Bachelor Thesis

August 25, 2022

I am Stuck

Characteristics of Stuck Phases During Software Development

Samuel Andreas Brügger

of Graben (BE), Switzerland (18-919-498)

supervised by

Prof. Dr. Thomas Fritz
Alexander Lill
Roy Rutishauser

University of Zurich^UZH

HASEL

Bachelor Thesis

I am Stuck

Characteristics of Stuck Phases During Software Development

Samuel Andreas Brügger

University of Zurich^UZH

HASEL

Bachelor Thesis

Author: Samuel Andreas Brügger, samuelandreas.bruegger@uzh.ch

Project period: 28.02.2022 - 28.08.2022

Human Aspects of Software Engineering Lab
Department of Informatics, University of Zurich

Acknowledgements

I thank Prof. Dr. Thomas Fritz, Alexander Lill, and Roy Rutishauser for their time and their expertise which is very well appreciated and helped me a lot during the work on this thesis. A big thank you to all the authors and software engineers of the User Input Tracker, the Windows Activity Tracker, the OpenFace 2.0, and especially to Roy Rutishauser for allowing me to use their projects and code to create many of the key features of the Stuck Detector. I also want to thank all the participants of the user study for their time and the data they have provided. I am also very grateful for the support of my family and friends and especially for Nicole for her love and motivation during this time.

Abstract

A very common reason for the unhappiness of software engineers is being stuck in problem-solving. We developed a data gathering tool named Stuck Detector which uses face recognition and computer interaction trackers for users to report their stuck phases. The study with six software engineers consisted of a pre-study questionnaire, a two-week data gathering phase, and a final interview. The data of this study was used to create a definition for the term stuck and to observe some characteristics of stuck phases. A software engineer can be labelled as stuck, if she or he is working on a software engineering problem for a longer period, has tried different options, which didn't work, lacks information, or lacks knowledge, and can't make any progress towards the goal of the task. The results of the study implied that the frequency of stuck phases varies and is heavily dependent on the task. Furthermore, it was stated by the participants that a stuck phase occurs on average once a day. In the period of around 20 minutes before a stuck phase, the user input activity tends to decrease while the number of window changes and frustration had a tendency to increase. The participants stated that privacy and being monitored for productivity reasons were the biggest concerns about data gathering and stuck detecting tools.

Zusammenfassung

Ein sehr häufiger Grund für die Unzufriedenheit von Softwareentwicklern ist das "stuck" sein beim Lösen von Problemen. Wir haben ein Datenerfassungstool namens Stuck Detector entwickelt, welches Gesichtserkennung und Computerinteraktions-Tracker verwendet, mit welchem Benutzer ihre Phasen, in welchen sie "stuck" sind, melden können. Die Studie mit sechs Softwareentwicklern bestand aus einem Vorstudienfragebogen, einer zweiwöchigen Datenerhebungsphase und einem abschliessenden Interview. Die Daten aus dieser Studie wurden verwendet, um eine Definition von "stuck" sein zu erstellen und einige Merkmale von "stuck"-Phasen zu beschreiben. Ein Softwareentwickler kann als "stuck" bezeichnet werden, wenn sie oder er über einen längeren Zeitraum an einem Softwareentwicklungsproblem arbeitet, verschiedene Lösungsoptionen ausprobiert hat, welche nicht funktioniert haben, der Person Informationen oder Kenntnisse fehlen und keine Fortschritte erzielt werden können. Die Studie deutete an, dass die Häufigkeit von "stuck"-Phasen unterschiedlich ist und stark von der Aufgabe abhängig ist. Die Studienteilnehmer gaben an, dass Phasen, in welchen sie "stuck" waren im Durchschnitt einmal täglich vorkommen. Im Zeitraum von etwa 20 Minuten vor einer "stuck"-Phase nimmt die User-Input-Aktivität tendenziell ab, Anzahl Fensterwechsel und Frustration nehmen tendenziell zu. Die Teilnehmer gaben an, dass Datenschutz und Überwachung aus Produktivitätsgründen die größten Bedenken hinsichtlich der Datenerfassung und für Tools zur Erkennung von "stuck"-Phasen sind.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software engineering is a field that gained the interest of researchers in recent years [BS08]. Some studies look at different factors of productivity of software engineers with the potential to improve the productivity with their findings [CD09]. Thus understanding why software engineers are unhappy and because of that unproductive is a crucial step [GFWA17]. Graziotin et al. asked software engineers for the reasons why they were unhappy. They reported that the most common reason was being stuck in problem-solving.

Even though the term stuck is used in literature, it has not one consistent definition used by different research papers [BP04, CD12, MF15]. The term is sometimes rather described or explained with examples because it is a subjective feeling thus people often have already an idea of what it could mean. [BP04]

There is already research about the productivity or engagement of software engineers which includes statements about being stuck [BS08, CD12, GFWA17, MF15]. But often these papers had another main focus like the whole spectrum of productivity [MF15], or all possible difficulties an inexperienced software engineer could have [BS08, GFWA17] or they were only interested in a specific niche like distributed programmers [CD12].

A clear definition of the term stuck and other characterizing information about stuck phases could help this field of study to gain more traction and to make statements about being stuck less ambiguous. This paper aims to make a more precise definition of the term stuck based not only on subjective descriptions but also on the data of the change in the behaviour of a software engineer during these phases. It also tries to list some observed characteristics of the stuck phases and it shows a possible approach for a data gathering tool and reviews that to list possible requirements for such a tool and for a stuck detection program.

To achieve the objectives of this paper we developed a data gathering program called Stuck Detector with user input trackers, window activity trackers, and facial recognition. We asked the users to report possible stuck phases and to answer a few questions to further describe these phases. Before this program was used by six software engineers for two weeks, we conducted a pre-study questionnaire and after the two-week long data gathering phase, each participant was also interviewed and was asked further questions about the subject of being stuck as a software engineer.

The gathered data was used to answer the following research questions.

- **RQ1.1:** How can you define being stuck in the field of software engineering?
- **RQ1.2:** How often are software engineers stuck?
- **RQ2:** To what extent is it possible to detect stuck phases of software engineers using data from face recognition and computer interaction?
- **RQ3.1:** What are the requirements for a program to be suitable for collecting such data?
- **RQ3.2:** What are the requirements for a program to recognize stuck phases?

Based on the data from the pre-study questionnaire, the user study, and the interview, the following definition for the term stuck in the context of software engineering was created. A software engineer can be labelled as stuck, if she or he is working on a software engineering problem for a longer period, has tried different options, which didn't work, lacks information or lacks knowledge, and can't make any progress towards the goal of the task. This term aims to include most situations the participants considered as stuck and excludes other situations with low productivity, like not being focused on the task.

The frequency in which the participants were stuck was heavily dependent on their current task. The gathered data from the user study indicates that a software engineer is stuck on average around every four hours. This time doesn't include meetings. The reported frequency in the interview was on average around once a day but with an emphasis that it is dependent on the tasks.

The data of the user study showed that there is a tendency that the user input activity, like mouse movement, mouse clicks, and key strokes, decreases before a stuck phase. The number of window changes showed a clear tendency to increase, while the frustration level had a lower tendency to increase. Head movement showed no dominant tendency of change before or during stuck phases.

A data gathering tool or a stuck detection tool shouldn't interrupt the work of a user unless it is crucial for the task. The participants preferred to report stuck phases when they occurred or reflect on the last hour rather than reflecting on the end of the work day. Privacy was a big concern of the participants. Most of them would only use such a tool when the data is gathered only locally, it is only used to check for stuck phases and not to monitor the user or their work and for a stuck detection program, it should delete the data as soon as it isn't necessary anymore for the real-time detection of stuck phases.

Chapter 2

Related Work

The area of studying factors of productivity in the work field gets a lot of interest from research and practice [BS08, BP04, CD12, CD09], as there is potential for optimization as stated by Carter and Dewan [CD09]. There are already studies, that mainly focus on the field of software engineering [ZMMF18, MF15]. These studies are focused on different factors that influence productivity, for example interruptibility [ZMMF18]. The hypothesis that happy workers are more productive is supported by recent research [GFWA17]. It follows that if one can better control the reasons for unhappiness of software engineers in a way that increases their happiness, their performance should improve. In a survey where software engineers were asked to name reasons for their unhappiness, the most frequent answer was being stuck in problem-solving [GFWA17]. This chapter gives a short overview which definitions of stuck are used, how well being stuck is researched in the field of software engineering, and how face recognition and computer interaction data is used in comparable research areas.

2.1 Stuck

This section gives an overview of how the term stuck was described and used in research so far and how much attention stuck phases got in the field of software engineering, and shows the lack of research regarding overcoming stuck phases.

2.1.1 Definitions of Stuck

There are multiple definitions of the term stuck. These are rather descriptions than definitions, such as a negative corollary to flow [BP04], which leaves us with the question of how flow is defined. Other definitions include descriptions like having difficulties [CD12] or not making progress [MF15]. Studies can be found which use the term stuck without a definition [BS08]. Stuckness is rather subjective than objective and gets described for example as a feeling of not being in control, having only little concentration, difficulties focusing and being inattentive, mental exhaustion, desperation with the task, and more similar descriptions [BP04]. Being stuck is often connected to getting frustrated [CD12, KBP07, BP04, MF15] and frustration is defined more clearly [GWB+13a].

2.1.2 Stuck during Software Engineering

Being stuck is a major reason for unhappiness of software engineers and thus lowers their productivity [GFWA17]. Studies that involve stuck phases of software engineers often have a broader focus, for example it includes all possible struggles or difficulties of a software developer [BS08, CD12, GFWA17] or it focuses on a wider spectrum of productivity [MF15]. Furthermore, there is research that focuses only on a smaller niche of software engineers like distributed programmers [CD09]. They hypothesize that it is possible to automatically identify that distributed programmers are stuck.

2.1.3 Possible Solutions

The studies we found, where solutions for stuck phases were discussed, were not conducted in a work setting or were focused on learning in general [BP04]. There was no study found that only focuses on the resolution of stuck phases of software engineers. The lack of resources can be seen as an indicator of why there is a need to better understand what being stuck is.
One method which is used in the field of software engineering is rubber ducking[1]. With this method you explain every line of your code to a rubber duck with the goal to find errors or problems with your logic. No research could be found about the effectiveness of this method.

2.2 Face Recognition

In recent research, face recognition is a well-established tool to use for detecting or predicting expressions and emotions as many studies use such data [BP04, KBP07, BSN$^+$20, HLH$^+$13, AKN$^+$19, BDO$^+$16, GWB$^+$13a, GWB$^+$13b, UJ11, VAG$^+$15, WSL$^+$14]. A well-established taxonomy for classifying different facial expressions is the Facial Action Coding System (FACS) proposed by Ekman and Friesen in 1978 [WSL$^+$14]. This system is used in a facial recognition tool called OpenFace 2.0 which also measures eye gaze and head orientation [BZLM18].
So far researchers are not as interested in head pose estimates as in facial landmark detection [BZLM18], even though head pose is a reliable feature to determine the focus of people's attention [HLH$^+$13].
Face recognition is used for detecting emotions in several studies [KBP07, BDO$^+$16, WSL$^+$14]. Some of them are focused only on frustration [GWB$^+$13a, GWB$^+$13b, KBP07]. But it can be also used to predict engagement [WSL$^+$14].

2.3 Computer Interaction

Computer interaction which includes the input of the user and what she or he does like changing windows is a data source used in different studies [ZMMF18, KBH13]. There was no general correlation found between the computer interaction behaviour and mood, but there was a correlation when the regression model was personalised [KBH13]. But for other fields like predicting interruptibility, models with the data from the computer interaction had an accuracy of 74.8% and thus had a higher predicting accuracy than models using biometric data with 68.3% accuracy [ZMMF18].

[1] https://medium.com/@katiebrouwers/why-rubber-ducking-is-one-of-your-greatest-resources-as-a-developer-99ac0ee5b70a

Chapter 3

Approach and Research Prototype

We developed a computer program for this study called Stuck Detector which was designed to gather data about stuck moments and the users' activity and facial expressions during their work time, with the goal to see if there are signs in the data that one is or soon will be stuck. The user can start the data gathering by clicking on a button, can see the activity data from the last hour, and can report stuck phases. For this study the tool is only gathering data and does not detect stuck phases as we wanted to gather this data without a bias from a used model of prediction for stuck phases. The main concepts and key features of the Stuck Detector, like the session and the user activity graphs, are presented in this chapter.

3.1 Approach

Our approach is based on the idea, that a tool should intervene as less as possible with the user to not alter the data. With this in mind, we developed a tool called Stuck Detector where the user could control when the computer interaction and face recognition data is gathered and only has to interact with the program when she or he wants to report a stuck phase or when she or he has to answer the reminder pop ups. To be able to control the data gathering the user could start and stop sessions. These sessions showed graphs with the activity data of the last hour when data was recorded in this period. The graphs were an assistance for the user to reflect if there was a stuck phase in the last hour. We didn't know whether a software engineer could identify every stuck phase when it occurred or rather in retrospect. That's why they were asked every hour if they were stuck. With this reminder and the graphs with the activity of the last hour, we hoped all or at least as much as possible of the stuck phases a user encounters will be reported.

3.2 Prototype

Stuck Detector was developed for the operating system Microsoft Windows 10 and is based on Electron[1]. The tool works completely offline and gathers the data locally. NodeJS[2] was used due to the fact that the program needs access to the camera and the file system for writing into the database and because it needs the information about processes.

[1] https://www.electronjs.org/

[2] https://www.nodejs.org/

3.2.1 Session

The main features of the Stuck Detector are embedded in the session. The term session can be used as a description of the time span in which a software engineer is working without a break. If the session isn't started, no data is saved and the sensors are not gathering data with the only exception of the processes and window title tracking which checks if the user is using one of the popular Integrated Development Environments (IDEs) and would suggest starting a session in this case. This was made to ensure, that a participant wouldn't forget to start a session after opening the Stuck Detector and beginning to work.
When a session is started, every sensor gathers data, which is saved to the database in an interval of a second. The data of the mouse and keyboard activity as well as the used processes of the last hour will be presented in two different graphs and they update every 20 seconds. These graphs can be used by the user to reflect, if or when she or he was stuck in the last hour and to better pinpoint the moment the stuck phase began.

Activity Graph

For the activity graph we used the data from the key input and the data from the mouse movement. This graph should help the user to see how its activity changed over time. If, for example, it changes from a high amount of key strokes to a high activity of the mouse, this could be an indication for a software engineer, who was programming, that the workflow halted or a stuck phase began. An example of an activity graph can be seen in Figure 3.1.

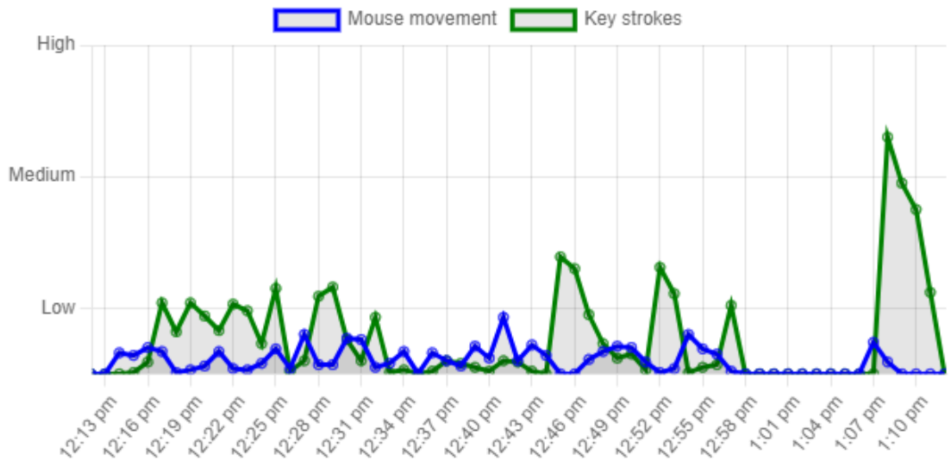


Figure 3.1: Activity Graph

We used a time frame of ten seconds for each data point, which means, that the program sums up the key strokes of ten seconds which are ten different data points, to one data point and the moved distance from the mouse cursor to one point for every ten seconds. The graph shows the data from the last hour and classifies the data points as low, medium, or high activity. A user can hover over the data points with the mouse to get the exact value as a number. For the key strokes, this is just the number of keys pressed in this time frame of ten seconds. The value shown for the mouse movement is the travelled distance of the mouse cursor in pixels.

Processes Graph

With the processes graph, we wanted the user to see at which time he used which processes. This should help her or him to be more precise when determining the starting time of a stuck phase. If for example, the user started to change windows more often because she or he is searching for information on the internet and then going back to the IDE, this could be an indication of the beginning of a stuck phase. An example of a processes graph can be seen in Figure 3.2.

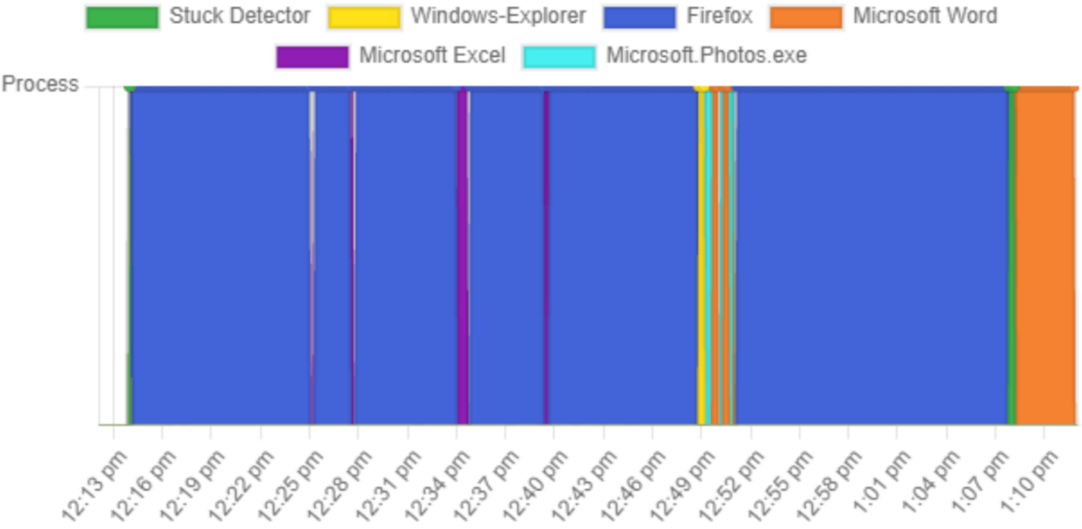
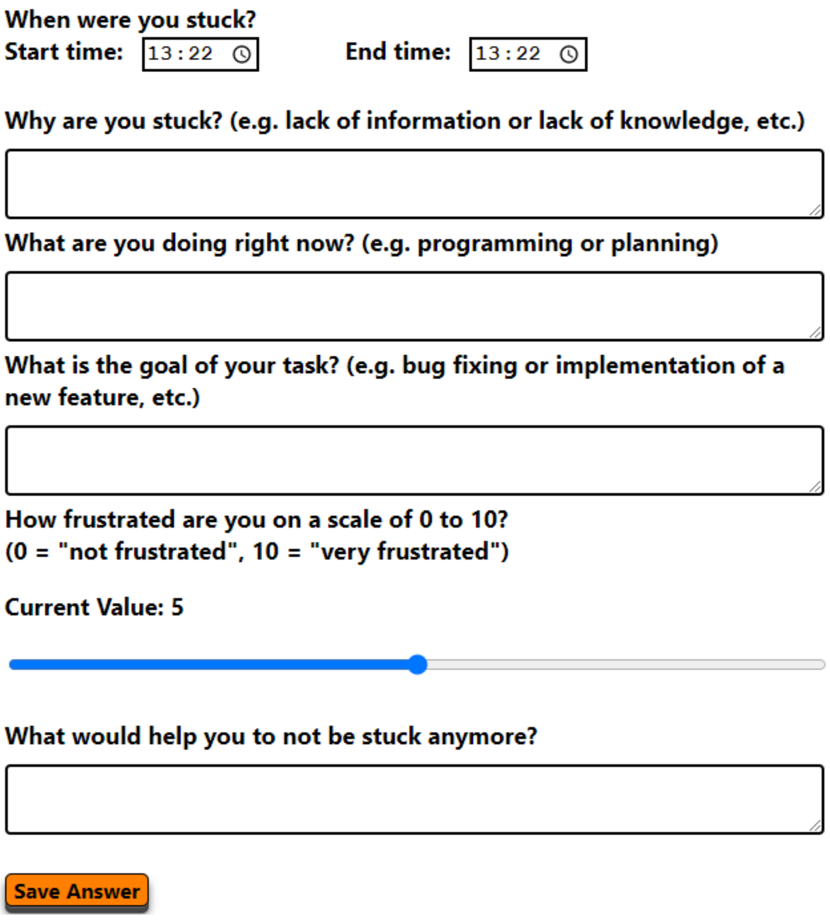


Figure 3.2: Processes Graph

To make the graph less noisy, we made time frames of ten seconds, in which every process gets noted, but the whole block will be coloured only in one colour. If a user hovers with the mouse cursor over the data point, every process which was in focus during this time frame will be shown. This gives us a colour coded time series, where a series painted in only one colour would represent a phase, where only one program was dominantly used.

3.2.2 Stuck Report

If a user reported a stuck phase, a stuck report field opened and the user had the opportunity to give some information about the stuck phase. Most importantly, they could set the start and end times of the stuck phase. The users were also asked why they were stuck, what they were doing at that moment and what the goal of their task was. Additionally, participants were asked how frustrated they were on a scale from zero to ten and they had the opportunity to give information about what would have helped them not be stuck anymore. The stuck report form can be seen in Figure 3.3.



When were you stuck?

Start time: 13:22 End time: 13:22

Why are you stuck? (e.g. lack of information or lack of knowledge, etc.)

What are you doing right now? (e.g. programming or planning)

What is the goal of your task? (e.g. bug fixing or implementation of a new feature, etc.)

How frustrated are you on a scale of 0 to 10?
(0 = "not frustrated", 10 = "very frustrated")

Current Value: 5

What would help you to not be stuck anymore?

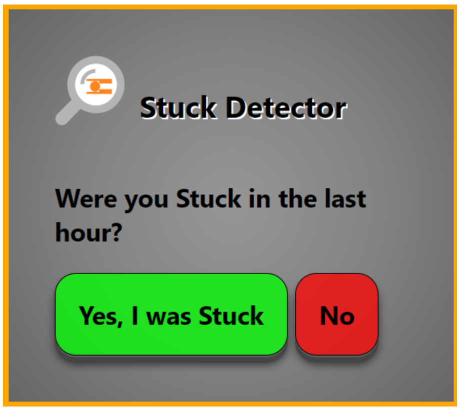
Figure 3.3: Stuck Report Form

3.2.3 Pop Ups

We used two different pop ups for the program. One was a reminder to ask if the user was stuck and the other was a reminder to start a session if no session was started yet but the user was working with a known IDE. The purpose of these pop ups was to ensure that as many stuck phases as possible can be captured and that the data is gathered, whenever the participant was working.

Hourly Reminder

This pop up was shown every hour in the lower right corner of the screen when the session was running and when no stuck phase was reported in the last hour. It was programmed to get in focus and closes after the question was answered. The user was asked if she or he was stuck in the last hour and if the user pressed no, the pop up just closed. If the user answered yes, the pop up also closed, but the main window with the session view came into focus, such that the user could report the stuck phase. The hourly reminder is shown in Figure 3.4.



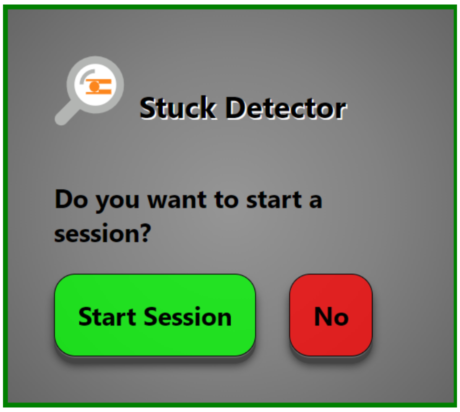
Stuck Detector

Were you Stuck in the last hour?

Figure 3.4: Hourly Reminder

Session starting Reminder

The session starting reminder is a pop up, which is shown in the lower right corner of the screen when no session was running but the user was working with an IDE for the last 20 minutes. It asked if the user would want to start a session. If the user answered no, the pop up closed, and if the user answered yes, the pop up closed, and the main window with the session view showing the starting button came into focus. The session starting reminder is shown in Figure 3.5.



Stuck Detector

Do you want to start a session?

Figure 3.5: Reminder for Session Start

Chapter 4

Methodology

We used a pre-study questionnaire, a two-week-long field study, and a final interview to gather data. This chapter gives an overview of the procedure, the participants, the recruitment process, and the concerns of the participants regarding the data gathering tool. It also introduces the different components of the user study in detail and describes the data which was gathered, how it was preprocessed before analysing, and gives an overview of the collected data.

4.1 Procedure

The user study had three main components with the goal to get data on stuck phases from software engineers in real work situations with the goal to get a better understanding of stuck phases and their characteristics. Before the usage of the Stuck Detector, the participants filled out the pre-study questionnaire. After that, they used the tool for two weeks and then had an interview with one of the researchers. During the two weeks of the study, the participants had to follow their normal routine and start a session, whenever they did work related to software engineering. All interactions with the Stuck Detector were saved to the local database.

4.1.1 Participants

The focus of the user study was on software engineers. There were no restrictions on the profession, as long as it was related to the field of software engineering. This also includes students of computer science-related studies with or without practical experience. We were able to find seven participants for the user study. The participants were between 23 and 28 years old and all males. Unfortunately, one of them had to leave the study, because his device was lost during the study. This left us with six complete data sets with a filled-out pre-study questionnaire, data from the two weeks using the tool, and a completed final interview.
The participants had between half a year and ten years of work experience in a software engineering-related field and between two and eleven years of programming experience. Three of the participants were students and the other three worked as software engineers. Only two of the participants had the permission of their employer to use the tool at work. The others either weren't allowed to use the tool during work and thus, only used it on their private device for personal projects or were students and thus used the tool while working on projects, which were mostly related to their studies.

Recruitment Process

The way used for the recruitment was the personal network of the researchers. All possible participants were contacted individually and there was no contact with an enterprise, as one of the aims of the study was to get data from different fields and occupations to not get a bias from a possible unique work environment. As the program was developed only for Microsoft Windows, only participants using this operating system qualified for the study. The information given upfront was limited to only a little information about the topic, the data which would get gathered and the sensors used for that, and also the tasks a participant has when joining the study, like the pre-study questionnaire, the use of the tool for two weeks and the final interview. The reason for this was, that the participants shouldn't be influenced by the explanation and purpose of the study before they would fill out the pre-study questionnaire, because some aspects, like the definition of being stuck, are rather subjective.

Concerns

There were several concerns by the asked participants about the data gathering. One of these concerns was that the user input tracker collects data about keyboard input and they were worried that it would work like a key logger which tracks not primarily the frequency but mainly the value of the pressed button which could be used to get confidential data like passwords or other internal information. These concerns were therefore quickly eliminated as we were only interested in the frequency of the button presses.

Another great concern was the camera which most participants would not have connected all the time or even taped off when there was no meeting or other use for the camera. The data which was saved to the database were numbers describing the action units of facial recognition, the presence of these at each time, described as integers, and the pose of the head. The fact that no picture was saved to the database was important for five of the participants.

The last major concern was the data gathering in general. There were four participants who would have declined if there were no possibility to start and stop the data gathering feature of the tool by themselves. This and the fact that the data was only gathered locally, that they could easily look at the complete data set at any time, and that they could decide to not send any data at the end of the study if other concerns arose during these two weeks, convinced them to participate.

Because of this setting, no asked participants refused to take part in the study because of a safety or privacy concern on their behalf. But several participants couldn't take part in the study at their workplace, as their enterprise rule on external software wouldn't allow this tool.

4.1.2 Pre-Study Questionnaire

Before the data gathering phase, we sent participants a pre-study questionnaire with the aim to get their uninfluenced view as software engineers on the topic of being stuck. The pre-study questionnaire consisted of eight questions. These questions were aimed to give feedback on what being stuck means for different software engineers. They were asked about their definition of being stuck, how often they encounter these stuck moments, how they realise it, what they do to overcome these phases, what they could additionally do in these moments, if they would appreciate if a program could tell them when they are stuck, and how it should do that. All answers were text-based and the questions about the frequency didn't have any scale to not frame a possible answer. The questions of the pre-study can be found in Appendix C.

4.1.3 Data Gathering Phase

During a time span of two weeks, the participants used the tool and gathered data whenever they were working on a project as software engineers. They could contact us any time if they had problems or questions and they had the possibility to view their own data at any time. The data was collected locally and not sent to the researchers before the end of the study.

During this phase, they were asked every hour with a pop up if they were stuck, but besides that, they would work normally and only had to start and stop the sessions and fill out the stuck report in the program when they realised that they were stuck.

4.1.4 Final Interview

The final interview was held in German, with every participant individually, and took between 16 and 28 minutes. They were conducted online and with the permission of the participants the audio was recorded and transcribed. It consisted of 25 main questions, though the first eight were the same as from the pre-study questionnaire because we wanted to know if the participants changed their views on some of these questions after the field study.

The other questions of the final interview were aimed to see how well the program was suited for data collection, what they would change about it, and what their opinion is about such programs, concerning the data collection and the goal of the tool. We also asked them some demographic questions like age, occupation, and experience in software engineering. The interview guide with the main questions can be found in Appendix D.

4.2 Data Collection and Preprocessing

The user study has collected data from the pre-study questionnaire, from the usage of the program, and from the final interview. We were able to collect six complete data sets, even though the number of hours the participants recorded with the tool varies.

4.2.1 Quantitative Data Collection

If a session was started, a data point was written every second to the local database with the information of the face recognition, the windows activity tracker, and the user input tracker. Because of the interval of around one second in which data points were added, we used a timestamp with a precision of milliseconds. Whenever a participant reported a stuck moment, information about this stuck phase was saved to the local database.

Program Interaction Data

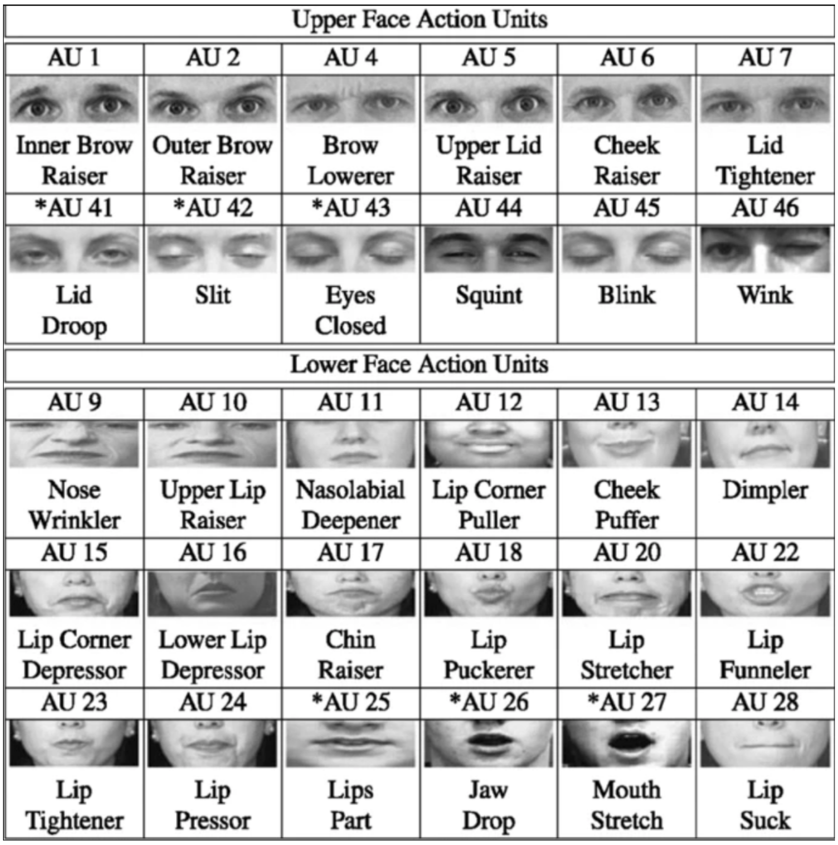
One table in the database held all information about the usage of the program. Every interaction was saved with a timestamp. This includes every starting and closing of the program, every starting and stopping of a session, every pop up and every answer to these pop ups, and the reporting of stuck phases. With this information, we can analyse how the program was used and see how the different features, like the hourly reminder, function in a real work environment.

User Data

The data from the windows activity tracker, the user input tracker, and the face recognition form the user data. All data concerning the user was saved in its own table in the database. We got four different values from the user input tracker. One was for the number of key strokes (keyTotal), one was for the total number of mouse clicks (clickTotal), one was for the moved distance of the mouse courser (movedDistance) and the last one was about the scroll data from the mouse (scrollDelta). All these data values were given as integers.
We got three different bits of information from the windows activity tracker. One was the title of the window (windowTitle) of the running application which was in focus, another one was the process name (process) of the application and the last one is the possible activity the user could have been doing (activity), which is based on a mapping from processes to activities. This mapping was already provided from the windows activity tracker.
The face recognition tracker gives us 40 different information about facial features and head pose. We got the information for the intensity of 17 facial action units measured in a decimal number, the presence of the same facial features expressed as an integer, and the head pose measured with six different decimal numbers. The used action units are listed in Table 4.1 with their action unit (AU) number as an identifier and the facial action coding system (FACS) name as a description.

AU number	FACS name
01	Inner brow raiser
02	Outer brow raiser
04	Brow lowerer
05	Upper lid raiser
06	Cheek raiser
07	Lid tightener
09	Nose wrinkler
10	Upper lip raiser
12	Lip corner puller
14	Dimpler
15	Lip corner depressor
17	Chin raiser
20	Lip stretcher
23	Lip tightener
25	Lips part
26	Jaw drop
45	Blink

Table 4.1: Action Units and their According FACS Name



Upper Face Action Units					
AU 1	AU 2	AU 4	AU 5	AU 6	AU 7
Inner Brow Raiser	Outer Brow Raiser	Brow Lowerer	Upper Lid Raiser	Cheek Raiser	Lid Tightener
*AU 41	*AU 42	*AU 43	AU 44	AU 45	AU 46
Lid Droop	Slit	Eyes Closed	Squint	Blink	Wink
Lower Face Action Units					
AU 9	AU 10	AU 11	AU 12	AU 13	AU 14
Nose Wrinkler	Upper Lip Raiser	Nasolabial Deepener	Lip Corner Puller	Cheek Puffer	Dimpler
AU 15	AU 16	AU 17	AU 18	AU 20	AU 22
Lip Corner Depressor	Lower Lip Depressor	Chin Raiser	Lip Puckerer	Lip Stretcher	Lip Funneler
AU 23	AU 24	*AU 25	*AU 26	*AU 27	AU 28
Lip Tightener	Lip Pressor	Lips Part	Jaw Drop	Mouth Stretch	Lip Suck

Figure 4.1: FACS Action Units[1]

Figure 4.1 shows the actions units and gives a picture as example for each. The asterisks (*) besides some of the AU labels are an indication, that the criteria for this action unit have changed[1].

Stuck Data

In the table of the stuck data the program stored every report from the participants with a start and an end time and the answers the users gave in the stuck report. If they didn't answer the questions or didn't specify the end time, the end time was set to the time, the report was sent and the other fields remained empty.

[1] http://what-when-how.com/face-recognition/facial-expression-recognition-face-recognition-techniques-part-1/

4.2.2 Qualitative Data Collection

With the questions from the pre-study questionnaire and the interview, we wanted to gather additional information concerning being stuck as a software engineer, which could help us understand the user data and could be used to help us answer the research questions 1.1, 1.2, 3.1, and 3.2.

Questionnaire Data

The data from the pre-study questionnaire was sent back by the participants before or shortly after the start of the user study. They answered all the questions by text in German. We grouped the answers per question and used thematic analysis according to Braun and Clarke [BC06].

Interview Data

We transcribed the audio data from the interviews and used the same thematic analysis methods as with the pre-study questionnaire [BC06]. The answers were also grouped by the main questions of the interview, which are listed in Appendix D.

4.2.3 Data Preprocessing

To be able to analyse the data we calculated the number of window changes, marked the time a user was stuck in the user data and aggregated the data points such that we had the data for every minute as one data point and not as 60 data points, one for every second. The next step was to normalize the data so that the values were in a range from 0 to 1. This smoothed out a lot of noise and made it possible to get meaningful data plots.
We combined the data of the stuck reports and the user data simply by adding a new field for every data point, which indicated if the user was stuck at this moment. We also cleaned up the data set, for example by removing the stuck reports which were made right after the installation of the program, which resulted from the testing of the stuck report function by the participants. We also combined stuck phases, when they were only seconds apart from each other and when one of them had no answers, thus we handled them as double reports of the same stuck phase. Another step was to make a count for the window changes, so we looked if the window title has changed for each second compared to the data from the second before and when they differed, we set the number of window changes for this second to one. For all other data points without a change, we set this number to zero. This way we could just sum up this column for a given time span and had the total amount of window changes for this frame. To normalise the data, we divided the values by the distance from the value two standard deviations below the mean and the value two standard deviations above the mean and took the absolute value as the result. Every point outside of this boundary got either the value zero or one depending if it was more than two standard deviations below or above the mean.

4.3 Data Overview

We got six complete data sets from the user study. Table 4.2 gives a short overview of essential points of the user data from the tool.

Participant	Time recorded	Different Days	Stuck Phases	Sessions started
1	35h 23m 41s	7	1	41
2	06h 00m 02s	3	8	14
3	08h 36m 28s	4	1	19
4	36h 35m 24s	11	12	69
5	10h 06m 29s	4	0	6
6	03h 12m 42s	4	2	5
TOTAL	**99h 54m 46**	**33**	**24**	**154**

Table 4.2: Overview of the User Data

One can see that the recorded times differ a lot. This is partially due to the fact, that participants one and four were allowed to use the tool at work, which also could explain the difference in the number of days the tool was used. All other participants used it either for private or for school projects.

The number of sessions started shows that during a work day there are a lot of breaks, interceptions, meetings, or other events, which interrupt the times when the users were working on their own. The average time per session of every participant is shown in Table 4.3. If we sum up the recorded times of all participants and divide it by the number of sessions started, we get the average length of a session.

Participant	Average Length of Session
1	00h 51m 48s
2	00h 25m 43s
3	00h 27m 11s
4	00h 31m 49s
5	01h 41m 05s
6	00h 38m 32s
AVERAGE	**00h 38m 56s**

Table 4.3: Length of Sessions

The average time per session was 38 minutes and 56 seconds. Participant 5 has a higher average time per session of 1 hour 41 minutes and 5 seconds, which could indicate, that his private work environment doesn't have as many interruption possibilities as the ones of the others.

Figure 4.2 gives an overview of all the gathered data from one day of participant 4.

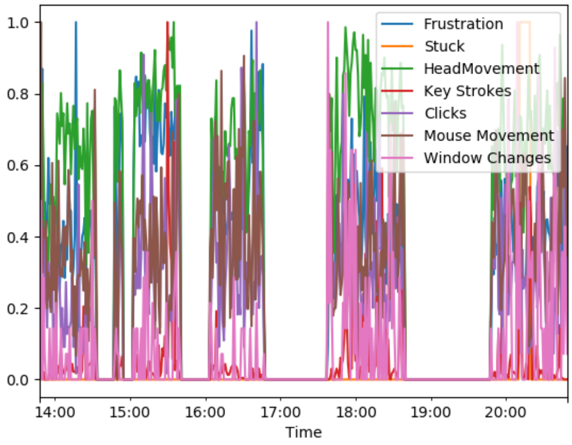


Figure 4.2: Example Overview of Data from one Day

Figure 4.3 gives an overview of all the gathered data from one session of participant 4.

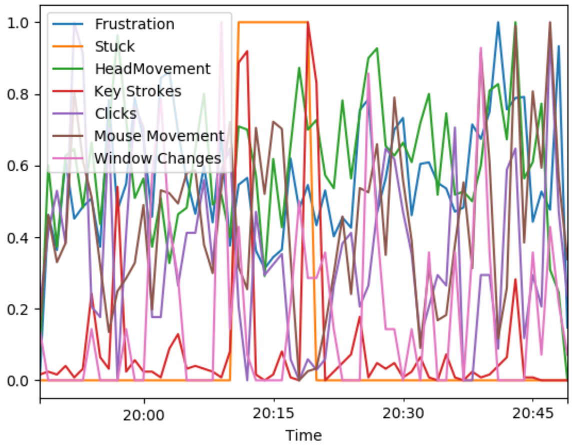


Figure 4.3: Example Overview of Data from one Session

Chapter 5

Results and Analysis

The following chapter answers the research questions with the help of the data from the tool and the data from the pre-study questionnaire and the final interview. The research questions are listed in the introduction. A definition for stuck phases based on the statements of the participants can be summarized as a time when they can't make any progress and don't know how to proceed. We developed a definition with more detail to include the described stuck moments of the participants (see section 5.1). The frequency and duration of stuck phases vary and are dependent on the user, the task, and the knowledge the user has of the technology (see section 5.2). One can see in the data of the field study that there are behavioural changes before and during a stuck phase, there is for example a tendency for the user input activity to decrease and for the number of window changes and frustration level to increase (see section 5.3). There was also data from the stuck reports describing how the participants could overcome the stuck phase (see section 5.4). We could also use the data from the interview and the feedback from the participants to list some requirements for data gathering and stuck detection tools. It was important for them to not get monitored about their productivity and that the data is only used for detecting stuck phases (see section 5.5).

5.1 Definition of Stuck for Software Engineers (RQ1.1)

The participants were asked how they would define being stuck. All their answers describe a situation, in which they can't proceed with their work and don't make any progress toward a goal. It was also mentioned by four participants, that one runs out of ideas or options for solutions before they would call it stuck. Three of them mentioned time as an additional factor because they said that one must be at a problem for a longer time before one is stuck. Interestingly the four members, that gave an example all described a situation, in which they were coding.
There were different reasons listed, which can lead to such a stuck phase. All six mentioned the possibility that one has not enough information, five also mentioned the option of not having enough knowledge or experience about the technical aspect of the problem, and one member of the study also included concentration problems as a reason.

"I would define stuck as when I'm working on a problem, for example programming, and don't make any progress. Maybe because I'm not focused or I don't know what step I have to do next, because I have to solve a problem which I can't solve without help, or I may need more information." -Participant 2

A brief definition of being stuck for a software engineer could be the following: A software engineer can be labelled as stuck, if she or he is working on a software engineering problem for a longer time, has tried different options, which didn't work, lacks information or lacks knowledge and can't make any progress towards the goal of the task.

5.2 Characteristics of Stuck Phases

This section is focusing on the reasons, frequency, and duration of the stuck phases. It uses the answers from the stuck reports and from the interview to find reasons for software engineers to be stuck.

5.2.1 Reasons for Stuck Phases

There were 24 stuck reports with answers to the question of why they were stuck. We categorised the answers in Table 5.1.

Category	**Number of Mentions**
Lack of knowledge	8
Lack of information	6
Unknown error	5
Unexpected behaviour	3
New technology	2

Table 5.1: Reasons for being Stuck

The reasons "Unknown error", "Unexpected behaviour" and "New technology" are also partially related to "Lack of knowledge". "Lack of information" refers more to a situation where the software engineer doesn't know exactly what he has to implement.

"I don't have any description from the client where to put the menu bar and what style of menu bar he wants." -Participant 2

The reasons given by the participants show that a frequent reason for being stuck is that the software engineer doesn't have the required knowledge or information.
The answers from the interviews also suggest, that a lack of knowledge or information is a frequent component of being stuck, as it was mentioned in a form by every participant. It was often phrased as "not knowing how it should work" or "not knowing how to proceed". They also mentioned "being out of options to try" as a reason which was labeled the same as "not knowing how to proceed" and also that they "don't know where to get better information".

"Nothing I try works and Stack Overflow let me down." -Participant 4

5.2.2 Frequency of Stuck Phases (RQ1.2)

In Table 4.2, one can see how the 24 stuck phases are distributed over the study members. It is interesting to see, that participants 2 with 8 stuck reports and 4 with 12 stuck reports had significantly more stuck phases than the others. Participant 2 had reported 5 different stuck phases over a time span of only 5 hours and 16 minutes. Participant 4 has his 12 stuck phases distributed more evenly over the complete time span of the study.
In Table 5.2 the average time between stuck phases of each participant is listed. It was calculated by dividing the time the participant wasn't stuck with the number of stuck phases. All of them said that the frequency varies a lot depending on the task and if it is something new or if they already have experience with the technology.

"This depends a lot on what I'm doing. When I am coding on a new project with a new tech stack I am stuck all the time but when I know what I have to do it is maybe once a day, but it depends." -Participant 5

Three of the members of the study said that they are stuck one time a day when they are coding. Another one said it is around four times a week, one said between six to eight times a week and the last one said it is about once or twice a day. These estimated frequencies don't match the number of reported stuck phases for most participants, but this could be, because of the tasks they had to do during the time of the study.

Participant	**Average time between stuck phases**
1	35h 17m 28s
2	00h 26m 59s
3	08h 25m 15s
4	02h 48m 05s
5	No stuck phase reported
6	01h 29m 21s

Table 5.2: Frequency of Stuck Phases

The average time between stuck phases differs a lot. When the participants were asked in the report during which tasks they were stuck, in 14 cases they wrote they were stuck during coding, testing, or debugging. Only participant 4 reported eight times, that he was stuck when writing the documentation.
During the pre-study questionnaire and the interview when the participants were asked how often they were stuck, their answers were rather close together.

Duration of Stuck Phases

The reported duration of the stuck phases varied for every participant. In Table 5.3 the minimum, maximum and average duration of the stuck phases for every participant are listed.

Participant	Number of Stuck Phases	Minimum	Maximum	Average
1	1	06m 13s	06m 13s	06m 13s
2	8	07m 25s	25m 37s	18m 02s
3	1	11m 13s	11m 13s	11m 13s
4	12	05m 54s	30m 26s	14m 52s
5	0	-	-	-
6	2	03m 35s	10m 26s	07m 01s
TOTAL	**24**	**03m 35s**	**30m 26s**	**14m 29s**

Table 5.3: Duration of Stuck Phases

It is interesting to see, that the average duration of stuck phases of participants 2 and 4, which used the tool at work, are significantly longer than the average time span of the other study members, which used the tool for private or school projects.

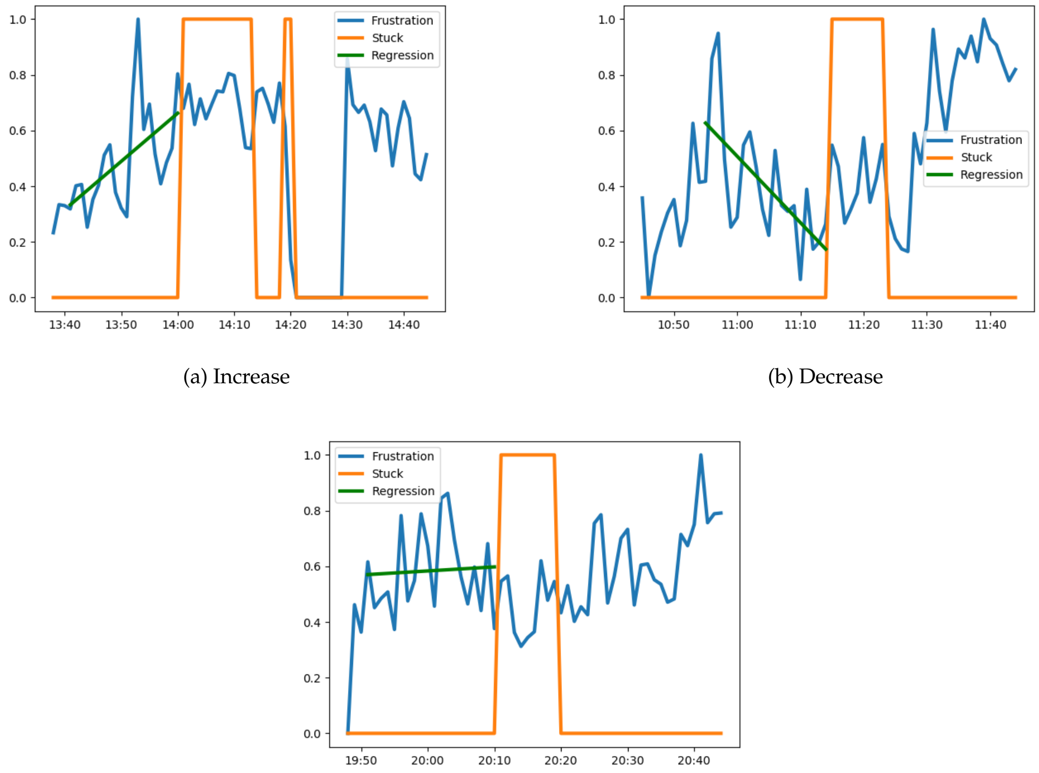
"When I'm coding for myself I sometimes just end the session when I got stuck. I sometimes don't want to deal with the problem in my free time and after some days it is often easier to go back and solve it."
-Participant 6

5.3 Behavioural Changes Before a Stuck Phase (RQ2)

This section covers different possible changes in behaviour the data of a participant could show before or during a stuck phase. To achieve this we looked at different time frames of the stuck moments. The base frame takes only the time into account when the study member was stuck. The other frames take also five, ten, and twenty minutes of data before the stuck phase began into account. We made a linear regression of the data of these periods and decided that the 20 minutes time frame before a stuck phase gave the most meaningful result as it was not that much influenced by the statistical outliers.
Unfortunately, not all stuck phases have user data because the session wasn't started at that time. This is the case for two of the 24 stuck moments. Another problem is that the stuck phases which are close together overlap with the larger time frames, so we can't analyse every stuck moment with the twenty minutes of additional data.

5.3.1 Face Recognition and Head Movement

A stuck phase is often a frustrating experience for a software engineer [MF15, GFWA17]. The participants were asked how frustrated they were on a scale from 0 to 10, where 0 means they were not frustrated at all and 10 is equal to being completely frustrated. The lowest value one participant gave was 3 and the highest was 9, with an average value of 6.5, which indicates that being stuck correlates with a higher frustration. That is why we checked the data from the facial recognition on how the values of the markers, which can be associated with frustration, behaved. These action units are AU04 and AU14 [GWB$^+$13a]. We didn't look for correlations with other emotions or markers. But there can also be other facial expressions, which could indicate, that a user was frustrated and only because the markers have a high combined value, it does not have to be the case, that the user was frustrated. We classified the data from the frustration value for all stuck phases in different tendencies which are shown in Figure 5.1, where the blue lines represent the level of frustration based on the marker values and the orange line indicates the reported stuck phases. If the value of the orange line is one the participant was stuck at this time. With the green line, we showed the linear regression of the data from 20 minutes before the start of the stuck phase.



(a) Increase

(b) Decrease

(c) No tendency

Figure 5.1: Examples of Tendencies from Frustration Marker Data

All stuck phases where we observed an increase of more than 10% over a time period of 20 minutes were labeled as "Increase". The stuck phases with an observed change of more than -10% over a time period of 20 minutes, for example -15%, were labelled as "Decrease". Every stuck phase with a change between 10% and -10% was labelled as "No tendency". Stuck phases where no data was available were categorised as "No Data". The result of the complete classification is shown in Figure 5.2

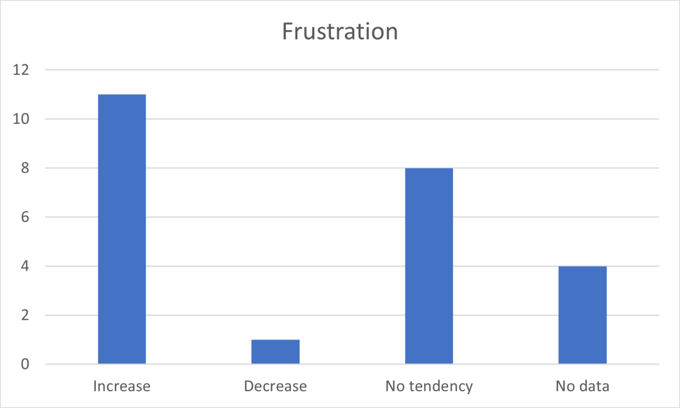


Figure 5.2: Results for Classification of Frustration Data

There were 11 cases where one could see an increase in the value of the frustration markers but there were also 8 stuck phases where no clear tendency was observed. In one case there was even a decrease before the stuck phase. In 4 cases, there was no data available. Two of them, were because no session was started and no data was collected and the other two, were because the camera was not active. On average the data from the frustration markers showed an increase of 24% over the time of 20 minutes before the stuck phase. The biggest increase was 183% and the biggest decrease was -72%. Most of the time, one could see a change in the behaviour of the data around 5 to 25 minutes before the start of the stuck phase. Interestingly, the value during the stuck phases sometimes rather declines than increases, which is caused by fewer available data points, either because no face was detected or the user was not in front of the camera.
The results of the classification of the head movement data, shown in Figure 5.3, do not show any clear tendency of change before a stuck phase. We used the same thresholds for the classification as before. The groups were divided into more than 10% change for "Increase", more than -10% change like -15% for "Decrease" and all change rates in between as "No tendency". There were 5 cases, where an increase was observed, 3 cases that showed a decrease, and 12 cases with no clear tendency. The average change rate was an increase of around 5% during the 20 minutes before a stuck phase. The biggest increase was 81% and the biggest decrease was -21%. During some stuck phases, the head movement value dropped, but this is for the same reason as for the decline of the value for frustration listed above, namely because the face recognition had gathered fewer data.

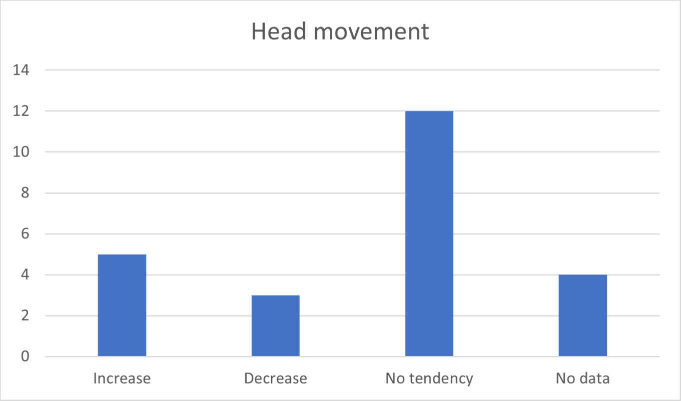


Figure 5.3: Results for Classification of Head Movement Data

5.3.2 User Input Activity

The user input data consists of data about key strokes, mouse clicks, and mouse movement. The tool also gathered data about the amount of scrolling a user did, but because there were only very few data points with scroll data values other than 0, no observation could be made about the relation between scrolling and being stuck. We used the same classification and the same thresholds for the analysis of all user input activity as before in the sections about frustration and head movement (see subsection 5.3.1). The groups were divided into more than 10% change for "Increase", more than -10% change like -15% for "Decrease" and all change rates in between as "No tendency".
The observation we could make about the behaviour of the key strokes was, that in 12 cases, there was a decrease in the number of keys pressed before a stuck phase. There were also 3 cases with an increase, 7 with no tendency, and 2 without data, as shown in Figure 5.4. The number of key strokes showed on average a change of -23% in the 20 minutes before a stuck phase. The biggest increase was 32% and the biggest decrease was -98%.

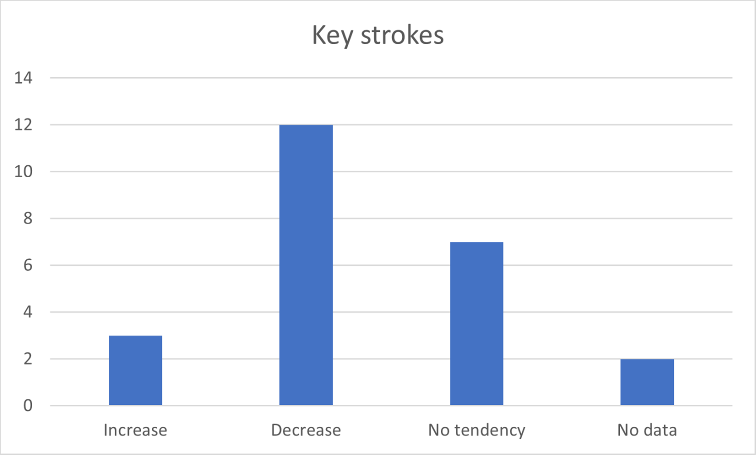


Figure 5.4: Results for Classification of Key Strokes Data

For the number of mouse clicks before a stuck phase, we got a similar result, as with the key strokes. In 14 cases, the number of mouse clicks decreased, in 5 cases it increased, in 3 cases there was no tendency and in 2 cases, there was no data available as shown in Figure 5.5. The number of mouse clicks showed on average a change of -25% in the 20 minutes before a stuck phase. The biggest increase was 42% and the biggest decrease was -93%.

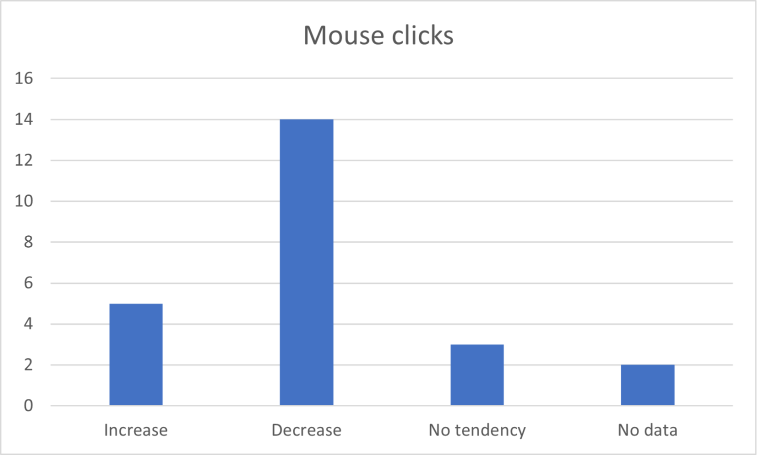


Figure 5.5: Results for Classification of Mouse Clicks Data

Like the two factors of user input above, the results for the mouse movement indicate, that the user input activity decreases before a stuck phase. As shown in Figure 5.6, we observed 14 cases with a decrease, 3 with an increase, 5 with no tendency, and 2 with no data. The amount of mouse movement showed on average a change of -22% in the 20 minutes before a stuck phase. The biggest increase was 31% and the biggest decrease was -92%.

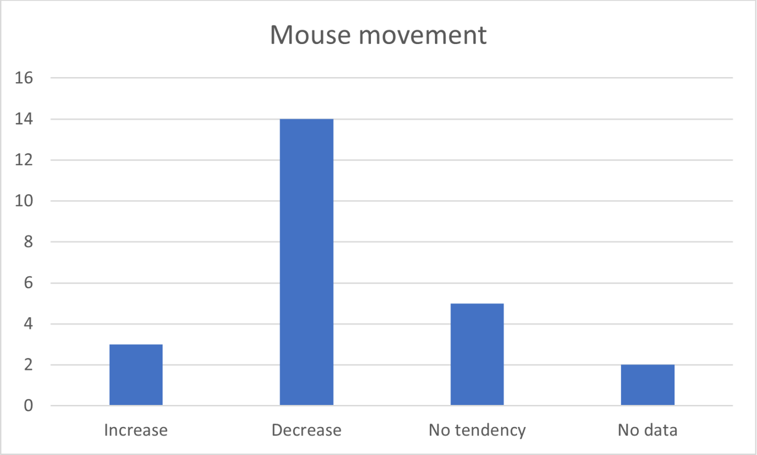


Figure 5.6: Results for Classification of Mouse Movement Data

5.3.3 Window Changes

In Figure 5.8, we can see that there were 16 cases, in which the frequency of window changes increased before a stuck phase, 1 case in which it decreased, 5 cases with no tendency, and 2 cases with no data. Linear regression was not suitable for the analysis of window changes because most of the time the value of window changes per minute was either zero or one and because there are spikes in the number of window changes around stuck phases and task changes the rate of change was very high. It was more suitable to analyse the tendency by comparing the absolute numbers of changes during the 20 minutes before a stuck phase and the number of changes in the time before these 20 minutes. Some examples of such graphs are shown in Figure 5.7. The blue line represents the window changes and the orange line shows if the user was stuck.

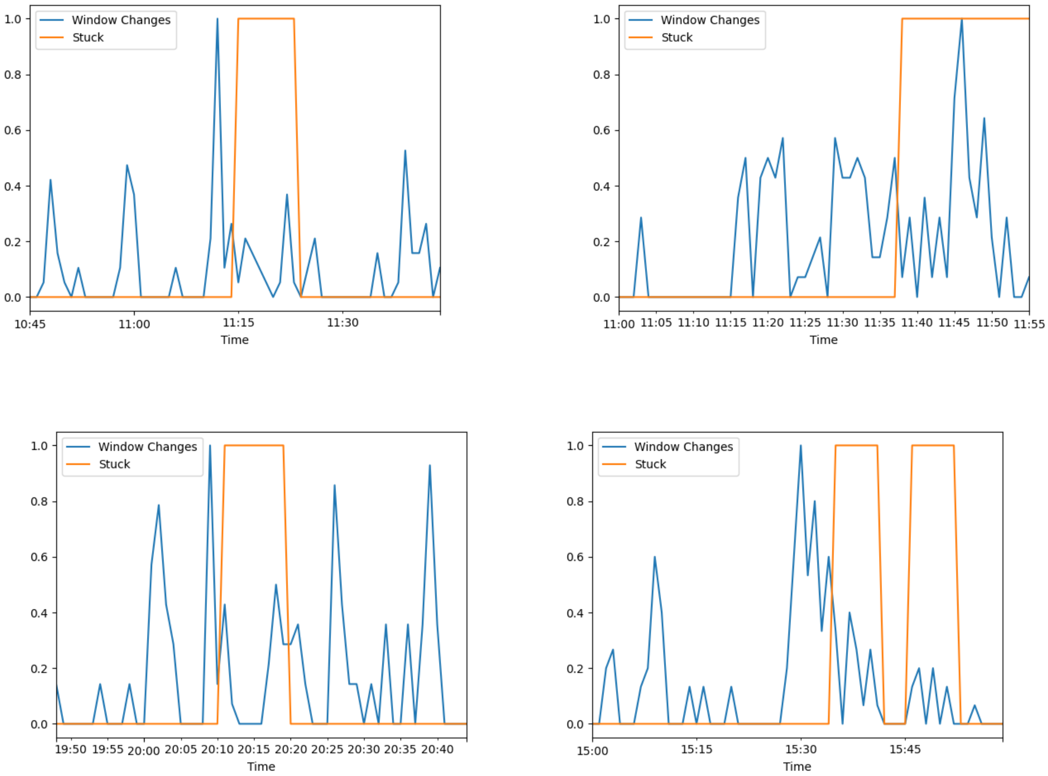


Figure 5.7: Examples of Window Changes Data

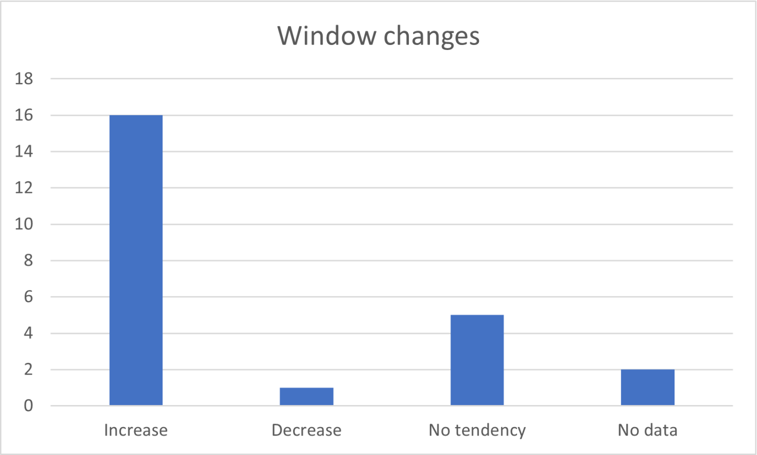


Figure 5.8: Results for Classification of Window Changes Data

5.4 Possible Solutions for Stuck Phases

Because there can be different reasons for a stuck phase, there may also be different solutions, and not every possible solution is a good fit for every stuck situation. The participant could state in the stuck report, what they would need or what could help them to overcome the stuck phase. Most of these answers included getting a better or another source of information, which could either be a co-worker, a boss, better documentation, a clearer task description, or something else. Of the 24 stuck phases, 23 were mentioning a form of getting information as a solution for their problems. The only other mention of a different solution was taking a break.

"Documentation is not complete and nobody knows how it works." -Participant 2

When the participants were asked during the pre-study questionnaire and the interview what they do, when they are stuck and what could help them, they mentioned that speaking to a colleague about the problem, reading the documentation, taking a break, or even stopping working on this project for the day, were options they frequently used. To the question asked whether they could imagine using rubber ducking, four of them considered it for future stuck phases and the other two wouldn't use it. In addition, concerns were expressed, because it would make a shared working environment too noisy and they would prefer talking to a colleague to get a second opinion. One of the participants said, that he thinks this technique would not work for him.

"I think it works what I did so far, like talking to colleagues or taking a break and I wouldn't know what else would be helpful. I don't think rubber ducking would help me." -Participant 6

5.5 Requirements for Stuck Related Tools

There are a variety of requirements for tools, which gather data or should detect stuck phases. Some of them are based on answers from the interviews with the participants, others are learnings from the data.

5.5.1 Data Gathering Tool (RQ3.1)

It is important for a data gathering tool, that it does not intervene with the work a participant is doing. The hourly reminder, which asked if they were stuck in the last hour was taken positively with the addition, that the user should be able to adjust the frequency of the pop up. The ability to start and stop a session by themselves was an important feature for all participants, even though that can yield less data, as some participants sometimes forgot to start a session. The tool had a reminder built in for that, which used a pop up when the user was working with an IDE for some time but has not started a session. Because a software engineer also uses other programs during his work, there was not a reminder for every occasion when a participant forgot to start a session. The graphs which showed the mouse and keyboard activity and the active processes got good feedback from the participants when asked about it in the interview because it helped them to make a more precise statement when they were stuck.

"It [the graphs with the activity of the last hour] *was helpful and very interesting even when I wasn't stuck. Just to see my own behaviour and so and how it varies over time. Also seeing when the activity started to get lower, really helped me to see when the time was I began to slowly get stuck."*
-Participant 5

They were also interested in the data on their normal behaviour during working sessions. One important point for all participants was to have the data only stored locally so that they can easily find the data file and look at the data which was possible with the "Show data file" button. The data stored in the data file was completely anonymous and the data from the face recognition can't be used to make statements about the user or the user's face.
There are some aspects of the data gathering sensors, that one must keep in mind when designing such a tool. If, for example, a virtual machine is primarily used for work, a window activity tracker can't detect the different windows, processes, or activities inside the virtual machine. If the tool is used to primarily gather data during the stuck phases, one must keep in mind that reporting the stuck phase in the tool and answering the questions alter the data during these times and that this data has to be filtered out. When a camera is used to gather sensors, it is necessary to make sure, that the camera is of a high enough quality and that the lighting in the physical work environment is suited for face recognition. The activity tracker was also raising questions for the participants, as they were afraid it could be a key logger, which would also detect, what key exactly was pressed. This would be a complete dealbreaker for all the participants.
In general, the privacy and what was tracked exactly were major concerns of the participants. They wouldn't use such a tool if it wasn't provided by a completely secure and trustworthy source. There were also concerns about enterprises, which could use such tools to monitor the productivity of the workers. Two of the participants said that firms, which would use such tools are at a large disadvantage for them if they would offer a job similar to that of a firm without such a tool. Thus, it is important for them, that the data from these tools would only be used for research purposes.
When looking at the performance of the program and the used resources, no participant had problems running the software on their device. One participant, who used a laptop reported, that the battery ran out quickly and the device got a little bit hotter than usual, which makes sense,

as the tool works all the time and uses face recognition with 30 pictures per second and writes to the database every second. Still, the program doesn't cause the performance of other programs to noticeably worsen.

One important insight the data and the interviews gave was that it is a good idea to differentiate between data from different tasks. If a user would specify for each session, if they were coding, debugging, testing, writing documentation, or doing something else, one would have more precise data to make different models for all these different tasks related to software engineering.

5.5.2 Stuck Detection Tool (RQ3.2)

Many of the requirements for a data gathering tool, which are listed above in Subsection 5.5.1, also apply to a stuck detection tool. One important difference is that it shouldn't store data for a longer period of time, namely, it should only store it as long as it is needed for real-time analysis, to check if the user is stuck.

"It's a strange feeling when the camera is on the whole time but as long as the program doesn't save the data and only uses it to detect the stuck phases, I would use such a stuck detecting program." -Participant 1

All participants wouldn't use such a tool if data would be sent to the firm or their boss. They only want to be notified, when they are stuck and four of them would like to get some options, on how they could proceed.

The camera is one of the sensors, which the participants were the most critical about. Some of them have their webcam lens covered with tape, in their normal work settings and only removed it for the study and for meetings. It could be an option to make a stuck detection tool, where each user can choose, which sensors should be used. This would maybe lower the effectiveness and accuracy, but it could raise the acceptance to use such a tool.

Chapter 6

Discussion

This section covers further insides of the topic with the thoughts and ideas of the researchers. It covers possible reasons for the made observations and the subjective valuation of the different sensors based on the presented data from Chapter 5. There is also a section about the practical use of the findings of this work and it includes possible threats to validity and an outlook for future works on the topic of stuck phases of software engineers.

6.1 Possible Reasons for the Observations

The number of reported stuck phases varied a lot between the different participants. This could have several reasons. One could be that these participants had a lower threshold to report themselves as stuck in comparison with the other participants because being stuck is a subjective feeling. Another reason could be that these participants reported one larger stuck phase as several smaller ones, so every time they tried something new, they would report themselves as not stuck anymore, but if the option they tried didn't succeed, they would report themselves as stuck again. But it could not only be because of the two reasons listed above but also because different fields of software engineering could have different frequencies of stuck moments.

There was also a difference in the length of the stuck phase between participants that used the tool at work and the others who used it at home for their own projects. The workers had on average longer stuck phases. These could be due to the fact how the participant dealt with being stuck because the participant who used the tool in private said that they sometimes ended the programming session when they were stuck and tried it at another time. This isn't necessarily an option for the workers, so they rather must find a solution than postpone the session. But because we only had six participants this result is not reliable.

During stuck phases, the values for frustration markers and head movement sometimes showed a tendency to decrease. This is partly due to the fact, that fewer data points had a value for the action units. This could be because the user had more head movement and the facial recognition had to recalibrate the landmarks or that the user was leaving the workspace and thus no facial recognition data was collected during these times.

6.2 Possible Ways to Detect Stuck Phases

As shown in Chapter 5, the behaviour of the data from the different sensors differs a lot in respect of how they change before a stuck phase. One sensor which doesn't seem to be well suited for such a prognosis about stuckness was the head movement sensor. The data from the head movement nearly showed no change before a stuck phase. Thus in this study head movement didn't indicate a stuck phase.

The frustration data could be used as an indicator for stuck phases, but it was observed that the increase in frustration doesn't only occur before, but also during the stuck phase. This could mean that this sensor alone is not the best solution when the goal is to detect stuck phases before the participant herself or himself feels stuck.

The user input activity data looks like it is well suited for detecting stuck phases, as one could see in the data that there is a decrease in the user activity before a stuck phase. This includes key strokes, mouse clicks, and mouse movement. These sensors don't need much computational power compared to facial recognition and seemed to have a higher forecast power than the data from the frustration markers or from the head movement.

It seemed that a good indicator for a stuck phase is the number of window changes a user does. The number of changes generally increased before a stuck phase. But this sensor alone isn't enough for reliably detecting stuck phases, because when starting a new task or changing from one task to another, there are also more window changes. To differentiate these two situations, we could use the other sensors like user input activity trackers.

In general, a good approach for detecting stuck phases is probably a combination of several trackers. The most important ones are a window activity tracker and a user input activity tracker. These are rather simple to implement, don't need much computational power, the data is easy to interpret, and they have a high prediction power for stuck phases. In addition, one can use face recognition to measure the frustration, but it is harder to implement, cost a lot more computational power compared to the other trackers, the data is harder to interpret and in this study, the prediction power was lower than the prediction power of the other sensors.

6.3 Practical Use

The learnings from this research paper can be used to design a stuck detection tool for private use or for a firm to utilise for improving productivity and reducing frustration. It might not be suited for every software engineer, but there was interest from four of the six participants for such a tool. The goal of a stuck detection program would be to detect a stuck phase, make the user aware of it, and maybe give her or him possible solutions to overcome this specific stuck phase. These possible solutions for example can be a platform for rubber ducking, a reminder to take a break or to ask a colleague, or a proposal to change the task. If it works as intended, it could save time and lower the frustration level of the software engineer. This saved time could yield an increase in productivity or output, which is interesting for firms with the goal to lower cost or work time for a project.

6.4 Future Work

The approach used for this study worked well and is suitable for future studies in this field. Some features like the frequency of the pop ups should be adjustable by the user. With the findings of this paper, one could design a better data gathering tool for stuck phases, which could include more sensors, give better live feedback of the gathered data, has a higher collection frequency, or has improved performance. Such a tool could be used to make a larger field study with a firm for a longer period. A future study could also take a closer look at other emotions and facial expressions to see if their behaviour changes before or during a stuck phase. There is also the potential to look closer at stuck phases and make a more granular classification of them to see if they behave differently.

It would also be possible to create a stuck detection tool with the sensors covered in this research paper or make several with different combinations of sensors and compare the effectiveness. Another possibility would be to combine both approaches and have a data gathering stuck detection tool which would improve its model for every user and task based on the live feedback of the user about them being stuck.

6.5 Threats to Validity

Most of the findings from this research paper are based on the data from the six participants of the user study. This is a low number of participants and yields only a small amount of data for a rather complex topic. The findings from the user data of the study are only tendencies of how the data from different sensors change before a stuck phase. It was not tested if these tendencies can be used to detect stuck phases. The small amount of data and the fact, that 20 of the 24 stuck phases were from only two different participants increases the chance for a bias in the data and that the seen trends could be caused by chance or special behaviour of the participants and that there is no or not such strong causation as the data indicates. This could be described as a sampling bias.

Participant selection is also a threat to the validity of this study as all participants were recruited out of the private network of the researcher and shared roughly the same age and most of them had similar education. There was also no differentiation between the different tasks of a software engineer, which can give very different data and may not be analysed as the same.

Another possible threat to the field study is the Hawthorne effect which states that participants change their behaviour because they know they are being studied. This was even implied by one of the participants when asked if the use of the tool influenced his work.

"I have a light on the laptop which is on when the camera is active. Every time it was on I was aware of it. I was aware that I was recorded and that has influenced me a bit. In my subconscious, I had the feeling I am actively watched at the moment and that irritated me." -Participant 4

Chapter 7

Conclusion

To gather data about the characteristics of stuck phases of software engineers, we developed a tool, which can be used to report stuck moments and collects data from face recognition, user input, and window activity. This tool was used in a study with six participants, which consisted of a pre-study questionnaire, a two-week usage of the tool, and a final interview.

We found that the data from the user input activity tracker and the window activity tracker showed the most significant and consistent tendency of change in behaviour before a stuck moment compared to the baseline of each user. Before a stuck phase started, it could be observed that the user input activity declined and the number of window changes increased. This is an indication, that a combination of the data of these sensors could be suitable for detecting when a software engineer is stuck.

The data from the interviews confirmed, that there is a general interest in a stuck detection tool. But there are several requirements for data gathering and stuck detecting tools. The most important factors for the participants were the privacy and anonymity of the data and that these tools won't be used for monitoring them or their work in any form. The camera was the sensor, which the participants were the most sceptic about and it did not yield as good results for characterizing stuck phases as the other sensors, thus it may be considered optional for such tools in the future.

Future studies could focus on additional sensors or on combinations of sensors to detect stuck phases. Such a tool could be well suited for a larger user study with a firm, as there would be potential to save time when working on a project.

Appendices

Appendix A

Consent Form

University of Zurich^UZH

Stuck Detector User Study

Thank you for participating in this user study.

Purpose

The purpose of this study is to find out if it is possible to detect moments of stuckness during software engineering, with the data from the user interaction (keyboard and mouse activity), with the program activity and with the data from facial recognition. Another aspect of this user study is to find out, how such a data collection program should look like for a larger study.

Study Procedure

1.) There will be a written pre-study interview send before the study start.
2.) During the two weeks from Monday, the 11. July 2022 to Sunday, 24. July 2022, participants will use the Stuck Detector to collect data and report on stuck moments.
3.) There will be an after-study interview, where the audio is recorded for transcription reasons.

Data, Storage and Confidentiality

If a session is started per button click, the program collects a datapoint every second. A datapoint consists of:

- time of measurement
- data from the input tracker (keystrokes, mouse clicks, mouse distance moved, scroll data)
- data from the windows activity tracker (window title, process, activity)
- data from the facial recognition tool (facial landmarks, head position)

You can find more information about the different external tools used on these sites:

- Personal Analytics input tracker and windows activity tracker:
  https://www.ifi.uzh.ch/en/seal/people/meyer/personal-analytics.html

- Openface 2.0 for facial recognition:
  https://github.com/TadasBaltrusaitis/OpenFace/blob/master/README.md

All data is only stored locally in a .sqlite3 file, which the participants can upload to a data sharing tool after the study.

Interview Data

The pre-study interview is in a written form. The post-study interview is in a normal interview setting via an online meeting tool. The audio of this interview is recorded only for transcription and will be deleted after the transcription of the interview.

Contact for Information about the Study

If there are any questions or comments concerning this study, please don’t hesitate to contact Samuel Brügger (samuelandreas.bruegger@uzh.ch).

Participant Consent Form

User Study of Stuck Detector - Consent Form

Please tick the appropriate boxes	Yes	No
Taking Part in the User Study		
I have read and understood the “Stuck Detector User Study and Program Information”, dated 11.07.2022, information sheet or the User Study has been fully explained to me. (If you will answer No to this question, please do not proceed with this consent form until you are fully aware of what your participation in the User Study meant.)	□	□
I have been given the opportunity to ask questions about the User Study.	□	□
I agree to take part in the user study. I understand that taking part in the user study will include completing self-assessments of stuck moments, being tracked, including the data from the facial expression, the keyboard (only stroke frequency) and mouse activity, the process activity, program usage* and being interviewed, which will be audio recorded for transcript purposes. Note that the audio record files will be deleted immediately after the transcription. *Different actions of the program such as starting, exiting, button presses, etc. are tracked. You are asked to send the data in form of a .sqlite3 file to samuelandreas.bruegger@uzh.ch. Any personal information is anonymized.	□	□
I understand that my taking part is voluntary and that I can withdraw from the study at any time before 24.07.2022; I do not have to give any reasons for why I no longer want to take part and there will be no adverse consequences if I choose to withdraw.	□	□
How my information will be used during and after the User Study		
I understand my personal details such as name, phone number, address and email address etc. will not be tracked, used nor shared to people outside the User Study.	□	□
I understand and agree that my words may be quoted in publications, reports, web pages, and other research outputs. I understand that I will not be named in these outputs unless I specifically request this.	□	□
The information you provide can be used legally by the researchers		
I agree to assign the copyright I hold in any materials generated as part of this User Study to The University of Zurich.	□	□

Name of participant Date Signature

Name of researcher Date Signature

User Study – contact details for further information:

Bachelor Student

- Samuel Brügger, samuelandreas.bruegger@uzh.ch

Supervisors

- Alexander Lill, lill@ifi.uzh.ch
- Roy Rutishauser, rutis@ifi.uzh.ch

Appendix B

Study Information

Stuck Detector User Study and Program Information

Thank you for participating in this User study.

Stuck Detector is a tool to gather data of stuck phases of software engineers, especially the facial expressions and the activity of the user before and during these phases.

Data collection and storage
The collected data consists of data of facial expressions, head pose, mouse and keyboard activity, which window was focused and what process was active. The program also collects data about the reported stuck phases and about the usage of the program, for example the use time and the button presses.
The data is only collected locally. User data will only be collected when the user has started a session. The data of the use of the program is always collected when the program is running.

Every hour you get a reminder popup, which asks if there were a potential stuck moment. You can see your collected activity data and the used processes of the last hour in the session view.

If the program is started, but there is no session started even though you are working with an IDE, there will be a pop up to ask you if you want to start the session.

If you want to look at the data, you can use the free online service SQLite Viewer at:
https://inloop.github.io/sqlite-viewer/

Installation
You need to download the setup file and follow the installation guide.
The setup file can be found here:
https://drive.google.com/file/d/1WQUVsgh6ZXIBc_VP6LLoFpbh8B6breeW/view?usp=sharing

There may be a windows security warning, which you can ignore.
You need to give the program access to your camera.
You need to run the program as administrator.

Study
After installation, you could add the program to "AutoStart". This way, you don't forget to start the tool before working. After starting up the program, you can press the button "Start Session" to start the data gathering of the tool. All buttons are explained in the app itself under the "Info" view. You can use the button "Show Data file" in the menu bar to get the data file.

Additional Information
Should there be any bugs stopping you from using the tool efficiently, please report them to me.

Appendix C

Pre-Study Questionnaire

Pre-Study Questionnaire

1.) How would you define "being stuck"?

2.) Do you realize it in the moments when you are stuck, and if so, how exactly?

3.) How many times per day/week are you stuck?

4.) Do you think it is easier to recognize stuck phases in the moment or in retrospect?

5.) To what extent would it be helpful if a program pointed out such stuck moments to you?

6.) What helps you in stuck moments so far?

7.) What would help you additionally in stuck moments?

8.) How could a program help you to recognize stuck moments?

Appendix D

Final Interview

Interview Questions User Study Stuck Detector

RQ1.1 How can you define "being stuck" in software engineers?
RQ1.2 How often are software engineers stuck?

RQ2: To what extent is it possible to detect stuck phases of software engineers using data from face recognition and computer interaction?

RQ3.1: What are the requirements for a program to be suitable for collecting such data?
RQ3.2: What are the requirements for a program to recognize stuck phases?

In general, about being stuck: (Pre-Study Questionnaire)

1.) (RQ1.1) How would you define "being stuck"?
2.) (RQ2) Do you realize it in the moments when you are stuck, and if so, how exactly?
3.) (RQ1.2) How many times per day/week are you stuck?
4.) (RQ2) Do you think it is easier to recognize stuck phases in the moment or in retrospect?
5.) (RQ3.2) To what extent would it be helpful if a program pointed out such stuck moments to you?
6.) (RQ3.2) What helps you in stuck moments so far?
7.) (RQ3.2) What would help you additionally in stuck moments?
8.) (RQ3.2) How could a program help you to recognize stuck moments?

After the user study:

9.) (RQ2) In what ways, if any, are you more aware of being stuck after the user study?
10.) (RQ2) How would co-workers recognize that you are stuck?
11.) (RQ2) Were you able to capture all of the stuck moments?
12.) (RQ2) How accurately could you give the start and end times of the stuck phases?
13.) (RQ3.1) How did you feel about the hourly reminder?
14.) (RQ3.1) To what extent was the graph with the info of the last hour's activity helpful in reflecting on whether or when you were stuck?

Use of the program:

15.) (RQ3.1) How did you feel about the operation of the program?
16.) (RQ3.1) What did you like or not like about the program?
17.) (RQ3.1) Did using the program interfere with your work?
18.) (RQ3.1) Would you change anything about the program?
19.) (RQ3.2) Would you voluntarily use a program at work or in private that collects such data and alerts you to possible stuck moments?

Personal data:

20.) (RQ2) Which tasks did you have during the Study?
21.) How old are you?
22.) What do you do for a living?
23.) How much work experience in software engineering do you have?
24.) How good was your webcam and the lighting conditions?
25.) Do you have any other comments or questions?

Bibliography

[AKN+19] Y. Abdelrahman, A. A. Khan, J. Newn, E. Velloso, S. A. Safwat, J. Bailey, A. Bulling, F. Vetere, and A. Schmidt. Classifying attention types with thermal imaging and eye tracking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(3):1–27, 2019.

[BC06] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, 2006.

[BDO+16] N. Bosch, S. K. D'mello, J. Ocumpaugh, R. S. Baker, and V. Shute. Using video to automatically detect learner affect in computer-enabled classrooms. *ACM Transactions on Interactive Intelligent Systems*, 6(2):1–26, 2016.

[BP04] W. Burleson and R. Picard. Affective agents: Sustaining motivation to learn through failure and a state of stuck. *Workshop on Social and Emotional Intelligence in Learning Environments*, 2004.

[BS08] A. Begel and B. Simon. Struggles of new college graduates in their first software development job. *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 226–330, 2008.

[BSN+20] E. Babaei, N. Srivastava, J. Newn, Q. Zhou, T. Dingler, and E. Velloso. Faces of focus: A study on the facial cues of attentional states. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, page 1–13, 2020.

[BZLM18] T. Baltrusaitis, A. Zadeh, Y.C. Lim, and L.-P. Morency. Openface 2.0: Facial behavior analysis toolkit. *Proceedings - 13th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 59–66, 2018.

[CD09] J. Carter and P. Dewan. Automatically identifying that distributed programmers are stuck. *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, page 12, 2009.

[CD12] J. Carter and P. Dewan. Design, implementation, and evaluation of an approach for determining when programmers are having difficulty. *Proceedings of the 16th ACM International Conference on Supporting Group Work*, pages 215–224, 2012.

[GFWA17] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson. On the unhappiness of software developers. *In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE'17)*, pages 324–333, 2017.

[GWB$^+$13a] J. F. Grafsgaard, J. B. Wiggins, K. E. Boyer, E. N. Wiebe, and J. C.Lester. Automatically recognizing facial expression: Predicting engagement and frustration. *Proceedings of the International Conference on Educational Data Mining (EDM)*, pages 43–50, 2013.

[GWB$^+$13b] J. F. Grafsgaard, J. B. Wiggins, K. E. Boyer, E. N. Wiebe, and J. C.Lester. Automatically recognizing facial indicators of frustration: A learning-centric analysis. *Proceedings of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction (ACII '13)*, pages 159–165, 2013.

[HLH$^+$13] J. Hernandez, Z. Liu, G. Hulten, D. DeBarr, K. Krum, and Z. Zhang. Measuring the engagement level of tv viewers. *10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition*, pages 1–7, 2013.

[KBH13] I. A. Khan, W.-P Brinkman, and R. Hierons. Towards estimating computer users' mood from interaction behaviour with keyboard and mouse. *Frontiers of Computer Science*, 7(6):943–954, 2013.

[KBP07] A. Kapoor, W. Burleson, and R. W. Picard. Automatic prediction of frustration. *International Journal of Human-Computer Studies*, 65(8):724–736, 2007.

[MF15] S. C. Müller and T. Fritz. Stuck and frustrated or in flow and happy: sensing developers' emotions and progress. *Proceedings of the 37th International Conference on Software Engineering*, 1:688–699, 2015.

[UJ11] S. Uzzaman and S. Joordens. The eyes know what you are thinking: Eye movements as an objective measure of mind wandering. *Consciousness and Cognition*, 20(4):1882–1886, 2011.

[VAG$^+$15] M. F. Valstar, T. Almaev, J. M. Girard, G. McKeown, M. Mehu, L. Yin, M. Pantic, and J. F. Cohn. Fera 2015 - second facial expression recognition and analysis challenge. *11th IEEE InternationalConference and Workshops on Automatic Face and Gesture Recognition (FG)*, 6:1–8, 2015.

[WSL$^+$14] J. Whitehill, Z. Serpell, Y. Lin, A. Foster, and J. R. Movellan. The faces of engagement: Automatic recognition of student engagement from facial expressions. *IEEE Transactions on Affective Computing*, 5(1):86–89, 2014.

[ZMMF18] M. Züger, S. C. Müller, A. N. Meyer, and T. Fritz. Sensing interruptibility in the office: A field study on the use of biometric and computer interaction sensors. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2018.