high scored ten second windows will be the smaller thumbnails in height.

For the thumbnails width, there is another process used. Every thumbnail belong to a specific app section this means that there's a limited space for a thumbnails set. The space available is the box width calculated in the previous step. This space is equally split between the ten seconds windows for each app section. The problem with this approach is that we can have very long app sections and although this also means boxes with a higher width, some thumbnails might be very small because of the amount of ten seconds windows inside the app section. To address this issue we developed an algorithm to generate a fair width for each thumbnail.

The initial size of a ten seconds window is defined by:

$$initial\_size = \lceil \frac{s}{amount\_ten\_sec\_windows} \rceil$$

The problem with this initial calculation is that we are only splitting the available size between the amount of ten seconds windows disregarding how this might look like. In a situation where the app section is short, it make sense to show all the ten seconds windows with a thumbnail, but when an app section is longer, the amount of

thumbnails would be large, with not much of a difference between them. This would also make them small or even tiny. To address this, we defined a minimal size for thumbnails. Despite that, it is important to note, that this minimal size only applies to thumbnails inside the app section box. Which means that there can be thumbnails smaller than the minimal in cases where the size of its app section is smaller than the minimal. This is not a bug though, because as previously mentioned, the app section size is time related in order to match the video and slider. But in cases where we have very small widths for the app section boxes and more than one thumbnail to show, we reduce the amount of ten seconds windows to just one, and it will occupy the entire space.

If a thumbnail ends up being smaller than the minimal but its app section is bigger than that, this can only means that we are trying to show more thumbnails than needed. In this case we apply the following formula to get exactly how many thumbnails with minimum width can be fit in the available space:

$$number\_of\_thumbnails = \lfloor \frac{s}{minimal\_thumbnail\_width} \rfloor$$

We make use of flooring function to ensure all thumbnails will fit, even though, there might be a small space to be redistributed. The result of this function will be the real amount of thumbnails shown. After finding this amount, we reapply the first function 3.1.4 replacing the amount of ten seconds windows with the amount found in the previous step. It is important to mentioned, that we also make use of a modulo function to ensure that the shown thumbnails will be distributed across the timeline. The way this function works, is to retrieve the remainder after a division of number1 by number2. When used as a condition inside a loop for example, it will create interval conditions. We will use this, to ensure that only a certain amount of

thumbnails will fit the condition.

In order to correctly generate the modulo condition. We first need to define the "number2" previously mentioned. This is done by:

$$modulo\_condition = \lfloor \frac{amount\_ten\_sec\_windows}{real\_amount\_thumbs\_shown} \rfloor$$

The flooring function will ensure more matches. The modulo function will work as a restrictive condition reducing the amount of valid thumbnails. Since the use of the modulo function will create the intervals, and that only one thumbnail will represent more than one ten seconds windows, we need to first ensure that this will not affect the match between video and thumbnail. As already stated, in a short amount of time, there is not much of a difference between thumbnails whithin the same app section, so, the shown thumbnail is the one that matches the modulo condition. But, to guarantee that the user will not loose too much information neither have issues with the match video-thumbnail, the second in the video will be defined the first video position of the sequence represented by that specific thumbnail. This can vary according to the result obtained in the last step.

For the thumbnails height, there is another technical issue. During the development, we found that the library would prioritize border over height, which means, that depending on the size of the monitor used, the height of the thumbnails could be in a position where no colored bar was seen. To address this, instead of really adjusting height of thumbnail, we would adjust borders according to each score. This approach works, but sometimes the difference between the scores would be minimal, or even too small to make a notable change in the visualization.

To address this and also guarantee that even section with zero score would be seen, we defined a minimal size border and applied a logarithmic function to the values.

The minimal size border defines a step, this way it is possible to always have visual difference between scores. The following formula is used to calculate the size of each border to the thumbnails according to their score. It is important to note that in this situation, the border would be seen more as a padding, since the entire timeline is aligned.

$$chart\_bar\_height = minimal\_border \times \lfloor \frac{(score + |min\_score| + 1) + 1)}{app\_min\_score)} \rceil$$

As can be seen, the tool makes use of the absolute minimum non-normalized score across all ten seconds windows of all applications. The non-normalized score for the specific ten seconds windows. And the minimum normalized score for the specific app section. Ones are added to ensure consistence and error proof due to forbid mathematical operations. This approach also adds values to the score, but since this is done equally throughout the sections, there should be no issue.

3.1.5 Security and Privacy

We understand that having an application that records one screen might have a privacy and security issue. The tool, outside the study, would be used and after the video has been seen, the generated folder should be destroyed, this might solve some of the concerns. We understand that the risk using the tool would be really connected to someone knowing that one uses the tool, knowing the address where the tool stored the video and have access to watch or copy this video in between folders generation and deletion so, it is not a very possible threat. To address some of these concerns the tool was already developed with privacy in mind, which means that we do not store, in text, what is typed. We also do not record images from the webcam, all of

these two process are done on-the-fly. The only concern would be the video, but, as said, a final version of the tool should delete the folder with the video as soon as it get watched.

We also got feedback that having a webcam turned on while one is working it is something that causes some stress. This can be addressed by the users themselves due to the fact that the webcam is a plus not a requirement. If one feels too stressed because the webcam might be turned on, it is possible to cover it with an adhesive or a paper or even to disable it to not have this in use. These actions would cause two minor issues: (1) If the webcam is disabled by software or configuration, the system will take more time to detect an interruption. Every time that there is an error with the webcam, the system will ignore the webcam step, and wait for five minutes without interactions to detect the developer as being interrupted. (2) If the webcam is somehow covered, the system, after two and a half minute without any system interaction, will turn the camera on for thirty seconds and receive as an input a black image and therefore will not find a face. It might then, detect the developer as being interrupted even if she is in fact looking at the screen but watching a video for example that does not need to be interacted with through mouse or keyboard.

Chapter 4

User Study

In order to test the tool efficacy it had to be tested in a scenario more similar to what developers face in their daily routine. The study then, was designed to be done with senior or grad students using the tool in their own office with low intervention from researchers or the environment itself.

The user study allowed us to not only assess the accuracy of the tool and the validity of the hypothesis, but also, to collect information in order to improve the underlying model that identified important moments.

4.1 Study design

The study was designed to mimic a developer's environment, therefore it would not be possible to make a controlled study. To put developers in a closed lab while we watched they work could interfere in there focus and also their interruptions due to the Hawthorne effect [28]. As previously stated there are several variables that might affect how one recovers from an interruption - more details on Chapter 2. We did not want to add another variable into account. The study then, was designed in two

parts:

1) Participants were asked to use the tool as much as possible over three days. During this period participant's feedback were collected on each interruption recovery. In order to avoid impacting the interruption recovery process, the participant's feedback on a given recovery was obtained during the following interruption.

2) Based on the anonymized data from part one, the tool was improved for the second round with same participants. In this phase the procedure was exactly the same. This allowed assessment if modified recovery tool is more accurate than the original. If the modified tool is more accurate, this may indicate that with more refined methods (e.g. machine learning), the developed tool could be further improved.

On the last day of each phase in the study, a semi-structured interview in person was scheduled to talk about the tool. Unless required by the participants all interviews were audio-recorded. Initial questions can be seen in Appendix G other questions were added according to answers or according to some concerns or suggestions expressed by other participants, the idea behind this approach was to detect a pattern that the participant might had forget to mention. In both phases, the user activity inside the tool was also recorded in order to answer the following questions: did they watched the entire video?; Did they have to watch in normal speed parts that the tool had speed-up?; Did they skip some parts that the algorithm considered important?; And, for the parts that they watched, did the weights for the parameters match?

The design of the study also considered the possibility of having extra interviews after the data analysis. We wanted to understand discrepancies between our algorithm and the real use, and therefore, if any were found we would like to talk with the participant to understand why.

The study design was presented and approved by the Research Ethics Board inside the university.

4.1.1 Study Environment

Since we wanted to not interfere in the student's normal environment with all the interruptions that would come with it, the study was conducted were the student wanted to use the tool. Only the interviews were conducted in a lab specially designed for interviews. The use of the tool was controlled by the participant. We asked them to use it as much as possible during 6 days split in 2 periods of 3 days each.

4.1.2 Data Management

During the study we have collected only aggregated data that were not considered sensitive and therefore not offering risks to privacy. But even not offering risks to privacy we anonymized everything collected from the participant's computer. Regarding the student's contact, only one researcher had access to it to send further information about the results of the study. the participant had the right to refuse receiving this email and if so, did not need to provide an email.

We did not have access to the generated video unless requested in the analysis phase, but even then, the participant was asked to be present while we would watch the video in the participant's computer. The participant had the right to refuse to participate in this extra interview, to refuse showing the video entirely or parts of it. The participant would be in control of the play/pause button.

4.2 Participants

Participants were recruited inside the University of Ontario Institute of Technology (UOIT) corpora. We put posters around campus, we sent emails to graduate students mail list and also we Tweeted through the official university account on Twitter. We got 10 answers but only six students confirmed participation. Among these 6 students,

there were 5 male and 1 female; 3 graduate students and 3 third year students from different programming-related courses. All students received 50 Canadian dollars per participation and, in case of withdraw, the participant would receive $25 per phase finalized or nothing if withdrawn before finishing phase one.

4.3 Results

The study was designed to mimic a developer's routine. On top of time and resources constraints to find a company willing to let their employees have their screen recorded, and employees willing to participate, we had the assumption that a senior student could have the same issues with interruptions and therefore could be helped and help us the closest was the way a company would have.

Phase one

The first discovery after the data analysis is that students do not have the same issues with interruption as developers. Although they might face some, it is not in the same frequency as it is in a company and the nature of it is also different. Participants mentioned that almost all of the interruptions they faced were self-interruptions, such as bathroom or snacks. Self-interruptions tend to be easier to recover if we consider that when a developer is aware that s/he might be interrupted soon, there is time to rehearse [2, 39].

Another finding is that students do not have complex - or at least complex enough - projects to be working over three days or even over different sections during reading week - a week inside the university's calendar dedicated to study for projects and exams. All they had were small tasks that could be easily finished in one sitting mainly considering that since they were not attending to class, they could sit for

some hours without interruption. Although the students have the option to go to the university to do studies in group, or in places where they would not be alone - and might cause some distractions or interruptions. They preferred to be at home. Alone. In their room. Which means that there was no background speech noise to distract them, no friends, or colleagues to call them and therefore not much as an interruption, at least not for the first period of study.

Another finding about recruiting students is related with the nature of working. It is not expected from a professional developer to have long distractions throughout the day to watch funny videos, movies or even play games. But this is not a reality for the students. Some of the feedback that we collected is that the tool was not able to catch them as interrupted while they were playing or watching videos. But as stated in Chapter 2, computer-based interruptions are not supported by the tool, not because of a technical issue, but because of its concept. Non-work related activities in a computer is common to happen in a work routine but they should be short. Something that takes more than 5 minutes such as a call from family would take the developer away from the computer. If a developer is watching a video at work, without in anyway interacting with the computer, then, there is an assumption that the video is work related and should not be treated as an interruption.

Some more consistent feedback came from Master Students that, in some ways, have a more similar work with a company developer. There is normally a longer task to work on, they have lab partners and background speech and not by a coincidence were the most positive feedback about the tool being helpful, accurate and that they would use it outside of the study. Some of the undergrad participants with working experience mentioned that they think the tool would be better for the time that they were working at a business environment.

The survey, although requested in the first day interview to be always answered,

was not helpful due to several people just saving it with no feedback or "don't recall" messages - but we kept it for the second round because it is important to understand what they think while they are using the tool. For the second phase, we again explained the participants that answering the survey was required for the study - we received one participant's feedback suggesting the survey to be presented after the recovery process. But, we kept as it was because the real important result with this study is the efficacy of the tool and we didn't want the survey to affect it. Even if it means to not have as much feedback data outside the interviews.

Changes between phases

As a direct result from the received feedback in phase one, we updated the application for phase two. Since 5 out of 6 participants complained about the length of the final video, 2 of them, saying that even 1 minute - minimal configuration available - were too much. We made a new configuration available with only 30 seconds, we also made a new option with only 10 to 17 minutes of content. We were concerned though, because one of the participants mentioned that s/he didn't alter the content length configuration because s/he might need, and once altered, the configuration would be changed only for the next time and the content for the section where s/he really needed would be lost.

With this feedback in mind, we implemented a new on-the-fly settings and the video would not be cut to small portions anymore (see Figure 4.1). We cut the video using our longest configuration just because we still believe - and also from the received feedback we can attest - that more than 40 to 47 minutes of content will not be reviewed. That said, the new version of the application cuts the video with the same logic mentioned in Chapter 3 for a maximum length of 47 minutes. But, when the video is presented, we adjust the starting point according to the selected

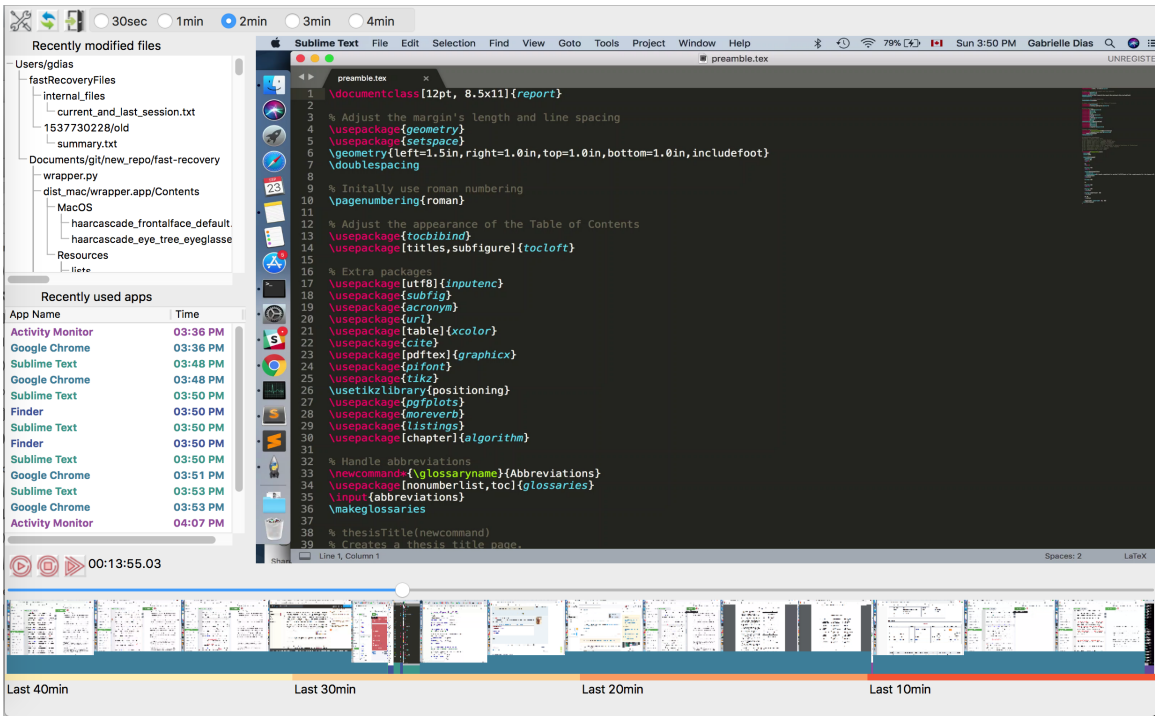

Figure 4.1: Second version of application

configuration. Which means that an user will be able to see more content if needed, but, if not, the selected settings will be applied.

Also, we made available to the user to change the length of the video for the current section. The playback rate will change accordingly not only when a new video length is applied but also, when an user drags the slider, clicks on a thumbnails or clicks on app of other portion of the video outside the original settings. This creates consistence with the selected video length.

In order to make it easy to the user to understand the amount of work being watched, we included a timeline varying from yellow to red, and a subtitle with the time representation for that part of the video as can be seen in Figure 4.2.

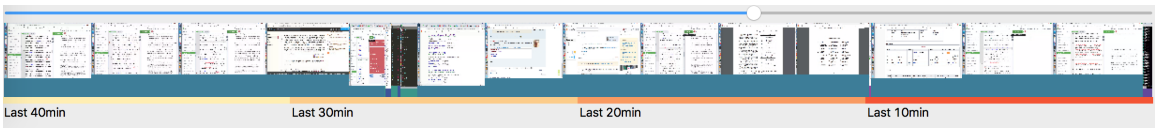

Figure 4.2: Second version of timeline bar detail

Phase two

For the second phase of the study, we had some interesting feedback. One of the participants - although this is not the main focus of the tool - reported that it helped him to improve his overall work by detecting distractions and work on it. He would look at the applications used, the time spent on each one and try to be aware that these were distracting him. This way, he could try and "resist" such forms of interruptions.

One feedback that were presented in some of the participant's feedback in both phases of the study was the ability to tell the tool that "I want to see the video now" or "I am interrupted now". This could not be implemented within the time frame designed between phases. Mainly because, the tool currently uses some external libraries to process videos and build the GUI, these libraries were chosen because they were open source, the most used, the best for our scenario or the only cross-platform in the open source market. But some of the process required by the tool might take several minutes. This is not a problem in the scenario where developers are interrupted and all the processing happens before they get back to their desk to work. But, this become an issue when we talk about having a button to see the video now.

To record someone's screen is a process that requires a lot of CPU as detailed in Chapter 3. On top of that, we have processes checking active processes, capturing keys, mouse movements, analyzing texts and sometimes turning the webcam on and processing the information provided by it. On top of all that, developers cannot have their environment compromised because of a tool that should help them to improve

their work. That said, there is no room to process everything as the work section is happening. And the button requested by the participants, would take up to 4 minutes to make the screen to show up depending on the length of the work section. When asked if they would be willing to wait for it, most of the participants said no. The one participant that said yes, changed his answer for the second phase.

Performance was also a shared concern among the participants. Several of them, when asked if they would use the tool outside the study, said yes, but only if the tool was faster and consuming less resources. Although there might be some space for improving, there are some limitations that forces the use of some time-consuming actions. We could not address them in the period available.

Collected data

Throughout the two phases of the study, we recorded 59 user sections. For phase two we had one student withdrawing from the study what left only 5 active students. We will split the data analysis in two parts: The user section length and the feedback from survey and interviews. It is worth to mention, that one participant had issues with his computer - not related with the tool - and the data was lost, so, for this particular student, we only have one section for phase one although his feedback mentions several.

User Section length

The length of the user section directly affects one's ability to recover from an interruption and might affect the feedback about the tool.

On Table 4.1 it is possible to see a summary from the study. It is interesting to notice that although the highest user section's length per phase decreased, the overall maximum per student increased. We believed that this is a direct result from the difference of working in a reading week and working in a normal routine week. It

is also interesting to note the distribution of user section's length - see Figure 4.3 - to understand how fragmented the work of the students were. One feedback we collected is that since the students knew that the tool was supposed to help with interruption, in phase one - having less 'natural' interruptions - some participants created their own, but on phase two where interruptions were happening, they did not had to create self-interruptions. This not only reflects on longer sections, but also improved the feedback on the tool being helpful maybe due three factors: changes on the tool, longer periods or work and different types of interruption. Since we noticed more engagement from graduate students on phase one, we grouped the user sections accordingly - see Figure 4.4 - and noticed that graduate students have more short sections, mainly between 25 to 35 minutes, but when asked, they mentioned that the interruptions occur naturally.

Sections for phase one and two						
Participants	Phase one			Phase two		
	Amount	Min length	Max length	Amount	Min length	Max length
1	4	3min	1h24min	3	4min	2h02min
2	1	5min	5min	15	4min	2h01min
3	6	3min	57min	7	3min	30min
4	6	3min	54min	5	22min	1h23min
6	5	39min	2h38min	-	-	-
7	1	6min	6min	6	17min	1h02min

Table 4.1: Section length from study

Survey and interview feedback

Participants did not addressed the survey as required. Some said that the issue was remembering the last section in order to answer the survey, some said that they did not know what to answer. There was also an issue with the tool for the first phase of the study where, if the recovery was not addressed the first time but the survey was answered, when re-opening the tool, the survey would be overwritten. From the answered surveys - 15 out of 23 for phase one and 18 out of 36 - we can say

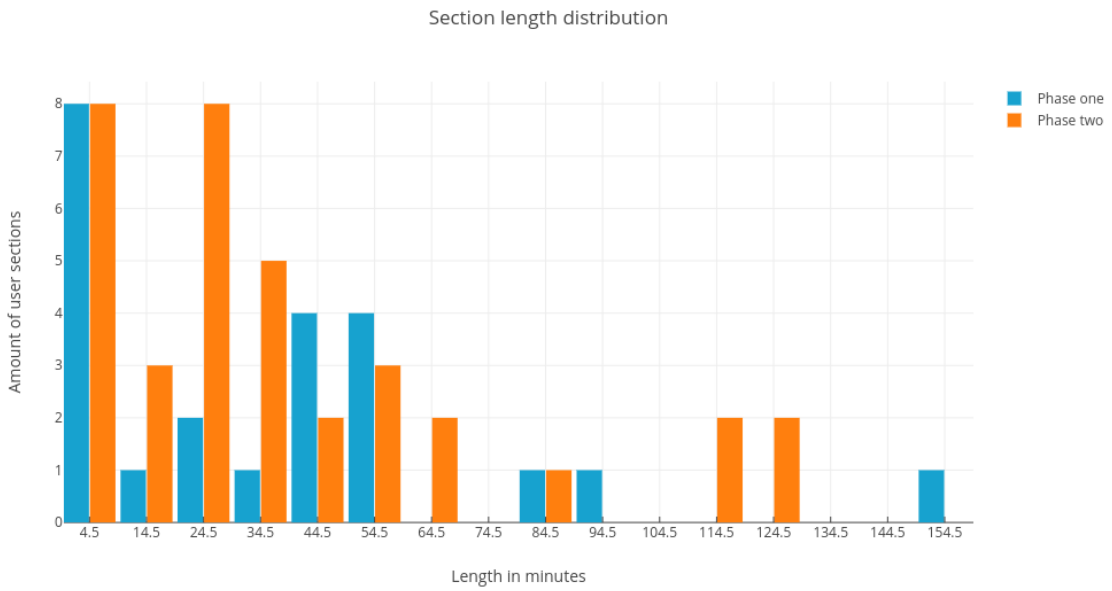

Figure 4.3: User section's length distribution

that for the most of the longer sections the answer for Question 1 *"Thinking about your last interruption, do you think the tool helped you to recover in a faster way¿"* was positive, also, the duration for the interruption itself was also higher for longer sections.

Although, during the interview all participants answered - for both phases - that the tool was easy to use, the data might say something different. We received feedback with suggestions that the features presented in the tool already covered it. When showed the feature in question, the participants then, would say that they did not remember or did not know that they could have done such a thing. One example of this, was a feedback saying that for the last 30 seconds or 1 minute of the section, the video would be too choppy making it hard to the participant to see the exact "last thing typed", his suggestion was to make the last minute of video in real time. The researcher showed the participant the "Turn speed On/Off" and asked if he knew about it, the participant said that he forgot and that this would be very used. We

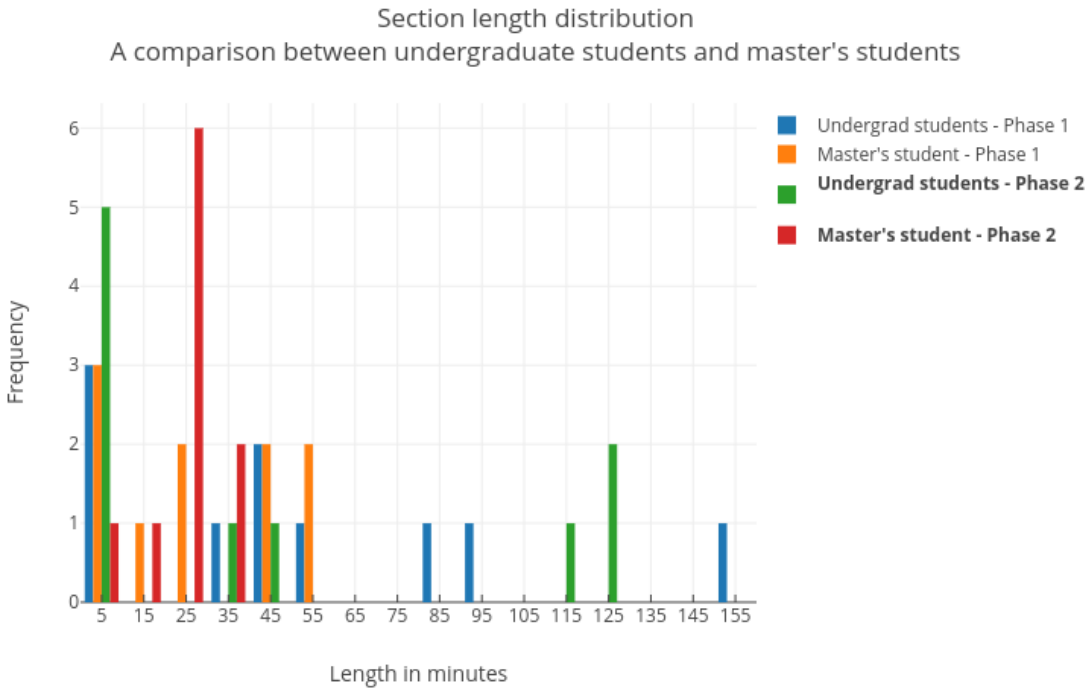

Figure 4.4: User section's length distribution - A comparison

then questioned what he would prefer, a slowed down video by default in the last minute or so, or the button which he replied, the button. Another scenario where we received feedback from already developed features, was due to an issue with some applications on Windows on phase one. The participant said that would be nice if the bars on the bottom had different colors to match if the applications. We showed a working version, which he replied showing us that for him, most of the applications were detected as "Unidentified app" making the bars all the same colors. For phase two we addressed this issue, and in the second phase interview he said that this was one of the best features.

About the important moments detection several participants mention that yes, the importance matched with their expectations, but they said that visually, there could be more of a difference between the heights of the bars. Also, a participant

mentioned, that he spent a lot of time typing - searching - and navigating trough a music player application and the tool was not as fast for this period as he would like, but when looking at the data, the score for this app section was not as high as the code sections, but was not as low as for other non-important activities either.

Chapter 5

Conclusions

5.1 Discussion

This thesis presented FastRecovery, a tool designed to help developers recover from interruptions in a faster way. Based on results collected from more than 50 user sections, it is possible to say that the tool can be definitely improved. Most of the feedback received about adoption of the tool outside the study were tied to some changes in the tool, mainly about performance and the ability to say when one wants to see the video or be detected as interrupted. Considering the target public of this work being developers in companies, we stand by the point that explicit telling the tool that an interruption is happening might not have the same importance as it does for students. A claim like this make a new user study with employees necessary. But we can say for sure that performance can be a real issue for this scenario.

5.2 Limitations

The thesis as a research project has some limitations, some related with the tool itself, and others related with the user study. As mentioned in Chapter 4, the study with students did not match our expectations regarding frequency and type of interruptions neither the complexity or time spent on tasks. We also didn't take in consideration the different mindset of a student compared with a employee. That said, one of the limitations with the work presented is the lack of validation with the real scenario to what the tool was developed to.

As about the limitation for the tool itself, as mentioned in Chapter 3, the tool only supports English speakers, and detects any other language as code. This is a known-limitation mainly if we consider that code is the most important moment for the tool.

There is a limitation regarding the number of monitors supported by the tool. Currently, we only support one due to two factors: the library that records the screen, and the face detection through the use of webcam. If one use two monitors, the webcam might not detect a face, because it will be turned over to the other monitor. The screen record, does not work as expected when using several monitors, currently, it is not possible to choose a "main monitor" neither change the recorded monitor according to the one being active. For a setup using high resolutions monitors the system crashes due to the heavy use of CPU and memory.

Currently, the main focus of this research team was to make the tool run smoothly in two operational systems: Windows and Mac. Mainly due to the availability of users for the user study. The researchers though, tested the tool in some Ubuntu computers, getting some random errors. The tool also does not work for Ubuntu versions lower than 18. For the ones higher, the tool presents some inconsistencies that were not

addressed due to lack of time.

Another point that is fair to be mentioned is a possible bias gender-related. Although the tool was developed with a woman in the team, it is necessary to point the lower number of women in the user study. Even though the user study was open to all students, only a few women contacted us to participate. And since the parameters were adjusted with the participant's feedback, we can't affirm that the result can be generalized.

5.3 Future Work

Although we tried to address the majority of problems we faced during this thesis development, due to time constraint and/or resources constraints there are some future work that can be done.

The first one would be to test it in a real work environment. Get to know how the tool would perform in an environment that would probably be more susceptible to a high frequency of interruptions considering the interaction required in a company in order to make a project to work.

Another point that be very beneficial to the tool would be to try and apply machine learning to really model the developer's behavior and use the data to improve the current model that describes a developer's task. Since the code developed in this thesis is open source, it should be easy to improve, apply machine learning or any other change in the important moments detection module. One just needs to replace the call for the processRecovery thread for their own, and if the generated output is the same, the tool should run as expected collecting data and showing the video the same way.

Lastly, one of the limitation that could be addressed is the support for multiple

languages.

5.4 Conclusions

The data collected in our study allows us to say that the tool might be a good addition to a developer setup, but further adjusts are required in order to make its adoption easier. It is important to mention though, that from the data collected, what helped the recovery process, was not only the curated video but the tool itself, that contains modified files, applications used and thumbnails. Some participants attested that the thumbnails combined with the the modified files were enough to remember the task but it was "nice to have" the video if they wanted to see more details about an specific app, which they in fact did, in some sections.

We also believe that it is crucial to make an user study in a company with developers in the workplace where the tool is supposed to help the most. Although we can project some of the results based on the feedback received, to really attest that the tool helps in the recovery process, it is necessary to test it in a scenario where the interruptions happen in a more frequent and natural way.

It is important to mention also, the difference not only between undergraduate and master's students but also for each individual. Even though the settings might change according to the length of the section or the length of the interruption, some participants, in average, needed more information than others, some participants needed a slower pace video than others. Having the settings available so each one could adjust the tool according to their need was one of the keys for the good feedback received.

Bibliography

[1] Adamczyk, P. D., and Bailey, B. P. If not now, when?: the effects of interruption at different moments within task execution. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2004), ACM, pp. 271–278.

[2] Altmann, E. M., and Trafton, J. G. Task interruption: Resumption lag and the role of cues. Tech. rep., MICHIGAN STATE UNIV EAST LANSING DEPT OF PSYCHOLOGY, 2004.

[3] Bird, S., and Loper, E. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions* (2004), Association for Computational Linguistics, p. 31.

[4] Chong, J., and Siino, R. Interruptions on software teams: a comparison of paired and solo programmers. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work* (2006), ACM, pp. 29–38.

[5] Cutrell, E. B., Czerwinski, M., and Horvitz, E. Effects of instant messaging interruptions on computing tasks. In *CHI'00 extended abstracts on Human factors in computing systems* (2000), ACM, pp. 99–100.

[6] Czerwinski, M., Horvitz, E., and Wilhite, S. A diary study of task switching and interruptions. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2004), ACM, pp. 175–182.

[7] Dabbish, L., Mark, G., and González, V. M. Why do i keep interrupting myself?: environment, habit and self-interruption. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2011), ACM, pp. 3127–3130.

[8] DeLine, R., Czerwinski, M., Meyers, B., Venolia, G., Drucker, S., and Robertson, G. Code thumbnails: Using spatial memory to navigate source code. In *Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on* (2006), IEEE, pp. 11–18.

[9] Gilbert, C. H. E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf* (2014).

[10] González, V. M., and Mark, G. Constant, constant, multi-tasking craziness: managing multiple working spheres. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2004), ACM, pp. 113–120.

[11] Grandhi, S. A., and Jones, Q. Knock, knock! who s there? putting the user in control of managing interruptions. *International Journal of Human-Computer Studies 79* (2015), 35–50.

[12] Gupta, A., Sharda, R., and Greve, R. A. You've got email! does it really matter to process emails now or later? *Information Systems Frontiers 13*, 5 (2011), 637–653.

[13] Hodgetts, H. M., and Jones, D. M. Contextual cues aid recovery from interruption: The role of associative activation. *Journal of Experimental Psychology: Learning, Memory, and Cognition 32*, 5 (2006), 1120.

[14] Iqbal, S. T., and Horvitz, E. Notifications and awareness: a field study of alert usage and preferences. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work* (2010), ACM, pp. 27–30.

[15] Jackson, T., Dawson, R., and Wilson, D. The cost of email interruption. *Journal of Systems and Information Technology 5*, 1 (2001), 81–92.

[16] Jackson, T., Dawson, R., and Wilson, D. Reducing the effect of email interruptions on employees. *International Journal of Information Management 23*, 1 (2003), 55–65.

[17] Jett, Q. R., and George, J. M. Work interrupted: A closer look at the role of interruptions in organizational life. *Academy of management Review 28*, 3 (2003), 494–507.

[18] John, M., Smallman, H. S., and Manes, D. I. Recovery from interruptions to a dynamic monitoring task: The beguiling utility of instant replay. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (2005), vol. 49, SAGE Publications Sage CA: Los Angeles, CA, pp. 473–477.

[19] Kersten, M., Elves, R., and Murphy, G. C. Wysiwyn: Using task focus to ease collaboration. *Supporting the social side of large scale software development* (2006), 19.

[20] Keus van de Poll, M., and Sörqvist, P. Effects of task interruption and background speech on word processed writing. *Applied cognitive psychology 30*, 3 (2016), 430–439.

[21] Labonté, K., Tremblay, S., and Vachon, F. Effects of a warning on interruption recovery in dynamic settings. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (2016), vol. 60, SAGE Publications Sage CA: Los Angeles, CA, pp. 1304–1308.

[22] LaToza, T. D., Venolia, G., and DeLine, R. Maintaining mental models: a study of developer work habits. In *Proceedings of the 28th international conference on Software engineering* (2006), ACM, pp. 492–501.

[23] Lo, R. T.-W., He, B., and Ounis, I. Automatically building a stopword list for an information retrieval system. In *Journal on Digital Information Management: Special Issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR)* (2005), vol. 5, pp. 17–24.

[24] LoPresti, E., Brienza, D. M., Angelo, J., Gilbertson, L., and Sakai, J. Neck range of motion and use of computer head controls. In *Proceedings of the fourth international ACM conference on Assistive technologies* (2000), ACM, pp. 121–128.

[25] Mahar, D., Henderson, R., and Deane, F. The effects of computer anxiety, state anxiety, and computer experience on users' performance of computer based tasks. *Personality and individual differences 22*, 5 (1997), 683–692.

[26] Mark, G., Gudith, D., and Klocke, U. The cost of interrupted work: More speed and stress. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2008), CHI '08, ACM, pp. 107–110.

[27] Mark, G., Iqbal, S., and Czerwinski, M. How blocking distractions affects workplace focus and productivity. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of*

the 2017 ACM International Symposium on Wearable Computers (2017), ACM, pp. 928–934.

[28] McCambridge, J., Witton, J., and Elbourne, D. R. Systematic review of the hawthorne effect: new concepts are needed to study research participation effects. *Journal of clinical epidemiology 67*, 3 (2014), 267–277.

[29] Meyer, A. N., Barton, L. E., Murphy, G. C., Zimmermann, T., and Fritz, T. The work life of developers: Activities, switches and perceived productivity. *IEEE Transactions on Software Engineering 43*, 12 (2017), 1178–1193.

[30] O'Conaill, B., and Frohlich, D. Timespace in the workplace: Dealing with interruptions. In *Conference companion on Human factors in computing systems* (1995), ACM, pp. 262–263.

[31] Parnin, C. Programmer, interrupted. In *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on* (2013), IEEE, pp. 171–172.

[32] Parnin, C., and Rugaber, S. Resumption strategies for interrupted programming tasks. *Software Quality Journal 19*, 1 (2011), 5–34.

[33] Parnin, C., and Rugaber, S. Programmer information needs after memory failure. In *Program Comprehension (ICPC), 2012 IEEE 20th International Conference on* (2012), IEEE, pp. 123–132.

[34] Robertson, T., Prabhakararao, S., Burnett, M., Cook, C., Ruthruff, J. R., Beckwith, L., and Phalgune, A. Impact of interruption style on end-user debugging. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2004), ACM, pp. 287–294.

[35] Rule, A., Tabard, A., and Hollan, J. Using visual histories to reconstruct the mental context of suspended activities. *Human–Computer Interaction 32*, 5-6 (2017), 511–558.

[36] Schlittmeier, S., Hellbrück, J., Thaden, R., and Vorländer, M. The impact of background speech varying in intelligibility: Effects on cognitive performance and perceived disturbance. *Ergonomics 51*, 5 (2008), 719–736.

[37] Singer, J., Lethbridge, T., Vinson, N., and Anquetil, N. An examination of software engineering work practices. In *CASCON First Decade High Impact Papers* (2010), IBM Corp., pp. 174–188.

[38] Sykes, E. R. Interruptions in the workplace: A case study to reduce their effects. *International Journal of Information Management 31*, 4 (2011), 385–394.

[39] Trafton, J. G., Altmann, E. M., Brock, D. P., and Mintz, F. E. Preparing to resume an interrupted task: Effects of prospective goal encoding and retrospective rehearsal. *International Journal of Human-Computer Studies 58*, 5 (2003), 583–603.

[40] van Solingen, R., Berghout, E., and van Latum, F. Interrupts: just a minute never is. *IEEE software 15*, 5 (1998), 97–103.

[41] Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (2001), vol. 1, IEEE, pp. I–I.

Appendix A

List of emoticons compiled from Wikipedia

#https://en.wikipedia.org/wiki/List_of_emoticons -> Western section
[":-)",":)",":-]",":]",":-3",":3",":-
>",":>","8-)","8)",":-}",":}",":o)",":c)",":^)","=]","=)",":-D",":D","
8-D","8D","x-D","xD","X-D","XD","=D","=3","B^D",":-))",":-(",":
(",":-c",":c",":-<",":<",":-[",":[",":-||",">:[",":{",":@",">:
(",":'-(",":'(",":'-)",":')","D-':","D:<","D:","D8","D;","D=","DX",":-
O",":O",":-o",":o",":-0","8-0",">:O",":-
",":",":×",";-)",";)","*-)","*)",";-]",";]",";^)",":-,",";D",":-P","
:P","X-P","XP","x-p","xp",":-p",":p",":-Þ",":Þ",":-þ",":þ",":-b",":b",
"d:","=p",">:P",":-/",":/",":-.",">:\\",">:/",":\\","=/","=\
\",":L","=L",":S",":-|",":|",":
$",":-X",":X",":-#",":#",":-&",":&","O:-)","O:)","0:-3","0:3","0:-)","
0:)","0;^)",">:-)",">:)","}:-)","}:)","3:-)","3:)",">;)",";-)","|
-O",":-J","#-)","%-)","%)",":-###..",":###..","<:-|","',:-|","',:-
l","~(_8^(I)","5:-)","~:-\\","*<|:-)","=:o]","7:^]",",:-)","</
3","<\3","<3","S2","s2","@};-","@}-
>--","@}-;-'---","@>-->--",">:>",">:<*)))-{",">:<(((*>","\o/","*\0/*","//
0-0\\","v.v","O_O","o-o","O_o","o_O","o_o","O-O",">.<","^5","o/
\o",">_>^","^<_<","ϫdarwiɳ⊃","ϫdarwiɳ⊃","ϫevolutioɳ⊃","ϫevolutioɳ⊃","
(ᵔ◡ᵔ)","(>_<)","(>_<)>","(';')","^^ゞ","(^_^;)","(-_-;)","(~_~;)
(・.・;)","(・
_・;)","(・・;)^^;","^_^;","(#^.^#)","(^","^;)","(^.^)y-.o○","(-.-)y-°°
°","(-_-)zzz","(^_-)","(^_-)-☆","((+_+))","(+o+)","(°°)","(°-°) (°.
°)","(°_°)","(°_°>)","(°レ°)","(o|o)","<(｀^
´)>","^_^","(°o°)","(^_^)/","(^O^)／","(^o^)／","(^^)/ (≧∇≦)/","(/◕ヮ
◕)/","(^o^)ノ","∩(·ω·)∩","(·ω·)","^ω^","(__)","_(._.)_","_(_^_)_","<(_
_)> <m(__)m>","m(__)m","m(_ _)m","\(°□\)","(／□°)／","('_')","(/
_;)","(T_T) (;_;)","(;_;","(;_:)","(;O;)","(:_;)","(ToT)","(Τ▽
Τ)",";_;",";-;",";n;",";;","Q.Q","T.T","TnT","QQ","Q_Q","(－
_－)!!","(-.-)","(-_-)","(——)","(; —_
—)","(=_=)","(=^·^=)","(=^··^=)","=_^=","(..)","(._.)","^m^","(・・?",
"(?_?)","(－‸ლ)",">^_^<","<^!^>","^/^","（*^_^*）","§^.^§","(^<^)
(^.^)","(^ム
^)","(^·^)","(^.^)","(^_^.)","(^_^)","(^^)","(^J^)","(*^.^*)","^_^","(
#^.^#)","（^—^）","(^^)/~~~","(^_^)/~","(;_;)/~~~","(^.^)/~~~","(-_-)/
~~~($･･)/~~~","(@^^)/~~~","(T_T)/~~~","(ToT)/~~~","(V)o¥o(V)","\(~o~)
／","\(^o^)／","\(-o-)／ ヽ(^。^)ノ","ヽ(^o^)
ノ","(*^0^*)","(*_*)","(*_*;","(+_+) (@_@)","(@_@。","(@_@;)","\(◎o◎)
/！","!(^^)!","(*^^)v","(^^)v","(^_^)v","（'−'*）（＾ｖ＾）","(＾▽
＾)","(・∀・)","(´∀`)","(⌒▽⌒) ","(~o~)","(~_~)","(^^ゞ","(p_−)","((d[−
_−]b))","(-\"-)","(−−゛)","(^_^メ)","(-_-メ)","(~_~メ)","(−−〆)（・
∧・)","(｀´)","<`～´>","<`∧´>","(——;)","(^0_0^)","(..)φ","φ(..)","(●
^o^●)","(＾ｖ＾)","(＾ｕ＾)","(＾◇＾)
(^)o(^)","(^O^)","(^o^)","(^○^)",")^o^((*^▽^*)","(✿◠‿◠)","(￣ー
￣)","(￣□￣;)","°o°","°O°",":O o_O","o_0","o.O","(o.o)","oO","(*´▽｀
~~~

∗)","(∗°∀°)=3","（ ゜ Д゜)","(°◇°)","(∗￣m￣)","ヽ(´ー｀)┌","¯_(ツ)_/¯","(´･ω･`)","(‘A`)　　","(∗^3^)/~☆","....φ(・∀・∗)",".o○","○o.","(^^)","_U~~","(^^)","_旦~~","☆彡☆ミ","☄",">°)))彡","><∃∃ (°))<<",">°))))彡","<°)))彡",">°))彡","<コ:彡","C:.ミ","~>°)~~~","～°._.°～","(°°)～","●～∗","(ﾉ°□°)ﾉ⌒ ┻┻","┬──┬¯_(ツ)","┻┻⌒ヽ(`Д´)ﾉ⌒ ┻┻","┬┬ノ(º _ ºノ) (ノ◔益◔)ノ彡┻┻",":3ミ"]

Appendix B

List of file extensions compiled from Wikipedia

#https://gist.github.com/aymen-mouelhi/82c93fbcd25f091f2c13faa5e0d61760
[{"name":"ABAP","type":"programming","extensions":[".abap"]},{"name":"AGS Script","type":"programming","extensions":
[".asc",".ash"]},{"name":"AMPL","type":"programming","extensions":[".ampl",".mod"]},
{"name":"ANTLR","type":"programming","extensions":[".g4"]},{"name":"API Blueprint","type":"markup","extensions":[".apib"]},
{"name":"APL","type":"programming","extensions":[".apl",".dyalog"]},{"name":"ASP","type":"programming","extensions":
[".asp",".asax",".ascx",".ashx",".asmx",".aspx",".axd"]},{"name":"ATS","type":"programming","extensions":
[".dats",".hats",".sats"]},{"name":"ActionScript","type":"programming","extensions":[".as"]},
{"name":"Ada","type":"programming","extensions":[".adb",".ada",".ads"]},{"name":"Agda","type":"programming","extensions":
[".agda"]},{"name":"Alloy","type":"programming","extensions":[".als"]},{"name":"Ant Build System","type":"data"},
{"name":"ApacheConf","type":"markup","extensions":[".apacheconf",".vhost"]},{"name":"Apex","type":"programming","extensions":
[".cls"]},{"name":"AppleScript","type":"programming","extensions":[".applescript",".scpt"]},
{"name":"Arc","type":"programming","extensions":[".arc"]},{"name":"Arduino","type":"programming","extensions":[".ino"]},
{"name":"AsciiDoc","type":"prose","extensions":[".asciidoc",".adoc",".asc"]},
{"name":"AspectJ","type":"programming","extensions":[".aj"]},{"name":"Assembly","type":"programming","extensions":
[".asm",".a51",".inc",".nasm"]},{"name":"Augeas","type":"programming","extensions":[".aug"]},
{"name":"AutoHotkey","type":"programming","extensions":[".ahk",".ahkl"]},{"name":"AutoIt","type":"programming","extensions":
[".au3"]},{"name":"Awk","type":"programming","extensions":[".awk",".auk",".gawk",".mawk",".nawk"]},
{"name":"Batchfile","type":"programming","extensions":[".bat",".cmd"]},{"name":"Befunge","type":"programming","extensions":
[".befunge"]},{"name":"Bison","type":"programming","extensions":[".bison"]},{"name":"BitBake","type":"programming","extensions":
[".bb"]},{"name":"BlitzBasic","type":"programming","extensions":[".bb",".decls"]},
{"name":"BlitzMax","type":"programming","extensions":[".bmx"]},{"name":"Bluespec","type":"programming","extensions":[".bsv"]},
{"name":"Boo","type":"programming","extensions":[".boo"]},{"name":"Brainfuck","type":"programming","extensions":[".b",".bf"]},
{"name":"Brightscript","type":"programming","extensions":[".brs"]},{"name":"Bro","type":"programming","extensions":[".bro"]},
{"name":"C","type":"programming","extensions":[".c",".cats",".h",".idc",".w"]},{"name":"C#","type":"programming","extensions":
[".cs",".cake",".cshtml",".csx"]},{"name":"C++","type":"programming","extensions":[".cpp",".c++",".cc",".cp",".cxx",".h",".h+
+",".hh",".hpp",".hxx",".inc",".inl",".ipp",".tcc",".tpp"]},{"name":"C-ObjDump","type":"data","extensions":[".c-objdump"]},
{"name":"C2hs Haskell","type":"programming","extensions":[".chs"]},{"name":"CLIPS","type":"programming","extensions":[".clp"]},
{"name":"CMake","type":"programming","extensions":[".cmake",".cmake.in"]},{"name":"COBOL","type":"programming","extensions":
[".cob",".cbl",".ccp",".cobol",".cpy"]},{"name":"CSS","type":"markup","extensions":[".css"]},
{"name":"CSV","type":"data","extensions":[".csv"]},{"name":"Cap'n Proto","type":"programming","extensions":[".capnp"]},
{"name":"CartoCSS","type":"programming","extensions":[".mss"]},{"name":"Ceylon","type":"programming","extensions":[".ceylon"]},
{"name":"Chapel","type":"programming","extensions":[".chpl"]},{"name":"Charity","type":"programming","extensions":[".ch"]},
{"name":"ChucK","type":"programming","extensions":[".ck"]},{"name":"Cirru","type":"programming","extensions":[".cirru"]},
{"name":"Clarion","type":"programming","extensions":[".clw"]},{"name":"Clean","type":"programming","extensions":
[".icl",".dcl"]},{"name":"Click","type":"programming","extensions":[".click"]},
{"name":"Clojure","type":"programming","extensions":
[".clj",".boot",".cl2",".cljc",".cljs",".cljs.hl",".cljscm",".cljx",".hic"]},
{"name":"CoffeeScript","type":"programming","extensions":[".coffee","._coffee",".cake",".cjsx",".cson",".iced"]},
{"name":"ColdFusion","type":"programming","extensions":[".cfm",".cfml"]},{"name":"ColdFusion
CFC","type":"programming","extensions":[".cfc"]},{"name":"Common Lisp","type":"programming","extensions":
[".lisp",".asd",".cl",".l",".lsp",".ny",".podsl",".sexp"]},{"name":"Component Pascal","type":"programming","extensions":
[".cp",".cps"]},{"name":"Cool","type":"programming","extensions":[".cl"]},{"name":"Coq","type":"programming","extensions":
[".coq",".v"]},{"name":"Cpp-ObjDump","type":"data","extensions":[".cppobjdump",".c++-objdump",".c++objdump",".cpp-
objdump",".cxx-objdump"]},{"name":"Creole","type":"prose","extensions":[".creole"]},
{"name":"Crystal","type":"programming","extensions":[".cr"]},{"name":"Cucumber","type":"programming","extensions":[".feature"]},
{"name":"Cuda","type":"programming","extensions":[".cu",".cuh"]},{"name":"Cycript","type":"programming","extensions":[".cy"]},
{"name":"Cython","type":"programming","extensions":[".pyx",".pxd",".pxi"]},{"name":"D","type":"programming","extensions":
[".d",".di"]},{"name":"D-ObjDump","type":"data","extensions":[".d-objdump"]},{"name":"DIGITAL Command
Language","type":"programming","extensions":[".com"]},{"name":"DM","type":"programming","extensions":[".dm"]},{"name":"DNS
Zone","type":"data","extensions":[".zone",".arpa"]},{"name":"DTrace","type":"programming","extensions":[".d"]},{"name":"Darcs
Patch","type":"data","extensions":[".darcspatch",".dpatch"]},{"name":"Dart","type":"programming","extensions":[".dart"]},
{"name":"Diff","type":"data","extensions":[".diff",".patch"]},{"name":"Dockerfile","type":"data","extensions":[".dockerfile"]},
{"name":"Dogescript","type":"programming","extensions":[".djs"]},{"name":"Dylan","type":"programming","extensions":
[".dylan",".dyl",".intr",".lid"]},{"name":"E","type":"programming","extensions":[".E"]},
{"name":"ECL","type":"programming","extensions":[".ecl",".eclxml"]},{"name":"ECLiPSe","type":"programming","extensions":
[".ecl"]},{"name":"Eagle","type":"markup","extensions":[".sch",".brd"]},{"name":"Ecere Projects","type":"data","extensions":
[".epj"]},{"name":"Eiffel","type":"programming","extensions":[".e"]},{"name":"Elixir","type":"programming","extensions":
[".ex",".exs"]},{"name":"Elm","type":"programming","extensions":[".elm"]},{"name":"Emacs
Lisp","type":"programming","extensions":[".el",".emacs",".emacs.desktop"]},
{"name":"EmberScript","type":"programming","extensions":[".em",".emberscript"]},
{"name":"Erlang","type":"programming","extensions":[".erl",".es",".escript",".hrl",".xrl",".yrl"]},
{"name":"F#","type":"programming","extensions":[".fs",".fsi",".fsx"]},{"name":"FLUX","type":"programming","extensions":
[".fx",".flux"]},{"name":"FORTRAN","type":"programming","extensions":[".f90",".f",".f03",".f08",".f77",".f95",".for",".fpp"]},
{"name":"Factor","type":"programming","extensions":[".factor"]},{"name":"Fancy","type":"programming","extensions":
[".fy",".fancypack"]},{"name":"Fantom","type":"programming","extensions":[".fan"]},
{"name":"Filterscript","type":"programming","extensions":[".fs"]},{"name":"Formatted","type":"data","extensions":
[".for",".eam.fs"]},{"name":"Forth","type":"programming","extensions":[".fth",".4th",".f",".for",".forth",".fr",".frt",".fs"]},
{"name":"FreeMarker","type":"programming","extensions":[".ftl"]},{"name":"Frege","type":"programming","extensions":[".fr"]},
{"name":"G-code","type":"data","extensions":[".g",".gco",".gcode"]},{"name":"GAMS","type":"programming","extensions":[".gms"]},
{"name":"GAP","type":"programming","extensions":[".g",".gap",".gd",".gi",".tst"]},
{"name":"GAS","type":"programming","extensions":[".s",".ms"]},{"name":"GDScript","type":"programming","extensions":[".gd"]},
{"name":"GLSL","type":"programming","extensions":
[".glsl",".fp",".frag",".frg",".fs",".fsh",".fshader",".geo",".geom",".glslv",".gshader",".shader",".vert",".vrx",".vsh",".vshad
er"]},{"name":"Game Maker Language","type":"programming","extensions":[".gml"]},
{"name":"Genshi","type":"programming","extensions":[".kid"]},{"name":"Gentoo Ebuild","type":"programming","extensions":
[".ebuild"]},{"name":"Gentoo Eclass","type":"programming","extensions":[".eclass"]},{"name":"Gettext
Catalog","type":"prose","extensions":[".po",".pot"]},{"name":"Glyph","type":"programming","extensions":[".glf"]},
{"name":"Gnuplot","type":"programming","extensions":[".gp",".gnu",".gnuplot",".plot",".plt"]},
{"name":"Go","type":"programming","extensions":[".go"]},{"name":"Golo","type":"programming","extensions":[".golo"]},
{"name":"Gosu","type":"programming","extensions":[".gs",".gst",".gsx",".vark"]},
{"name":"Grace","type":"programming","extensions":[".grace"]},{"name":"Gradle","type":"data","extensions":[".gradle"]},
{"name":"Grammatical Framework","type":"programming","extensions":[".gf"]},{"name":"Graph Modeling
Language","type":"data","extensions":[".gml"]},{"name":"GraphQL","type":"data","extensions":[".graphql"]},{"name":"Graphviz
(DOT)","type":"data","extensions":[".dot",".gv"]},{"name":"Groff","type":"markup","extensions":[".man",".1",".1in",".1m",".
1x",".2",".3",".3in",".3m",".3qt",".3x",".4",".5",".6",".7",".8",".9",".l",".me",".ms",".n",".rno",".roff"]},
{"name":"Groovy","type":"programming","extensions":[".groovy",".grt",".gtpl",".gvy"]},{"name":"Groovy Server
Pages","type":"programming","extensions":[".gsp"]},{"name":"HCL","type":"programming","extensions":[".hcl",".tf"]},
{"name":"HLSL","type":"programming","extensions":[".hlsl",".fx",".fxh",".hlsli"]},{"name":"HTML","type":"markup","extensions":
[".html",".htm",".html.hl",".inc",".st",".xht",".xhtml"]},{"name":"HTML+Django","type":"markup","extensions":
[".mustache",".jinja"]},{"name":"HTML+EEX","type":"markup","extensions":[".eex"]},
{"name":"HTML+ERB","type":"markup","extensions":[".erb",".erb.deface"]},{"name":"HTML+PHP","type":"markup","extensions":
[".phtml"]},{"name":"HTTP","type":"data","extensions":[".http"]},{"name":"Hack","type":"programming","extensions":
[".hh",".php"]},{"name":"Haml","type":"markup","extensions":[".haml",".haml.deface"]},
{"name":"Handlebars","type":"markup","extensions":[".handlebars",".hbs"]},{"name":"Harbour","type":"programming","extensions":
[".hb"]},{"name":"Haskell","type":"programming","extensions":[".hs",".hsc"]},{"name":"Haxe","type":"programming","extensions":
[".hx",".hxsl"]},{"name":"Hy","type":"programming","extensions":[".hy"]},{"name":"HyPhy","type":"programming","extensions":
[".bf"]},{"name":"IDL","type":"programming","extensions":[".pro",".dlm"]},{"name":"IGOR Pro","type":"programming","extensions":
[".ipf"]},{"name":"INI","type":"data","extensions":[".ini",".cfg",".prefs",".pro",".properties"]},{"name":"IRC
log","type":"data","extensions":[".irclog",".weechatlog"]},{"name":"Idris","type":"programming","extensions":[".idr",".lidr"]},
{"name":"Inform 7","type":"programming","extensions":[".ni",".i7x"]},{"name":"Inno Setup","type":"programming","extensions":
[".iss"]},{"name":"Io","type":"programming","extensions":[".io"]},{"name":"Ioke","type":"programming","extensions":[".ik"]},
{"name":"Isabelle","type":"programming","extensions":[".thy"]},{"name":"Isabelle ROOT","type":"programming"},
{"name":"J","type":"programming","extensions":[".ijs"]},{"name":"JFlex","type":"programming","extensions":[".flex",".jflex"]},
{"name":"JSON","type":"data","extensions":[".json",".geojson",".lock",".topojson"]},{"name":"JSON5","type":"data","extensions":
[".json5"]},{"name":"JSONLD","type":"data","extensions":[".jsonld"]},{"name":"JSONiq","type":"programming","extensions":
[".jq"]},{"name":"JSX","type":"programming","extensions":[".jsx"]},{"name":"Jade","type":"markup","extensions":[".jade"]},

```
{"name":"Jasmin","type":"programming","extensions":[".j"]},{"name":"Java","type":"programming","extensions":[".java"]},
{"name":"Java Server Pages","type":"programming","extensions":[".jsp"]},{"name":"JavaScript","type":"programming","extensions":
[".js","._js",".bones",".es",".es6",".frag",".gs",".jake",".jsb",".jscad",".jsfl",".jsm",".jss",".njs",".pac",".sjs",".ssjs",".s
ublime-build",".sublime-commands",".sublime-completions",".sublime-keymap",".sublime-macro",".sublime-menu",".sublime-
mousemap",".sublime-project",".sublime-settings",".sublime-theme",".sublime-
workspace",".sublime_metrics",".sublime_session",".xsjs",".xsjslib"]},{"name":"Julia","type":"programming","extensions":
[".jl"]},{"name":"Jupyter Notebook","type":"markup","extensions":[".ipynb"]},{"name":"KRL","type":"programming","extensions":
[".krl"]},{"name":"KiCad","type":"programming","extensions":[".sch",".brd",".kicad_pcb"]},
{"name":"Kit","type":"markup","extensions":[".kit"]},{"name":"Kotlin","type":"programming","extensions":[".kt",".ktm",".kts"]},
{"name":"LFE","type":"programming","extensions":[".lfe"]},{"name":"LLVM","type":"programming","extensions":[".ll"]},
{"name":"LOLCODE","type":"programming","extensions":[".lol"]},{"name":"LSL","type":"programming","extensions":[".lsl",".lslp"]},
{"name":"LabVIEW","type":"programming","extensions":[".lvproj"]},{"name":"Lasso","type":"programming","extensions":
[".lasso",".las",".lasso8",".lasso9",".ldml"]},{"name":"Latte","type":"markup","extensions":[".latte"]},
{"name":"Lean","type":"programming","extensions":[".lean",".hlean"]},{"name":"Less","type":"markup","extensions":[".less"]},
{"name":"Lex","type":"programming","extensions":[".l",".lex"]},{"name":"LilyPond","type":"programming","extensions":
[".ly",".ily"]},{"name":"Limbo","type":"programming","extensions":[".b",".m"]},{"name":"Linker
Script","type":"data","extensions":[".ld",".lds"]},{"name":"Linux Kernel Module","type":"data","extensions":[".mod"]},
{"name":"Liquid","type":"markup","extensions":[".liquid"]},{"name":"Literate Agda","type":"programming","extensions":
[".lagda"]},{"name":"Literate CoffeeScript","type":"programming","extensions":[".litcoffee"]},{"name":"Literate
Haskell","type":"programming","extensions":[".lhs"]},{"name":"LiveScript","type":"programming","extensions":[".ls","._ls"]},
{"name":"Logos","type":"programming","extensions":[".xm",".x",".xi"]},{"name":"Logtalk","type":"programming","extensions":
[".lgt",".logtalk"]},{"name":"LookML","type":"programming","extensions":[".lookml"]},
{"name":"LoomScript","type":"programming","extensions":[".ls"]},{"name":"Lua","type":"programming","extensions":
[".lua",".fcgi",".nse",".pd_lua",".rbxs",".wlua"]},{"name":"M","type":"programming","extensions":[".mumps",".m"]},
{"name":"M4","type":"programming","extensions":[".m4"]},{"name":"M4Sugar","type":"programming","extensions":[".m4"]},
{"name":"MAXScript","type":"programming","extensions":[".ms",".mcr"]},{"name":"MTML","type":"markup","extensions":[".mtml"]},
{"name":"MUF","type":"programming","extensions":[".muf",".m"]},{"name":"Makefile","type":"programming","extensions":
[".mak",".d",".mk",".mkfile"]},{"name":"Mako","type":"programming","extensions":[".mako",".mao"]},
{"name":"Markdown","type":"prose","extensions":[".md",".markdown",".mkd",".mkdn",".mkdown",".ron"]},
{"name":"Mask","type":"markup","extensions":[".mask"]},{"name":"Mathematica","type":"programming","extensions":
[".mathematica",".cdf",".m",".ma",".mt",".nb",".nbp",".wl",".wlt"]},{"name":"Matlab","type":"programming","extensions":
[".matlab",".m"]},{"name":"Maven POM","type":"data"},{"name":"Max","type":"programming","extensions":
[".maxpat",".maxhelp",".maxproj",".mxt",".pat"]},{"name":"MediaWiki","type":"prose","extensions":[".mediawiki",".wiki"]},
{"name":"Mercury","type":"programming","extensions":[".m",".moo"]},{"name":"Metal","type":"programming","extensions":
[".metal"]},{"name":"MiniD","type":"programming","extensions":[".minid"]},{"name":"Mirah","type":"programming","extensions":
[".druby",".duby",".mir",".mirah"]},{"name":"Modelica","type":"programming","extensions":[".mo"]},
{"name":"Modula-2","type":"programming","extensions":[".mod"]},{"name":"Module Management
System","type":"programming","extensions":[".mms",".mmk"]},{"name":"Monkey","type":"programming","extensions":[".monkey"]},
{"name":"Moocode","type":"programming","extensions":[".moo"]},{"name":"MoonScript","type":"programming","extensions":[".moon"]},
{"name":"Myghty","type":"programming","extensions":[".myt"]},{"name":"NCL","type":"programming","extensions":[".ncl"]},
{"name":"NL","type":"data","extensions":[".nl"]},{"name":"NSIS","type":"programming","extensions":[".nsi",".nsh"]},
{"name":"Nemerle","type":"programming","extensions":[".n"]},{"name":"NetLinx","type":"programming","extensions":
[".axs",".axi"]},{"name":"NetLinx+ERB","type":"programming","extensions":[".axs.erb",".axi.erb"]},
{"name":"NetLogo","type":"programming","extensions":[".nlogo"]},{"name":"NewLisp","type":"programming","extensions":
[".nl",".lisp",".lsp"]},{"name":"Nginx","type":"markup","extensions":[".nginxconf",".vhost"]},
{"name":"Nimrod","type":"programming","extensions":[".nim",".nimrod"]},{"name":"Ninja","type":"data","extensions":[".ninja"]},
{"name":"Nit","type":"programming","extensions":[".nit"]},{"name":"Nix","type":"programming","extensions":[".nix"]},
{"name":"Nu","type":"programming","extensions":[".nu"]},{"name":"NumPy","type":"programming","extensions":
[".numpy",".numpyw",".numsc"]},{"name":"OCaml","type":"programming","extensions":
[".ml",".eliom",".eliomi",".ml4",".mli",".mll",".mly"]},{"name":"ObjDump","type":"data","extensions":[".objdump"]},
{"name":"Objective-C","type":"programming","extensions":[".m",".h"]},{"name":"Objective-C++","type":"programming","extensions":
[".mm"]},{"name":"Objective-J","type":"programming","extensions":[".j",".sj"]},
{"name":"Omgrofl","type":"programming","extensions":[".omgrofl"]},{"name":"Opa","type":"programming","extensions":[".opa"]},
{"name":"Opal","type":"programming","extensions":[".opal"]},{"name":"OpenCL","type":"programming","extensions":
[".cl",".opencl"]},{"name":"OpenEdge ABL","type":"programming","extensions":[".p",".cls"]},
{"name":"OpenSCAD","type":"programming","extensions":[".scad"]},{"name":"Org","type":"prose","extensions":[".org"]},
{"name":"Ox","type":"programming","extensions":[".ox",".oxh",".oxo"]},{"name":"Oxygene","type":"programming","extensions":
[".oxygene"]},{"name":"Oz","type":"programming","extensions":[".oz"]},{"name":"PAWN","type":"programming","extensions":
[".pwn",".inc"]},{"name":"PHP","type":"programming","extensions":
[".php",".aw",".ctp",".fcgi",".inc",".php3",".php4",".php5",".phps",".phpt"]},{"name":"PLSQL","type":"programming","extensions":
[".pls",".pck",".pkb",".pks",".plb",".plsql",".sql"]},{"name":"PLpgSQL","type":"programming","extensions":[".sql"]},
{"name":"POV-Ray SDL","type":"programming","extensions":[".pov",".inc"]},{"name":"Pan","type":"programming","extensions":
[".pan"]},{"name":"Papyrus","type":"programming","extensions":[".psc"]},{"name":"Parrot","type":"programming","extensions":
[".parrot"]},{"name":"Parrot Assembly","type":"programming","extensions":[".pasm"]},{"name":"Parrot Internal
Representation","type":"programming","extensions":[".pir"]},{"name":"Pascal","type":"programming","extensions":
[".pas",".dfm",".dpr",".inc",".lpr",".pp"]},{"name":"Perl","type":"programming","extensions":
[".pl",".al",".cgi",".fcgi",".perl",".ph",".plx",".pm",".pod",".psgi",".t"]},{"name":"Perl6","type":"programming","extensions":
[".6pl",".6pm",".nqp",".p6",".p6l",".p6m",".pl",".pl6",".pm",".pm6",".t"]},{"name":"Pickle","type":"data","extensions":
[".pkl"]},{"name":"PicoLisp","type":"programming","extensions":[".l"]},{"name":"PigLatin","type":"programming","extensions":
[".pig"]},{"name":"Pike","type":"programming","extensions":[".pike",".pmod"]},{"name":"Pod","type":"prose","extensions":
[".pod"]},{"name":"PogoScript","type":"programming","extensions":[".pogo"]},{"name":"Pony","type":"programming","extensions":
[".pony"]},{"name":"PostScript","type":"markup","extensions":[".ps",".eps"]},
{"name":"PowerShell","type":"programming","extensions":[".ps1",".psd1",".psm1"]},
{"name":"Processing","type":"programming","extensions":[".pde"]},{"name":"Prolog","type":"programming","extensions":
[".pl",".pro",".prolog",".yap"]},{"name":"Propeller Spin","type":"programming","extensions":[".spin"]},{"name":"Protocol
Buffer","type":"markup","extensions":[".proto"]},{"name":"Public Key","type":"data","extensions":[".asc",".pub"]},
{"name":"Puppet","type":"programming","extensions":[".pp"]},{"name":"Pure Data","type":"programming","extensions":[".pd"]},
{"name":"PureBasic","type":"programming","extensions":[".pb",".pbi"]},{"name":"PureScript","type":"programming","extensions":
[".purs"]},{"name":"Python","type":"programming","extensions":
[".py",".bzl",".cgi",".fcgi",".gyp",".lmi",".pyde",".pyp",".pyt",".pyw",".rpy",".tac",".wsgi",".xpy"]},{"name":"Python
traceback","type":"data","extensions":[".pytb"]},{"name":"QML","type":"programming","extensions":[".qml",".qbs"]},
{"name":"QMake","type":"programming","extensions":[".pro",".pri"]},{"name":"R","type":"programming","extensions":
[".r",".rd",".rsx"]},{"name":"RAML","type":"markup","extensions":[".raml"]},{"name":"RDoc","type":"prose","extensions":
[".rdoc"]},{"name":"REALbasic","type":"programming","extensions":[".rbbas",".rbfrm",".rbmnu",".rbres",".rbtbar",".rbuistate"]},
{"name":"RHTML","type":"markup","extensions":[".rhtml"]},{"name":"RMarkdown","type":"prose","extensions":[".rmd"]},
{"name":"Racket","type":"programming","extensions":[".rkt",".rktd",".rktl",".scrbl"]},{"name":"Ragel in Ruby
Host","type":"programming","extensions":[".rl"]},{"name":"Raw token data","type":"data","extensions":[".raw"]},
{"name":"Rebol","type":"programming","extensions":[".reb",".r",".r2",".r3",".rebol"]},
{"name":"Red","type":"programming","extensions":[".red",".reds"]},{"name":"Redcode","type":"programming","extensions":[".cw"]},
{"name":"Ren'Py","type":"programming","extensions":[".rpy"]},{"name":"RenderScript","type":"programming","extensions":
[".rs",".rsh"]},{"name":"RobotFramework","type":"programming","extensions":[".robot"]},
{"name":"Rouge","type":"programming","extensions":[".rg"]},{"name":"Ruby","type":"programming","extensions":
[".rb",".builder",".fcgi",".gemspec",".god",".irbrc",".jbuilder",".mspec",".pluginspec",".podspec",".rabl",".rake",".rbuild",".r
bw",".rbx",".ru",".ruby",".thor",".watchr"]},{"name":"Rust","type":"programming","extensions":[".rs",".rs.in"]},
{"name":"SAS","type":"programming","extensions":[".sas"]},{"name":"SCSS","type":"markup","extensions":[".scss"]},
{"name":"SMT","type":"programming","extensions":[".smt2",".smt"]},{"name":"SPARQL","type":"data","extensions":
[".sparql",".rq"]},{"name":"SQF","type":"programming","extensions":[".sqf",".hqf"]},{"name":"SQL","type":"data","extensions":
[".sql",".cql",".ddl",".inc",".prc",".tab",".udf",".viw"]},{"name":"SQLPL","type":"programming","extensions":[".sql",".db2"]},
{"name":"STON","type":"data","extensions":[".ston"]},{"name":"SVG","type":"data","extensions":[".svg"]},
{"name":"Sage","type":"programming","extensions":[".sage",".sagews"]},{"name":"SaltStack","type":"programming","extensions":
[".sls"]},{"name":"Sass","type":"markup","extensions":[".sass"]},{"name":"Scala","type":"programming","extensions":
[".scala",".sbt",".sc"]},{"name":"Scaml","type":"markup","extensions":[".scaml"]},
{"name":"Scheme","type":"programming","extensions":[".scm",".sld",".sls",".sps",".ss"]},
{"name":"Scilab","type":"programming","extensions":[".sci",".sce",".tst"]},{"name":"Self","type":"programming","extensions":
[".self"]},{"name":"Shell","type":"programming","extensions":
[".sh",".bash",".bats",".cgi",".command",".fcgi",".ksh",".sh.in",".tmux",".tool",".zsh"]},
```

{"name":"ShellSession","type":"programming","extensions":[".sh-session"]},{"name":"Shen","type":"programming","extensions":
[".shen"]},{"name":"Slash","type":"programming","extensions":[".sl"]},{"name":"Slim","type":"markup","extensions":[".slim"]},
{"name":"Smali","type":"programming","extensions":[".smali"]},{"name":"Smalltalk","type":"programming","extensions":
[".st",".cs"]},{"name":"Smarty","type":"programming","extensions":[".tpl"]},
{"name":"SourcePawn","type":"programming","extensions":[".sp",".inc",".sma"]},
{"name":"Squirrel","type":"programming","extensions":[".nut"]},{"name":"Stan","type":"programming","extensions":[".stan"]},
{"name":"Standard ML","type":"programming","extensions":[".ML",".fun",".sig",".sml"]},
{"name":"Stata","type":"programming","extensions":[".do",".ado",".doh",".ihlp",".mata",".matah",".sthlp"]},
{"name":"Stylus","type":"markup","extensions":[".styl"]},{"name":"SuperCollider","type":"programming","extensions":
[".sc",".scd"]},{"name":"Swift","type":"programming","extensions":[".swift"]},
{"name":"SystemVerilog","type":"programming","extensions":[".sv",".svh",".vh"]},{"name":"TOML","type":"data","extensions":
[".toml"]},{"name":"TXL","type":"programming","extensions":[".txl"]},{"name":"Tcl","type":"programming","extensions":
[".tcl",".adp",".tm"]},{"name":"Tcsh","type":"programming","extensions":[".tcsh",".csh"]},
{"name":"TeX","type":"markup","extensions":
[".tex",".aux",".bbx",".bib",".cbx",".cls",".dtx",".ins",".lbx",".ltx",".mkii",".mkiv",".mkvi",".sty",".toc"]},
{"name":"Tea","type":"markup","extensions":[".tea"]},{"name":"Terra","type":"programming","extensions":[".t"]},
{"name":"Text","type":"prose","extensions":[".txt",".fr",".nb",".ncl",".no"]},{"name":"Textile","type":"prose","extensions":
[".textile"]},{"name":"Thrift","type":"programming","extensions":[".thrift"]},
{"name":"Turing","type":"programming","extensions":[".t",".tu"]},{"name":"Turtle","type":"data","extensions":[".ttl"]},
{"name":"Twig","type":"markup","extensions":[".twig"]},{"name":"TypeScript","type":"programming","extensions":[".ts",".tsx"]},
{"name":"Unified Parallel C","type":"programming","extensions":[".upc"]},{"name":"Unity3D Asset","type":"data","extensions":
[".anim",".asset",".mat",".meta",".prefab",".unity"]},{"name":"Uno","type":"programming","extensions":[".uno"]},
{"name":"UnrealScript","type":"programming","extensions":[".uc"]},{"name":"UrWeb","type":"programming","extensions":
[".ur",".urs"]},{"name":"VCL","type":"programming","extensions":[".vcl"]},{"name":"VHDL","type":"programming","extensions":
[".vhdl",".vhd",".vhf",".vhi",".vho",".vhs",".vht",".vhw"]},{"name":"Vala","type":"programming","extensions":[".vala",".vapi"]},
{"name":"Verilog","type":"programming","extensions":[".v",".veo"]},{"name":"VimL","type":"programming","extensions":[".vim"]},
{"name":"Visual Basic","type":"programming","extensions":[".vb",".bas",".cls",".frm",".frx",".vba",".vbhtml",".vbs"]},
{"name":"Volt","type":"programming","extensions":[".volt"]},{"name":"Vue","type":"markup","extensions":[".vue"]},{"name":"Web
Ontology Language","type":"markup","extensions":[".owl"]},{"name":"WebIDL","type":"programming","extensions":[".webidl"]},
{"name":"X10","type":"programming","extensions":[".x10"]},{"name":"XC","type":"programming","extensions":[".xc"]},
{"name":"XML","type":"data","extensions":[".xml"]},{"name":"XPages","type":"programming","extensions":[".xsp-
config",".xsp.metadata"]},{"name":"XProc","type":"programming","extensions":[".xpl",".xproc"]},
{"name":"XQuery","type":"programming","extensions":[".xquery",".xq",".xql",".xqm",".xqy"]},
{"name":"XS","type":"programming","extensions":[".xs"]},{"name":"XSLT","type":"programming","extensions":[".xslt",".xsl"]},
{"name":"Xojo","type":"programming","extensions":
[".xojo_code",".xojo_menu",".xojo_report",".xojo_script",".xojo_toolbar",".xojo_window"]},
{"name":"Xtend","type":"programming","extensions":[".xtend"]},{"name":"YAML","type":"data","extensions":
[".yml",".reek",".rviz",".sublime-syntax",".syntax",".yaml",".yaml-tmlanguage"]},{"name":"YANG","type":"data","extensions":
[".yang"]},{"name":"Yacc","type":"programming","extensions":[".y",".yacc",".yy"]},
{"name":"Zephir","type":"programming","extensions":[".zep"]},{"name":"Zimpl","type":"programming","extensions":
[".zimpl",".zmpl",".zpl"]},{"name":"desktop","type":"data","extensions":[".desktop",".desktop.in"]},
{"name":"eC","type":"programming","extensions":[".ec",".eh"]},{"name":"edn","type":"data","extensions":[".edn"]},
{"name":"fish","type":"programming","extensions":[".fish"]},{"name":"mupad","type":"programming","extensions":[".mu"]},
{"name":"nesC","type":"programming","extensions":[".nc"]},{"name":"ooc","type":"programming","extensions":[".ooc"]},
{"name":"reStructuredText","type":"prose","extensions":[".rst",".rest",".rest.txt",".rst.txt"]},
{"name":"wisp","type":"programming","extensions":[".wisp"]},{"name":"xBase","type":"programming","extensions":
[".prg",".ch",".prw"]}]

Appendix C

List of Programming Keywords

```
# JAVASCRIPT

--start
new, function, this, let, private, protected, public, static, var
--decision
case, else, if, switch
--other
try, catch, finally, break, while, for
--debug
console, debugger
--finish
return

# PYTHON

--start
def, class, self, import
--decision
else, if, elif
--other
try, except, finally, break, while, for
--debug
print
--finish
return

# JAVA

--start
new, public, private, protected, static, this, void, int, long, char
--decision
else, if
--other
try, catch, finally, break, while, for, do
--debug
System.out.println
--finish
return

# RUBY

--start
def, class, self, begin, end
--decision
else, if, elsif, case, when, then
--other
break, while, for, do
--debug
print, debugger, puts
--finish
```

```
return, end

# PHP

--start
public, private, protected, static, class, $this, require,
--decision
else, if, elseif, endif, case, switch
--other
try, catch, finally, while, for, do
--debug
print
--finish
return

# C++

--start
public, private, protected, static, class, main, this, float, char,
double, void, bool
--decision
else, if, case, switch
--other
try, catch, while, for, do
--debug
cout
--finish
return

# CSS
-- No words only symbols, but since we are not detecting the language,
this could disturb the other languages

# C#

--start
private, public, protected, static, class, new, override, this, float,
char, double, void, bool, var
--decision
else, if, case, switch
--other
try, catch, finally, while, for, do
--debug
Debug.WriteLine
--finish
return

# GO

--start
```

```
func, var
--decision
else, if, case, switch
--other
break, go, goto, for
--debug
godebug
--finish
return

# C

--start
static, void, using, namespace, #include, char, long
--decision
else, if, case, switch
--other
while, for, do
--debug
print, break
--finish
return

# TYPESCRIPT

--start
new, function, private, public, static, protected, class
--decision
case, else, if, switch
--other
try, catch, finally, while, for, do
--debug
debugger
--finish
return

# SHELL

--start
!/bin/bash
--decision
case, else, if
--other
try, catch, finally, while, for, do
--debug
echo
--finish
exit

# SWIFT
```

```
--start
private, public, static, class, func, import, let, var, self
--decision
switch, case, else, if
--other
break, try, catch, while, repeat, for, do
--debug
print, debugPrint, println, DEBUG
--finish
return

# SCALA

--start
def, object, class, new, import, private
--decision
match, case, else, if
--other
try, catch, finally
--debug
println
--finish
return

# OBJECTIVE-C

--start
main, long, int, short, long, int, void
--decision
case, else, if, switch
--other
break, while, for, do
--debug
DEBUG, NDEBUG
--finish
return
```

Appendix D

List of other keywords

GIT

--send
push, release, commit, add, delete

SVN

--send
add, update, commit, delete

Appendix E

Questions prompted by the tool

Questions Prompted By The Interruption Recovery Tool

1. Do you want to see the video from last session?
2. Thinking about your last interruption, do you think the tool helped you to recover in a faster way?
3. If you were not in a study, do you think you would use the tool to help now in your recovery?
4. What, if anything, did you miss in the last video?
5. What, if anything, would you remove from last video?
6. About the length of the video, it was:
7. Do you have any other feedback you think would improve the tool?

Appendix F

Removing tool instructions

Managing developer interruption

INSTALL INSTRUCTIONS

1. We need you to adjust your power save settings to ensure that your computer will not go into sleep mode for at least 5 minutes. If you are using a laptop this applies to when your laptop is plugged in and when your laptop is on battery.

2. Go to < SAMPLE sqrlab.uoit.ca/address_TBA> and download the package for your platform. Once the download is complete you can unzip the file anywhere on your computer.

USAGE INSTRUCTIONS

For Mac users: Open the command-line terminal/console and navigate to the ***wrapper*** folder inside the ***dist_mac*** folder. Once inside the ***wrapper*** folder you can start the tool by executing ***sudo ./wrapper.py*** [*IMPORTANT: You must use* ***sudo*** *in order to execute the tool.*]

For Windows users: Navigate to the ***wrapper*** folder inside the ***dist_windows*** folder. The tool is started by double clicking the ***wrapper.exe*** file.

For Ubuntu Linux users: Open the command-line terminal/console and navigate to the ***wrapper*** folder inside the ***dist_linux*** folder. Once inside the ***wrapper*** folder you can start the tool by executing ***./wrapper.py***

UNINSTALL INSTRUCTIONS

All of the collected data is anonymized and non-sensitive, and it is necessary for the study. Therefore, we ask that you please wait to uninstall the interruption recovery tool after the last day of the study and you also wait to delete the folder containing the study data until one week after the last day of the study.

1. To remove the interruption recovery tool you just need to delete the unzipped folder (***dist_<platform>***) and the ***dist_<platform>.zip*** file downloaded on the first day.

2. To remove the collected data, you just need to delete the folder ***<HOME_FOLDER>/FastRecoveryFiles***

If you have any problems please contact Gabrielle Dias (gabrielle.perezdias@uoit.net).

Opting Out of the Study: If you opt out of the study you can also use the above uninstall instructions to fully remove the interruption recovery tool and already collected data at any time.

HOW TO HANDLE PROBLEMS WITH THE TOOL

Although we have tested the interruption recovery tool, it is a research prototype and it is possible to encounter an occasional problem. Below are some solutions to the most common problems however we also ask you to report all issues to Gabrielle Dias (gabrielle.perezdias@uoit.net).

PROBLEM:

- When the tool is running for long, my computer gets very slow.

SOLUTION:

- First, try to go in **Settings** (you can see it in the tool screen, for mac users there's a settings icon in the toolbar) and reduce the **Video Quality**. If this does not solve the issue, please contact us sending the RAM available, platform (Windows, Mac, Linux) and its version. If there's any issue collecting this information, just mention this in the email.

PROBLEM:

- The tool quit unexpectedly and without any warning.

SOLUTION:

- *[Mac]:* Before restarting the tool you will need to make sure that part of the tool is not still running. To do this you need to go to a terminal/console and type **ps aux** to display a list of processes currently running. Look for a process called ***ffmpeg*** or ***dependencies_mac/ffmpeg***. If you find this process, copy its ID and kill it with ***sudo kill -9 <number of the process>.*** Finally, just restart the tool by executing ***sudo ./wrapper.py***.
- *[Windows]:* Before restarting the tool you will need to make sure that part of the tool is not still running. First you will need to type ***ctrl+shift+esc*** to open the task manager. Next select ***More details*** and look for a process called **ffmpeg** or **dependencies_windows/ffmpeg**. If you find the process, you need to right-click on it and select ***End Task*.** Finally, just restart the tool by double-clicking the ***wrapper.exe***.
- *[Ubuntu Linux]:* Before restarting the tool you will need to make sure that part of the tool is not still running. To do this you need to go to a terminal/console and type **ps aux** to display a list of processes currently running. Look for a process called **ffmpeg** or **dependencies_linux/ffmpeg**. If you find this process, copy its ID and kill it with **kill -9 <number of the process>.** Finally, just restart the tool by executing ***./wrapper.py***.

PROBLEM:

- I've clicked to restart the tool, but something happened and it did not start.

SOLUTION:

- Just restart the tool manually as described in the ***USAGE INSTRUCTION*** section.

PROBLEM [Mac only]:

- A questions dialog pops up, in front of a video window, but I can't interact it and I can't close it.

SOLUTION:

- Go to the terminal/console that was used to launch the interruption recovery tool and type ***ctrl+c*** to exit the tool. Reopen the tool and click "**Yes**" to see the last video that was displayed when the application froze.

PROBLEM [Windows only]:

- I've clicked to exit the tool, but it keeps running.

SOLUTION:

- Go to the terminal/console and type ***ctrl+c***. After the tool closes, you will need to type ***ctrl+shift+esc*** to open the task manager. Next select ***More details*** and look for a process called **ffmpeg** or **dependencies_windows/ffmpeg**. If you find the process, you need to right-click on it and select ***End Task*****.** If you can't find the process you don't need to do anything else.

Appendix G

Last Day interview

Last Day Interview

1. Overall what do you think about the tool?
2. Do you think it helped you during the recovery process?
3. What do you think might be improved in the tool in order to make it better for you?
4. What do you think about the duration of the video?
    a. If it's too long, do you have a suggestion of what could be cut?
5. What do you think about the way the video and other information are presented to you? Would you change anything?
6. What helped you the most?
    a. If they do not have an answer, use these prompts: The video? The thumbnails? The changed files?
7. Is the tool easy or difficult to use?
8. Do you think the video was accurate? Did you miss parts?
9. Did you generally watch the video?
    a. If not, why not?
10. If you were asked to use this tool outside the study, would you use it?
11. Do you think it would improve your work?
12. Do you think the time spent watching the videos were valuable to you?

Only for 2nd phase:

1. Did you notice any change in the tool?
2. Do you think it has improved since last phase?
3. Do you think the video was accurate? Did you miss parts?
4. If you were asked to use this tool outside the study, would you use it? Do you think it would improve your work?