

Automatic Conceptual Window Grouping with Frequent Pattern Matching

by

Anna Scholtz

B.Sc., Technische Universität Chemnitz, 2016

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

October 2019

© Anna Scholtz, 2019

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Automatic Conceptual Window Grouping with Frequent Pattern Matching

submitted by **Anna Scholtz** in partial fulfillment of the requirements for the degree of **Master of Science**
in **Computer Science**.

Examining Committee:

Reid Holmes, Computer Science
Supervisor

Thomas Fritz, Computer Science
Supervisory Committee Member

Abstract

While working, software developers constantly switch between different projects and tasks and use many different applications, web resources and files. These diverse resources are scattered across many windows and lead to cluttered workspaces that can distract developers in their workflows.

Having mechanisms to determine which resources belong together for working on a project, would allow us to develop tools that could support developers in organizing their work, declutter their workspace and switch between projects. Existing approaches in this area often either require users to manually define which resources belong together, or do not examine how users would group the resources themselves and how to best support them.

In this thesis we present an approach that automatically detects groups of applications and resources that developers use and are relevant to the tasks and projects they are working on. These groups are referred to as *Conceptual Groups*. The approach applies frequent pattern analysis on recorded interaction data and clusters these to retrieve conceptual groups. To measure the accuracy of our approach, we conducted a study with 11 participants and compared it to existing approaches which were outperformed by up to 50%.

Lay Summary

Software developers use many applications and files while working on different tasks and projects on their computers. Over time, they open more and more windows, tabs and applications which makes it harder to find the right window to switch to and potentially distracts workers. In this thesis, I describe an approach that automatically determines which applications and documents developers use and are relevant to the tasks and projects they work on. This approach can be used for various productivity tools for decluttering workspaces or navigating between different projects. To determine how accurately this approach can determine which applications and documents are related to a certain project, I evaluated it in a study with 11 participants and compared it to existing approaches.

Preface

All of the work presented henceforth was conducted in the Software Practices Laboratory at the University of British Columbia. All projects and associated methods were approved by the University of British Columbia's Research Ethics Board [certificate #H18-02647]

I was the lead investigator, responsible for the concept formation, data collection, analysis and manuscript composition. Reid Holmes and Thomas Fritz were involved throughout the concept formation and manuscript composition.

Table of Contents

Abstract	iii
Lay Summary	iv
Preface	v
Table of Contents	vi
List of Tables	ix
List of Figures	x
Acknowledgments	xii
1 Introduction	1
2 Related Work	4
2.1 Concepts	4
2.2 Context Detection	4
2.2.1 Manual Grouping	5
2.2.2 Automatic Grouping	5
2.3 Frequent Pattern Detection	5
2.4 Navigation Support	6
3 Pilot Study	7
3.1 Study Procedure	7
3.2 Study Support	8

3.3	Participants	8
3.4	Results	9
3.4.1	How Many Applications and Resources Are Usually Open?	10
3.4.2	How Much Time Is Spent in Active Windows?	10
3.4.3	Are There Frequently Recurring Interaction Patterns?	10
3.5	Discussion	12
4	Ground Truth Study	13
4.1	Study Procedure	13
4.1.1	Study Tasks	14
4.2	Study Support	14
4.3	Participants	15
4.4	Results	15
4.4.1	Group Size and Evolution	15
4.4.2	Group Composition	17
4.4.3	Frequent Patterns	17
5	Approach for Detecting Conceptual Groups	20
5.1	Event Data Logging	21
5.2	Pre-Processing Events	21
5.3	Frequent Pattern Analysis	21
5.4	Clustering	22
5.5	Application and Resource Assignment	23
6	Evaluation	26
6.1	Accuracy	27
6.2	Comparison to Related Work	29
7	Threats	32
7.1	External Validity	32
7.2	Internal Validity	33
7.3	Construct Validity	33
8	Discussion	34
8.1	Supporting Professionals with Conceptual Groups	34

8.1.1	Tool Prototype	34
8.2	Prototype Feedback	36
8.2.1	Accuracy Improvements	37
8.3	Future Work	37
8.3.1	Field Study	38
8.3.2	Applications	38
9	Conclusion	39
	Bibliography	40
A	Pilot Study Details	43
A.1	Recruiting Participants	43
A.2	Setup and Introduction Session	46
A.3	Final Survey	51
B	Ground Truth Study Details	54
B.1	Recruiting Participants	54
B.2	Setup and Introduction Session	55
B.3	Study Execution	61
B.3.1	Task 1 - Let's Go Travel	61
B.3.2	Task 2 - Step by Step Blockchain	63
B.3.3	Task 3 - Raytracer Documentation	80
B.4	Final Interview	83

List of Tables

Table 3.1 Pilot study: Total open applications, tabs, windows. 10

Table A.1 Pilot study participant metadata. 45

List of Figures

Figure 3.1	Pilot study: Pop-up for indicating recent workflow.	9
Figure 3.2	Pilot study: Usage duration of specific tabs and windows aggregated across all participants.	11
Figure 3.3	Screenshot of tool for visualising detected patterns from pilot data. The time-line on the top indicates the end and start of frequent patterns that have been detected. Frequent patterns consisting of the same applications and resources appear in the same row. Below is a log of every recorded event which was used for manual inspection. If events are part of a pattern then they get highlighted in a color that correspond to a frequent pattern and indicated in the legend on the right side.	12
Figure 4.1	Pop-up for self-reporting conceptual groups.	16
Figure 4.2	Frequent pattern occurrences (P3, P6, P8). Each row indicates the start and end of a specific pattern. Patterns consist of a specific set of applications and resources. Different patterns developed for different participants. Vertical lines mark project switches that participants worked on: T1 - blockchain, T2 - travel planning, T3 - raytracer.	18
Figure 5.1	Overview of the steps involved to detect conceptual groups.	24
Figure 5.2	Constraints for creating frequent patterns. Different patterns are indicated on a timeline based on their occurrence and denoted by different characters (A,B,X,...) Actual: patterns detected in the event data. (1) only different patterns, (2) the closest occurrences of patterns and (3) patterns with less than $c = 3$ events inbetween can be merged.	25

Figure 6.1	Accuracy of our approach for each response of each participant. “Correct”: data assigned to same groups as in ground truth, “Wrong”: incorrectly assigned data, “Missing”: data indicated in the ground truth but missing in the generated groups, “Likely Correct”: data that has been part of the group earlier or later but not at the time of the response. Vertical black lines mark the average correctness. Number on the left refer to the participant. Numbers on the right indicate the difference in number of generated groups vs. number of groups indicated by participants.	30
Figure 6.2	Boxplots of accuracies of related approaches for each response.	31
Figure 8.1	Prototype of our tool for displaying and interacting with conceptual groups. . .	35
Figure A.1	Pilot study consent form.	50
Figure A.2	Pilot study final survey.	53
Figure B.1	Ground truth study consent form.	60
Figure B.2	Travel planning task description.	62
Figure B.3	Spreadsheet template for the travel planning task.	63
Figure B.4	Blockchain implementation task description.	79
Figure B.5	Raytracer documentation task description.	82

Acknowledgments

I would like to thank my supervisor Reid Holmes and Thomas Fritz for their support and guidance throughout my research and writing process.

I would like to thank my family for supporting me from afar, and especially my brother Michael who always had an open ear for my complaints and helped me brainstorm various ideas.

I would also like to thank the members of the Software Practices Lab for providing me advice on my research and piloting my studies as well as the participants of all my studies for dedicating their time.

Finally, I must thank my friends I have made in Canada and across the globe for keeping me sane: most notably Puneet Mehrotra, Siddhesh Khandelwal, Hayley Guillou, Giovanni Viviani, Nico Ritschel, Neil Newman and many more.

Chapter 1

Introduction

Working on multiple projects and tasks at the same time and constantly switching between them is quite common for software development professionals in today's dynamic work environment [4, 11]. Previous work [14, 15] and results of a pilot study in which knowledge workers were monitored for up to 10 work days show that while working on their tasks, individuals interact with up to 60 different computer applications, access different web pages to seek information, and use various documents to store data. Over time, users open dozens of applications and resources which often leads to cluttered workspaces, making it more tedious to find the relevant resources, to resume previous work, as well as it increases the potential to get distracted [18].

To help with the organization of resources and workspaces, various approaches have been devised for grouping documents that are related to specific tasks. However, those either require users to manually define their groups [27], do not account for resources that are part of multiple groups [1], or do not evaluate how accurately these groups match with how users would want to group these resources themselves [22, 24].

The objective of this thesis work is to automatically detect groups of applications and resources from interaction data to support developers in switching between different projects and tasks and organizing their workspaces. We refer to these groups as *Conceptual Groups*. Conceptual groups consist of resources, such as documents or web pages, and applications as well as previous actions that were performed in the same context. Conceptual groups could be related to a certain project, which again can consist of different subtasks. For example, the conceptual group related to implementing a game for iOS might consist of Xcode, all created program files, a Terminal used for version control, a web browser and browsed websites and an iOS simulator. A conceptual group related to writing a scientific paper might be composed of a text editor, all LaTeX files the paper

consists of, a Terminal to compile the paper, and a PDF viewer. It is possible that some applications or resources are part of multiple different conceptual groups. For example, web browsers and web engines are often used in many different contexts to search for information. Conceptual groups potentially change over time as users start new subtasks which require different applications or resources.

In our work, we are interested in the characteristics of these conceptual groups and whether they can get detected automatically. For the automatic detection, we assume that developers use and switch between the same group of applications and resources repeatedly for a certain task or project and that we can use frequent pattern mining to detect the groups to support them in their work.

For our work, we address the following research questions:

RQ1 What are the characteristics and similarities of the way that different developers group their applications and artifacts?

RQ2 How accurately can conceptual groups be detected using frequent pattern analysis?

RQ3 How accurate is our model relative to approaches from related work in group detection?

To address these research questions, our main contributions in this thesis are a pilot study that investigates how users switch between applications and documents and whether certain action sequences are frequently repeated. Based on our initial insights, we developed an approach to automatically detect conceptual groups from recorded computer interactions and a study to collect ground truth to uncover how users group their applications and artifacts for their work projects performed, which are also used to measure the accuracy of our proposed method and to compare it to related approaches. We showcase a prototype tool that implements our approach and allows users to access their conceptual groups and to customize their computer workspaces. We also discuss a wide range of applications conceptual groups could be used for, such as tools for organizing and decluttering the workspace. Initial results suggest that frequent patterns can be used to extract conceptual groups from interaction data outperforming existing approaches by up to 50% in accuracy.

The rest of this thesis is organized as follows: we discuss related work in Chapter 2 and describe the result of our pilot study in Chapter 3 which we conducted as a first exploration step. Next, we explain how we collected ground truth data used to evaluate our approach in Chapter 4 and describe our approach in Chapter 5 followed by an evaluation of its accuracy in Chapter 6. We finish the

paper with a discussion section in which we comment on the conceptual group concept, describe our prototype and other application scenarios of conceptual groups and future work, commenting on limitations in Chapter 7, and a conclusion in Chapter 9.

Chapter 2

Related Work

Related work can be categorised into concepts derived by analysing user interaction data for grouping applications and resources and approaches to detect these groups.

2.1 Concepts

Previous work examined how knowledge workers group and structure their work by observing them or capturing their computer interaction during their work. Generally, these studies found that knowledge workers have groups of applications and resources that they use for completing their tasks. For example, Bannon et al. analysed computer interaction data and inferred that users work in *workspaces* that they defined as “[...] *tools and data relevant to the users’ goals* [...]” and “[...] *highly dynamic internal structures which can be modified as users reformulate their goals*” [4]. While workspaces just list “[...] *software tools that the computer system can provide for accomplishing these goals* [...]”, our concept of conceptual groups keeps a history of tools that users actually previously used while working to achieve their goals and is likely to be using again.

Similarly, Morteo et al. performed an observation study. Based on their observations, they coined the term *working spheres* [19] which are “[...] *higher levels of unit of work or activities that people divide their work into* [...]”, however, unlike conceptual groups, do not include the tools and applications necessary for completing this work.

2.2 Context Detection

Detecting the current context of what users are working on can help to mitigate problems such as finding related files [5], reducing distractions [18] and switching between tasks frequently [4, 11].

2.2.1 Manual Grouping

Most related approaches that support grouping of applications and resources require users to manually group either windows [27] or define working spheres [19] that consist of emails, contacts or files relevant to specific activities with a common goal. Users have to constantly review and adapt their groups when opening new windows or switching tasks which can get cumbersome, interruptive and time-consuming over time. A better alternative are approaches that automatically detect these groups.

2.2.2 Automatic Grouping

Some approaches that automatically retrieve the user's context rely on Bayesian networks but often require users to initialize the network, by manually assigning randomly chosen files to the right context [8] or only infrequently updating the groups [24]. Fully automatic approaches often only detect task switches [9, 17, 20, 26], but do not consider that there might be a hierarchy of tasks consisting of various subtasks and do not take into account that users switch back and forth between the same tasks.

Other fully automatic approaches count the number of switches between windows [1] or determine the semantic similarity of window titles and temporal proximity [22] to group windows. For determining the temporal closeness between windows, the number of switches between windows is counted and they are grouped together if they exceed a certain threshold. However, these approaches either do not take into account that the same resources and applications can be used across different groups and that switch patterns occur or require the number of groups to be detected.

Our approach is fully automatic in detecting groups, it is not necessary to define a specific number of groups that can be detected and requires no training.

2.3 Frequent Pattern Detection

More recent approaches detect frequently occurring patterns in workflows or routines to identify tasks [7, 25] using different techniques [12, 21] to extract recurring patterns from event data. Although these approaches return a set of detected frequent patterns and do not group or summarize them further, this temporal information could be used to detect conceptual groups since using the same applications and resources over and over again suggests that they belong together and could be seen as forming groups.

2.4 Navigation Support

Different window manager [6, 28] and navigation tools [23] have been introduced that visualize which windows belong to the same task or recommend windows that are related to the currently active window when switching between them. These tools determine relatedness based on previous direct switches between windows. While these tools show that having applications users can actually use is beneficial, none of the tools consider groups of applications and resources or use automatic grouping.

Chapter 3

Pilot Study

As a first exploratory step, we conducted a pilot study to gain insights into how users switch between applications and windows while working on different tasks. Our research questions were:

- How many and for how long do users actively use applications and resources during their work?
- How often do users switch between windows and applications?
- Are there frequently recurring interaction patterns?

To address these research questions we performed an exploratory field study with 5 participants over a period of 7 to 10 workdays, analyzing their user interactions and examining the use of frequent pattern matching on user interactions.

3.1 Study Procedure

In a short introduction session we explained participants the study, asked them for consent, and installed a tool on their work machines for monitoring all interactions. Once the tool was installed, we asked participants to continue their work as usual for up to 10 working days. During this time, our tool prompted users approximately every 20 minutes, asking them to report their current workflows in a popup. We asked about workflows, since we thought of them as frequently recurring sequences of steps or patterns that users perform during their work and wanted to investigate if we can detect them.

The maximum duration of the study was 10 working days, however the study automatically ended as soon as the participant provided 100 pop-up responses. In a short follow-up survey,

we asked about demographics and more generally about the workflows and workflow switches participants performed.

To validate frequently detected workflows, we conducted a followup survey with our participants, in which we presented frequently recurring interaction patterns that we extracted from recorded interaction data.

More information about the study, consent form and collected data can be found in Appendix A.

3.2 Study Support

For this in situ study, participants had to install a tool on their work machine, which was required to run macOS Sierra or higher, that collects the following data:

- Name, title and window position of active and background applications and idle times. Recorded every time made users changes to them.
- Aggregated cursor movements, clicks and number of keyboard presses. Individual key strokes were not recorded, only the number of keys pressed during a time window of 10 seconds. This prevents recording of passwords.
- Keystrokes for switching applications (`cmd`+`tab`) and switching windows (`cmd`+`'`).
- Self-reported workflow samples.

The pop-up as shown in Figure 3.1 allowed participants to create their recent workflows by selecting used applications and windows and also showed a timeline of previously used applications to make it easier for participants to remember their earlier activities.

All recorded data was stored locally on the participants device. Since one potential risk was that private data might get recorded, we mitigated this risk by showing participants the storage location of the data and provided instructions on how they can delete or censor data entries they do not want to share. Since some participants also used their work machine for private or non-work-related activities, the monitoring tool provided an option to pause recording interactions. Participants were also able to quit the tool at any time.

3.3 Participants

For this study we recruited participants through personal contact. Only participants that use working machines running macOS were eligible since the monitoring tool was developed for this platform.

Please provide one or two of your most recent workflows.
Response: 11/100

We consider a workflow as a sequence of steps and actions performed as part of a bigger task to achieve a certain goal. Workflows might be repeated multiple times for a given task. An example could be: (1) interacting with IDE editor, (2) web browser stackoverflow, (3) interacting with IDE editor, (4) running test case, (5) committing changes in Terminal/shell to repository.

To help you remember your most-recent workflows, here is an overview of the applications you used in the past 30 minutes:

Workflow 1

Add Step

Step 1
Remove

Application: Atom
Document: rope.dfy — ~/myd...

Step 2
Remove

Application: Firefox
Document: Dafny @ rise4fun f...

Step 3
Remove

Application: Firefox
Document:

Submit

Figure 3.1: Pilot study: Pop-up for indicating recent workflow.

Overall, we recruited 5 participants of which one was female and four were male (age range/mean/ \pm : 24-29/25.8/1.72). All participants were graduate students in computer science that had used their computer setup between two to seven years (range/mean/ \pm : 2-7/4.3/1.89). Participants indicated that they mostly work on their research, course projects or course assignments.

For each participant we collected between 3553 to 11091 events and recorded between 27.7 to 52.88 hours of interactions.

3.4 Results

We analysed the collected data to gain a deeper understanding of how participants interact with their computer and to find out if there are frequently recurring interaction patterns. In the following, we present the results grouped by the three questions we investigated.

3.4.1 How Many Applications and Resources Are Usually Open?

Table 3.1 summarises how many applications as well as windows and tabs were open during the period of the study for each participant. The median number of applications open at the same time was between 7 to 14 and of open windows and tabs was between 12 to 38. The recorded data also shows that participants used a variety of different applications and a large number of different resources, with 60% (± 14.5) of them being web pages, during the study.

	P1	P2	P3	P4	P5
Total Recorded Time [hours]	47.91	48.29	52.88	27.7	35.55
Total Recorded Events	5076	5564	11091	3553	5278
Total Applications	36	37	24	60	32
Total Resources	731	900	2225	624	908
Number of applications open at the same time					
Max.	15	14	16	20	12
Median	9	9	12	14	7
Number of windows and tabs open at the same time					
Max.	121	46	66	73	34
Median	22	22	38	33	12

Table 3.1: Pilot study: Total open applications, tabs, windows.

3.4.2 How Much Time Is Spent in Active Windows?

Figure 3.2 shows the distribution of time spent in active windows and tabs. Participants most often spent less than 10 seconds in active windows and tabs. Within that time frame about 14% of the usage durations less than one second long. These quick switches might have occurred when participants try to find the right window to switch to and accidentally switch to a wrong window for a short period of time. These results indicate that users spent only short times in active windows and tend to perform frequent switches.

During periods of activity, switches occur frequently. On average, participants switch 1.67 times (± 2.45) per minute. The maximum number of switches within that timeframe was 35. All participants have a similar average switch frequency with between 1.29 to 2.37 switches per minute aligning with related findings [14, 15].

3.4.3 Are There Frequently Recurring Interaction Patterns?

By applying pattern detection techniques based on the apriori principle [3] (see more details in Section 5.3), we were able to extract frequently recurring patterns.

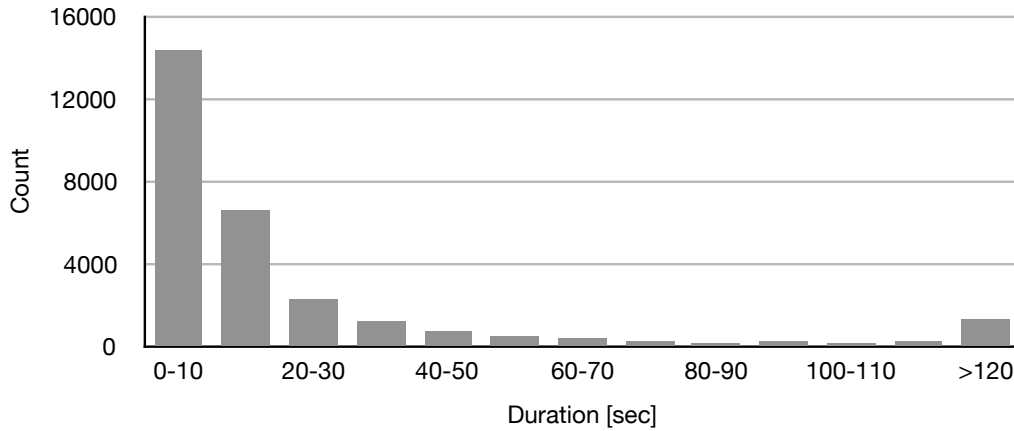


Figure 3.2: Pilot study: Usage duration of specific tabs and windows aggregated across all participants.

The core idea of this principle is that sequences are considered frequent if their sub sets are also frequent. Detected frequent sequences get concatenated with each other and extended to longer frequent sequences and get retained as long as they occur often enough. We determined the number of patterns that evolve during one hour of high activity. As parameters for the apriori algorithm we defined the minimum number of occurrences of an action must at least be 5 and a maximum switch distance between actions of 2 to be still considered as part of a pattern. These parameters ensure that only patterns are considered as frequent that have a relatively high number of occurrences and allowing some variances of patterns at the same time.

During their most active hours, between 87 to 417 different patterns emerged, with a median of 12 different patterns per hour. The median number steps of these patterns was 4.0. We only considered patterns that occurred at least 5 times, on average patterns had 8.9 occurrences in one hour.

We visualised the recorded interaction data and noticed recurring usage patterns. Figure 3.3 shows a screenshot of the tool used to visualise all occurrences of detected patterns.

The responses of participants in our follow-up survey about the detected patterns confirmed that most of these patterns were also perceived by participants as occurring frequently.

These findings also align with related work in terms of identifying temporal patterns in interaction data. [7, 16, 25].

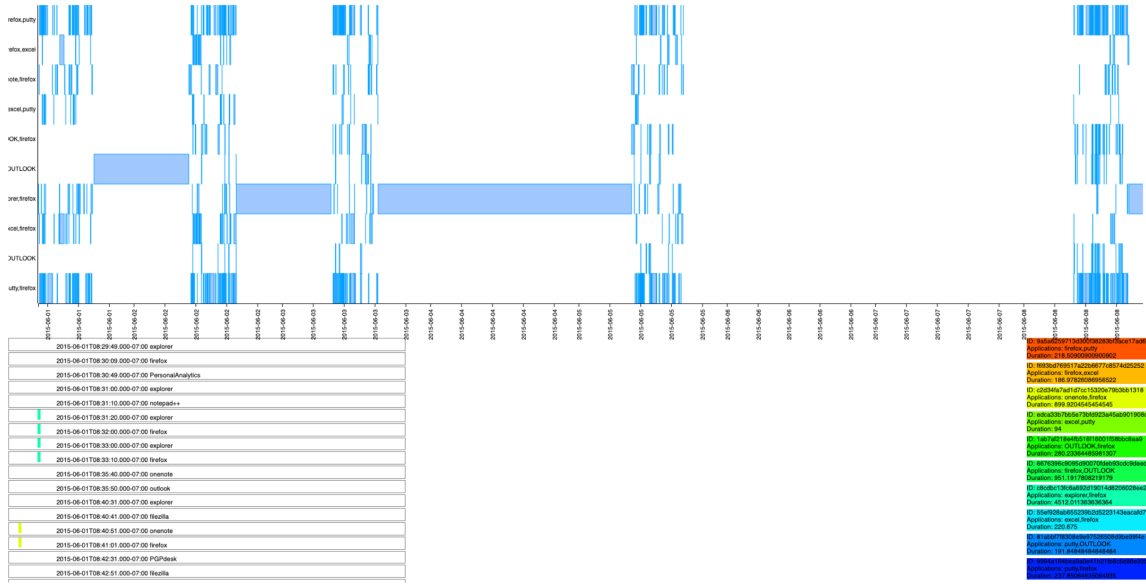


Figure 3.3: Screenshot of tool for visualising detected patterns from pilot data. The timeline on the top indicates the end and start of frequent patterns that have been detected. Frequent patterns consisting of the same applications and resources appear in the same row. Below is a log of every recorded event which was used for manual inspection. If events are part of a pattern then they get highlighted in a color that correspond to a frequent pattern and indicated in the legend on the right side.

3.5 Discussion

Our pilot study shows that users interact with a large number of different applications and resources and with many of them running at the same time. Also, windows are only active for a relatively short amount of time and switches between different windows occur repeatedly, and in recurring patterns, meaning that users tend to switch between the same sets of applications and resources over and over again.

These initial findings suggest that it could be beneficial to support users in these numerous switches and that these frequently occurring patterns could be leveraged to identify commonly appearing workflows or conceptual groups of a user.

Motivated by these findings, we conducted a study to gather ground truth data of how users define their conceptual groups while working on different projects. We describe our study method and results in the next section.

Chapter 4

Ground Truth Study

To get a better understanding of how users perceive their own conceptual groups, we performed a lab study. We designed this study to gather ground truth data that we could then use to develop and evaluate our approach. Additionally, we conducted interviews after the study asking participants about their perception of their own conceptual groups.

4.1 Study Procedure

We conducted a controlled lab study with 11 participants. In the study, participants were asked to work on three tasks and switch between them while we recorded their computer interactions and asked them to self-report on their conceptual groups.

We provided participants with a MacBook Air, an additional screen with a resolution of 1920×1200 , and an external mouse for the study. In an introduction sessions we explained the study and asked participants for consent. During the study, participants were asked to work on three different tasks and were free to use any programming language, application or web site. If an application was not already installed then they were free to install it.

A pre-installed background tool collected interaction data and prompted the participants approximately every 12 minutes (± 2 minutes) asking them to switch to the next project in a round-robin fashion. We chose a median of 12 minutes for these switches because previous work has shown that switches between working spheres occur about every 12 minutes [13].

Every 25 minutes a pop-up was presented asking participants to review and self-report their conceptual groups which we stated as being *applications/resources that are used together because of the tasks or projects that are worked on*. Having a time span of 25 minutes provided enough

time for participants to work on several projects for long enough that conceptual groups potentially changed without too many interruptions. The total duration of the study was 80 minutes during which three responses were collected. We made it clear to participants that they were not expected to finish all the tasks in the allocated time.

We concluded our study with a short interview to inquire how participants usually group their applications and resources, their understanding of conceptual groups and what tool support they would like to have to make their workflows and switches more efficient. We also demonstrated our prototype during the interview and asked for feedback.

4.1.1 Study Tasks

The three different tasks participants were working on during the study were:

1. coding a simplified blockchain implementation in any language by following a step-by-step tutorial,
2. documenting the software architecture of a provided raytracer implementation by creating a UML class diagram, and
3. planning a trip to a destination of the participant's choosing.

Each task assignment came with detailed descriptions and the order was varied between participants (see Appendix B). We chose to have three independent tasks which consisted of several different sub tasks and were related to software development related problems as well as general planning tasks. We consider projects as larger units of work consisting of multiple steps while tasks are considered as a single unit of work or a single step in a project. For example, the trip planning project consisted of multiple tasks, such as, filling a spreadsheet or writing a packing list. Having three projects allowed to present participants with a variety of different types of tasks but still gave them enough time to work on the individual projects to develop recurring interaction patterns. We chose these assignments for their similarity to projects that are encountered in a real-world setting. We ran a lab instead of an in situ because this allows for better comparability between participants and poses a lower risk of private information being recorded.

4.2 Study Support

To collect participants interaction data and their self-reports we used the monitoring tool as in our pilot study. We extended it to also keep track of URLs of web pages, file paths of used documents and task switch notifications.

For the self-reports, we developed a pop-up, as shown in Figure 4.1, that was prompted to participants and had a list of all applications and documents that were used and could be assigned to different conceptual groups. The pop-up allowed to add and delete groups and to assign applications and resources to groups by selecting the corresponding checkbox.

4.3 Participants

We recruited 11 participants (age range/mean/ \pm : 24-32/28.5/2.66, gender: 2 female, 9 male) through personal contacts for our lab study. All participants were graduate students in computer science and had experience in programming. Due to an initial error in the data collection tool, not all data of P1 was properly recorded. In the following, we will not consider the data collected from this participant for our analysis. The error that caused the tool to crash was fixed for all the participants that followed.

4.4 Results

First, we used the collected data and insights from interviews to investigate the characteristics and similarities of the way that different workers group their applications and artifacts (**RQ1**). Between 223 and 443 events (Mean/ \pm : 319.3/67.96) were recorded for each participant.

4.4.1 Group Size and Evolution

During the study, most participants created one conceptual group per task. P2 provided as the only participant four conceptual groups in the first response with three groups being for the three different projects and one being related to setting up the workspace and installing some missing applications. This group was merged with the conceptual group related to project 1 in the two remaining responses showing that conceptual groups can change quickly and get merged into other existing groups.

Groups consisted of 2 to 36 application and resource pairs, with an average of 10.5 (± 6.4). The sizes of groups increased over time. In the first responses, groups had an average size of 8.7 (± 4.9), in the second 10.5 (± 6.2) and in the last one 12.2 (± 7.6). Most of the resources and applications participants used got assigned to a conceptual group at some point. About 12.6% (± 4.7) of the resources used were completely ignored and could be considered as less relevant, for 0.02% (± 0.01) participants explicitly selected that they are not relevant.

Since switching was randomized with a median time of 12 minutes, four participants (P4, P5,

Please group the applications and resources (documents or web pages) according to the way you work with them, i.e. the applications/resources that are used together because of the tasks or projects that you work on, should be in the same group.

Some applications and resources might be part of multiple groups. Certain applications or resources might not be part of any group, for example in case of breaks or quick browsing, and can be marked as "Not relevant" or left unchecked. Think of these groups, for example, as the applications and resources you would like to open when resuming working on a project after rebooting your computer.

Add group

Application	Not relevant	Group 1	Group 2
related-work.tex	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
approach.tex	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
BibDesk	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Messages	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Mail	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Path Finder	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
▼ Terminal	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Terminal — -zsh — 170x46	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Firefox	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Affinity Designer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Preview	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Delete Group 1 Delete Group 2

Cancel Submit

Figure 4.1: Pop-up for self-reporting conceptual groups.

P6, P9) had only worked on two projects when their first review pop-up appeared and thus only provided two groups.

While our results show that most resources and applications are considered to be relevant at some point, we also discovered that conceptual groups undergo frequent changes. About 19% (± 8.8) of the resources are assigned to groups only once and did not get reassigned to a group when another review pop-up was prompted. 28.4% (± 8.9) appeared at least in conceptual groups

of two and 34.9% (± 7.6) in all three reviews. The resources that appear only once in conceptual groups tend to be used less often with a median of two usages while resources that are assigned to groups in all responses, like Google Search in a web browser application or viewing a specific PDF in a PDF viewer, have a median of five usages. This indicates that there are resources that are used more continuously and across various groups.

4.4.2 Group Composition

Most resources in conceptual groups were web pages with a mean of 46.7% (± 0.5) per group. Conceptual groups consisted to 34.5% (± 47.5) of documents, such as code or text files, to 9.36% (± 29.1) of command line actions, and to 8.22% (± 27.5) of file manager locations. This shows that web resources play a significant role when working on different tasks.

When asking for how participants would describe conceptual groups, all had generally the same understanding of the idea which was that conceptual groups are related to individual tasks or projects, describing them as “[...] *a set of windows or views, even tabs that fit together for a specific task or topic* [...]” (P8). While most groups are project or task-related, several participants (P1, P2, P3, P9) also noted that some of their daily conceptual groups outside of this study are related to communication, like mail applications or messengers, “[...] *can live across tasks* [...]” (P3) or are “*global*” (P3) and do not belong to a specific task.

4.4.3 Frequent Patterns

We visualised frequent pattern occurrences and noticed that while participants were working on individual tasks different sets of patterns are prominent. Figure 4.2 shows how patterns developed over time for participants P3, P6 and P8. Frequent patterns are different for each participant. An example of a pattern is a sequence of steps from File Authentication.swift in Xcode to File User.swift in Xcode to Web page google.com in Firefox.

Vertical lines indicate when participants received the notification to switch tasks and horizontal bars indicate the start and end of pattern occurrences. The tasks which participants were working on at any point in time are indicated in the top of the diagram with T1 being the blockchain implementation, T2 the travel planning and T3 the raytracer documentation. Different patterns that were active at a certain point in time are aligned along the y-axis. All occurrences of a specific pattern consisting of the same sequence of used applications and resources appear in the same row. For example, a pattern consisting of a sequence Google Search in Firefox to Terminal to StackOverflow in Firefox has multiple occurrences over the period of the study. All of the occurrences appear in

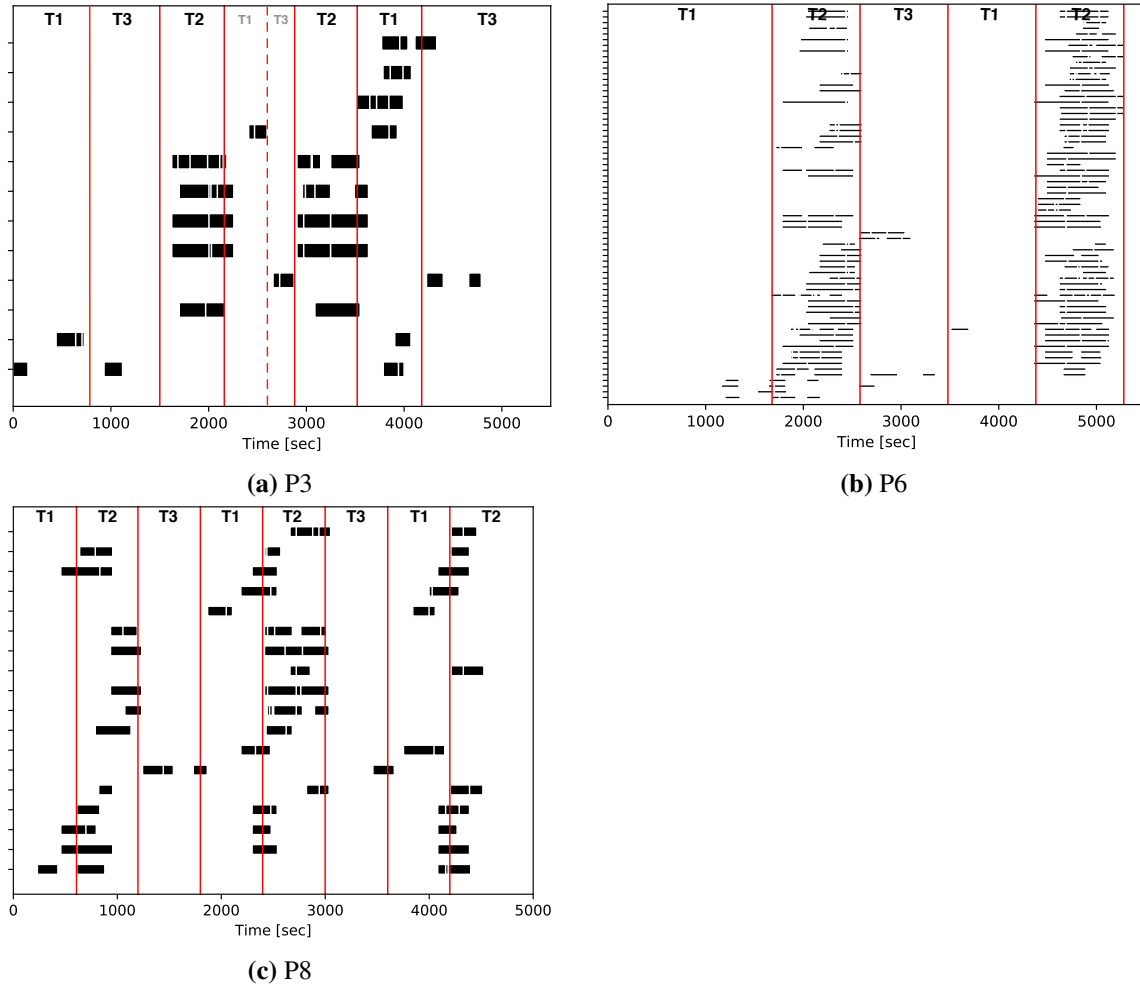


Figure 4.2: Frequent pattern occurrences (P3, P6, P8). Each row indicates the start and end of a specific pattern. Patterns consist of a specific set of applications and resources. Different patterns developed for different participants. Vertical lines mark project switches that participants worked on: T1 - blockchain, T2 - travel planning, T3 - raytracer.

the same row with marking the time when the sequence started and was finished. Patterns often overlap with other patterns that consist of very similar applications and resources. For example, the pattern consisting of a sequence Google Search in Firefox to Terminal to StackOverflow in Firefox often overlaps with a pattern Google Search in Firefox to Terminal to Authentication.swift in Xcode.

While working on a certain tasks the same patterns appear. For instance, for a specific parti-

ciapnt while working on documenting the ray tracer usually patterns that contain usages of Xcode and a tool for drawing diagrams occurred. Every time the participant was working on this project this pattern occurred. These sets of patterns could be thought of as footprint that is characteristic for the project and could be leveraged to identify conceptual groups. The visualisation also makes it apparent that P3 switched to project 3 while working on project 1 without receiving the notification to switch. The active patterns change although no notification has been received (see dotted vertical line). There is also a significant difference in the number of patterns created for each project. For all participants, most frequent patterns emerge when working on the travel planning since participants frequently switch between different websites for this task.

These results show that conceptual groups change over time, consist to a large amount of web pages, are usually related to a specific project and have a characteristic footprint consisting of frequently occurring patterns. This suggests that an approach to detect conceptual groups has to support frequent changes in conceptual groups and allow groups to consist of a wide variety of different applications and resources. In the following, we devise an approach that uses frequent pattern detection to detect conceptual groups. Our approach also tries to address that conceptual groups change over time and that some resources and applications are more or less relevant than others.

Chapter 5

Approach for Detecting Conceptual Groups

Our approach automatically detects conceptual groups from collected interaction event data for a single user. Conceptual groups are groups of applications and resources that are repeatedly used and relevant while working on a project. The main idea is that users have a mental concept for grouping their applications and resources that are used for certain tasks. They use the same applications and resources and perform the same sequences of actions repeatedly within a conceptual group for a task, such as, switching between two application, which allows to apply pattern mining on recorded interaction data to extract conceptual groups. Therefore, our approach searches for frequently recurring patterns which emerge when users switch between the same applications and resources. We visually analyzed patterns generated from the data collected from the ground truth study and noticed that the same patterns often appear temporally close to each other. This suggests that different conceptual groups could be distinguished by their characteristic pattern footprints. These footprints consists of sets of frequent patterns that appear together and overlap with each other in their occurrences.

We devised an approach to detect conceptual groups based on interaction event data. Figure 5.1 shows an overview of the steps involved, starting with event pre-processing, detecting patterns, clustering of patterns based on their overlapping occurrences, and assignment of applications and resources to detected groups.

5.1 Event Data Logging

Our approach takes as input computer interactions that are recorded as a sequence of non-overlapping events. Events can be recorded by a background monitoring tool, such as the one used in our previous studies. It is also possible to use a custom monitoring tool, e.g. to support other platforms, as long as for each event, the start and end timestamp, current application, resource used within that application, resource URL and the current window title is recorded. If the user does not perform any action, including keyboard inputs or scrolling, for more than 30 seconds, an idle event is recorded.

5.2 Pre-Processing Events

The recorded events are pre-processed so that only events that are recent enough with a start timestamp larger than γ are retained. As a starting point for our approach, we chose γ to be 2 weeks under the assumption that events from 2 weeks ago are not relevant for current conceptual groups anymore and can be ignored. Future research should look into providing users the option to choose gamma or identify which gamma works best.

The approach only considers resources that users spend a reasonable amount of time on. For our approach, we chose the time threshold to be 500ms and removed all shorter events, since we assume that events shorter than 500 milliseconds are accidental switches and can be ignored.

5.3 Frequent Pattern Analysis

Once we pre-processed the event data, we apply the apriori principle [3] to detect recurring patterns in the recorded activity event data. The main advantages of using the apriori principle is that it works efficiently with large amounts of data, is fully unsupervised, and finds all existing frequent patterns. The central rule of the apriori principle is that itemsets are considered frequent if their sub sets are also frequent. This allows the creation of new frequent itemsets by combining existing frequent itemsets.

In our case, if a short pattern, consisting of one or more events, is considered as frequent, meaning that it occurs more than ϕ times in the recorded events, then it gets extended by another frequent pattern. For the newly created pattern the number of occurrences are determined and the pattern will only be considered as frequent if it exceeds the threshold ϕ that needs to be configured. This procedure is repeated until all frequent patterns have been found and no new patterns get created.

After examining the data, we added four additional rules to our approach for the creation of frequent patterns which are also visualised in Figure 5.2 to ensure the correctness of created patterns which are denoted by A and B in the following:

1. Only two frequent patterns that are different can be merged into a new pattern. It is not allowed to merge the same patterns A and A to get the new pattern AA since that would mean that the user did not actually switch to a different resource.
2. When merging two frequent patterns A and B , the last event that is part of A must have occurred before the first event in B . This ensures that the generated patterns still consider the order of events. Also, an occurrence of A must be merged with the occurrence of B that appears closest to it.
3. For patterns A and B to be merged into one pattern, there must have been less than c events between the last event of A and the first event of B . c can be configured. When generating these frequent patterns, we want to be resistant to noise. Users might not always have the exact same behaviour when switching between applications, they might have a different switch order or skip certain applications from time to time.
4. Patterns with fewer than k events (for example $k = 1$) are removed since they do not provide any meaningful information.

5.4 Clustering

In a next step, our approach clusters frequent patterns that frequently occur together. Clustering is necessary since the footprint to identify different tasks is a combination of multiple patterns, so it is necessary to determine which patterns belong together and represent a characteristic footprint for a specific task.

For the clustering, we first calculate the overlap ratio is calculated pairwise between all patterns and for each pair the maximum overlap ratio is retained. A ratio of 0.0 implies that the two patterns do not overlap while 1.0 implies that at least one of the two patterns completely overlaps with the other. These ratios are used as input for the clustering using DBSCAN [10], which is a density-based clustering algorithm and will return an arbitrary number of clusters that we consider as the conceptual groups. DBSCAN requires two parameters ϵ and $minPts$ that have to be defined and are used as a similarity threshold so that patterns are considered as closely related and a minimum number of related patterns to form a conceptual group. The major benefit of using DBSCAN for

clustering is that it does not require our approach to define the number of expected clusters, is robust to outliers, and finds arbitrarily shaped clusters.

5.5 Application and Resource Assignment

Finally, we assign previously used applications and resources to the generated conceptual groups. Events might not be part of a frequent pattern, but they might still be used in context of a specific task and therefore should be part of the corresponding conceptual group. This is, for example, the case for events that are not frequent enough but always used in very close time proximity to frequent patterns.

For this, we iterate through all pre-processed events and check if they occur within close time proximity (e.g. $\pm 5s$) of a frequent pattern occurrence and add them to the conceptual group. For all events that do not appear in close time proximity to a frequent pattern the resource titles will be compared with resources that have been assigned to conceptual groups. Only resource titles of the same applications are compared and they are considered as part of the same group if their similarity score calculated using TF-IDF exceeds threshold s .

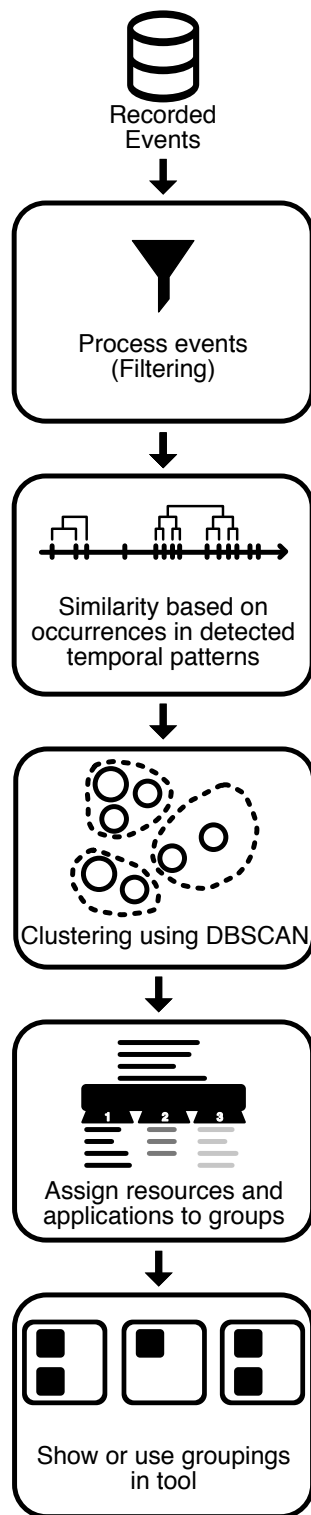


Figure 5.1: Overview of the steps involved to detect conceptual groups.

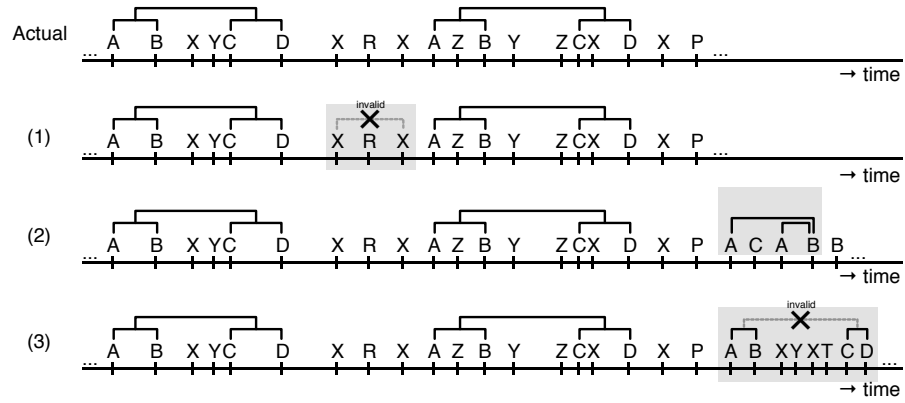


Figure 5.2: Constraints for creating frequent patterns. Different patterns are indicated on a timeline based on their occurrence and denoted by different characters (A, B, X, \dots) Actual: patterns detected in the event data. (1) only different patterns, (2) the closest occurrences of patterns and (3) patterns with less than $c = 3$ events inbetween can be merged.

Chapter 6

Evaluation

To evaluate the accuracy of our approach to detect conceptual groups, we used the data collected in our lab study. We examine the similarity between the conceptual groups self-reported by our participants with the groups our approach generated automatically. In particular, we examined how many applications and resources were assigned correctly and incorrectly to automatically generated groups, are missing, and for which it is unclear whether they are correct or wrong based on the collected data.

For our evaluation, we experimented with multiple sets of parameters for α , ϕ , ϵ , c , $minPts$, k and s . Specifically, we adjusted parameters to have our approach generate a reasonable number of groups (2 to 4 groups per response). The best result was achieved by choosing the following parameters:

- $\alpha = 1$ second (minimum event length to not get filtered out when pre-processing events)
- $\phi = \max\{\log(0.05 \times \text{total events}), 2\}$ (minimum number of occurrences). We chose this value to account for scaling of the number of patterns that should get detected based on the number of previous actions. If there only have been a few events, then patterns repeat less frequently. To detect patterns when users just started working, ϕ has to be relatively small. If more resources are used over time, ϕ should increase to prevent the detection of too many, potentially insignificant, patterns. However, to detect new patterns that emerge at a later point in time, ϕ needs to be bounded instead of steadily increasing. This behaviour is represented by the logarithm function.
- $c = 3$ (maximum number of switches in between two frequent patterns for them to still be

allowed to get combined). This number needs to be relatively small to prevent combinatorial explosion, but large enough to ensure noise resistance when combining patterns.

- $k = 1$ (minimum pattern length)
- $\epsilon = 1 - 0.15$ (DBSCAN). Patterns must at least have an overlap of 15% to be considered as part of the same group.
- $minPts = 1$ (DBSCAN). Minimum number of related patterns to form a group.
- Patterns within ± 5 seconds before or after a frequent pattern are added to the corresponding group. (Resource assignment)
- $s = 0.91$ (TF-IDF similarity threshold). This threshold is set quite high in order to ensure that only resources with similar titles are clustered together.

6.1 Accuracy

In order to calculate the accuracy of our approach, a label is assigned to each resource and application used which represents the generated cluster they are part of. This enables us to compare how similar the label assignments from the ground truth data and the generated conceptual groups are. Since our approach generates an arbitrary number of clusters, first it needs to be determined which of the generated clusters represent the clusters indicated in the ground truth. For this correlation, for each generated group the group of the ground truth that has the most applications and resources in common is selected as counterpart. A group can only be selected once, so if our approach generated more groups than have been indicated in the ground truth, some groups might not have a counterpart.

We analyse the accuracy of our approach as follows: “Correct” matches refer to the percentage of applications and resources that were assigned to the same conceptual groups as indicated in the participant responses. For this we calculate how many of the applications and resources are labeled according to the provided ground truth. “Wrong” matches data points that were assigned to the incorrect group and do not match the labeling of the collected ground truth while “Missing” are resources that are indicated in the ground truth but are not present in the generated groups. Occasionally, participants added resources to groups only in one response but did not reassign them to the group in the following responses, or only assigned them at a later point in time. This could either mean that these resources are no longer or not yet relevant or that participants forgot to select

them. Nevertheless, if a resource has been part of a group previously or at a later point in time then that might be an indicator that it is relevant to that conceptual group. These cases are indicated as “Likely Correct”. Overall, 41.8% (± 14.5) of applications and resources are assigned “Correct”ly, 25.6% (± 15.3) are assigned to “Wrong” groups, 13.5% (± 13.1) are “Missing”, and 17.4% (± 10.7) are “Likely Correct”. The overall accuracy improved over time with more collected data. After the first response, 33.9% (± 15.1) resources were correctly assigned, 42.3% (± 13.3) were correct after the second and 48.1% (± 13.1) were correct after the third response. Over time more patterns emerged which makes different footprints more discernable and leads to better accuracy.

Results for the measured accuracies for all participants are shown in Figure 6.1. The difference in the number of generated groups and the number of groups indicated in the ground truth is also provided for each participant. For example, if our approach generated 2 groups instead of 3 groups that a participant indicated, then the difference is indicated as -1.

Of the wrongly assigned data, on average 14.9% (± 11.3) are resources that participants did not assign to any groups. Participants might have considered these as less relevant or might have forgotten to select them. It would be interesting to analyse in a future study whether having these applications and resources in groups is unfavourable for users or whether these should actually be part of the groups. The remaining incorrectly assigned data points occurred due to our approach correlating them to the wrong patterns.

If we consider that the applications and resources classified as “Likely Correct” are potentially assigned correctly, then we would have an overall accuracy of 45.3% (± 18.6) and an accuracy of 51.3% (± 14) after the third response.

Notably, for some participants (P7, P9) the percentage of “Missing” fluctuates significantly. The main reason for this is that over time ϕ is changing. It can happen that patterns disappear over time resulting in some resources being added or removed from groups.

There are several reasons for the accuracy of the generated groups not being higher. Firstly, the number of groups that are generated sometimes does not match the number of groups indicated in the ground truth. Figure 6.1 shows the distribution of the difference in the number of groups between the ground truth on right side of the charts. Generally, the number of generated groups tended to be below the number of actual groups. The numbers increased over time, getting closer to the actual number of indicated groups. The difference in numbers of groups is also the main factor for incorrectly assigned resources and applications. 87.6% (± 29.9) of the wrongly assigned applications and resources belonged to a group that was missing or should have been part of another group. This was especially problematic in the beginning when only a small set of events have been recorded it is hard to discern different contexts. This happens because some applications, such

as web browsers and search engines, are used across all groups which means the characteristic footprints for the groups might be quite similar. It is possible that over a longer period of time our approach will more accurately determine the number of groups the user perceives since patterns will manifest and re-occur more often over time. To further verify this, it would be necessary to conduct a longer study.

Secondly, resources and applications that have been used less frequently are more likely to be treated as noise which occasionally results in data points not getting assigned to any group while users still consider them as relevant. Applications and resources that were considered “Missing” have been only used with a median of 4 times before each response while the ones that have been assigned to a group have been used with a median of 18 times.

6.2 Comparison to Related Work

In a further step we compared our approach to the automatic task-cluster generation approach based on document switching and revisitation [2] and clustering based on semantic similarity of window titles, similar to the one used in SWISH [22] (**RQ3**). We chose to compare against these two approaches since they are most similar and also automatically detect groups. We modified the approach used in SWISH and used DBSCAN instead of KMeans for clustering to allow a variable number of groups. We examined multiple different sets of parameters for ϵ and *minPts*, best results were achieved with $\epsilon = 0.85$ and *minPt* = 4. Using the data collected in the ground truth study, the accuracies for the approaches after each response are shown in Figure 6.2.

Our approach has an overall accuracy of 41.8% (± 14.5), higher than the other two approaches which have an overall accuracy of 24.4% (± 9.4) and 25.4% (± 13.5) respectively. Therefore, our approach exceeds the accuracy by 41.6% and 39.2% respectively. Additionally, for each of the responses, our approach has a higher accuracy and increases its accuracy over time while accuracy remains relatively constant and did not improve over time for the other approaches.

Overall, the results suggest that frequent patterns can serve as indicators for conceptual groups resulting in a higher accuracy than previous related approaches. After the third response we measured an average accuracy of 48.1% (± 13.1) with accuracies up to 72.3% for certain participants. However, using frequent patterns it is not always possible to detect new conceptual groups right away as they emerge, instead a certain number of actions need to be performed so that the characteristic footprints can be detected.

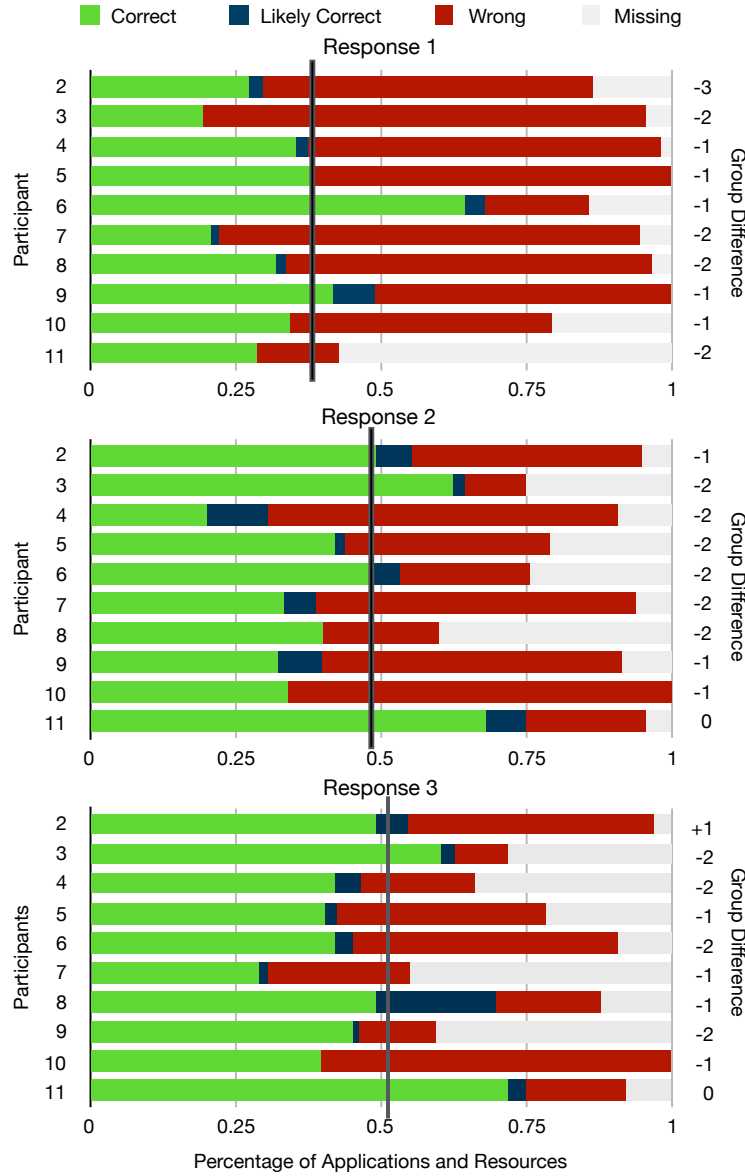


Figure 6.1: Accuracy of our approach for each response of each participant. “Correct”: data assigned to same groups as in ground truth, “Wrong”: incorrectly assigned data, “Missing”: data indicated in the ground truth but missing in the generated groups, “Likely Correct”: data that has been part of the group earlier or later but not at the time of the response. Vertical black lines mark the average correctness. Number on the left refer to the participant. Numbers on the right indicate the difference in number of generated groups vs. number of groups indicated by participants.

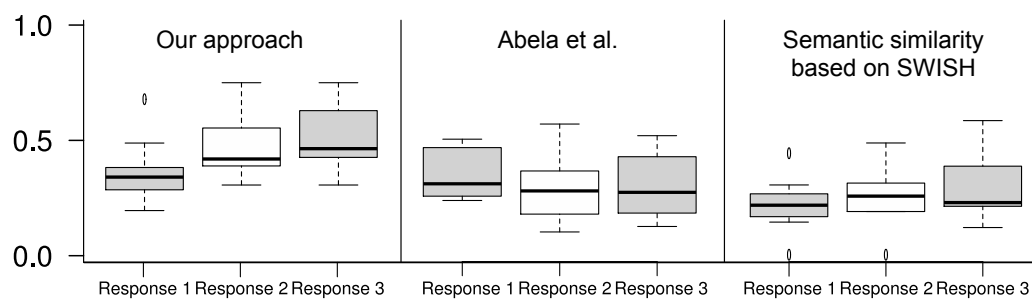


Figure 6.2: Boxplots of accuracies of related approaches for each response.

Chapter 7

Threats

As with any study there are some internal and external threats to its validity.

7.1 External Validity

Since our pilot study was executed as an in-situ study some variables are out of our control. For example, some participants stopped the tool for several hours during the study. Additionally, prompting the popup for reporting on their workflows at least twice an hour might have added distractions and impacted the way participants would normally work.

The tasks participants were working on in our ground truth study might not be generalizable to tasks that are performed in actual work environments. We tried to choose a variety of different tasks which are mostly related to common activities of software developers to mitigate this risk. In a real work environment, workers are repeatedly externally interrupted or interrupt themselves or get distracted. These different types of interruptions were not simulated in our lab study but might have an impact on the conceptual groups.

While our approach is not only applicable for software development related tasks, we only had participants that were working in a computer science research context and are developing software on a daily basis. Workers in other fields might have different working styles which might make our conceptual group approach more or less applicable. Also, all participants were grad student, so results might be different for software developers working in industry.

All of our participants were very experienced computer users working the majority of their work day with computers. Novice computer users might not have a different perception of their conceptual groups.

7.2 Internal Validity

We asked participants for their workflows during our pilot study, however, because participants struggled with identifying their workflows, we re-designed our approach to detecting conceptual groups. Since we did not further use the self-reports participants provided in the pilot study the impact on our developed approach is minimal.

During the ground truth study, we asked participants to indicate their own conceptual groups. We therefore provided participants an explanation of conceptual groups in the beginning. However, this concept is quite general and some users indicated that they were unsure about their exact groups. Therefore some of the collected results might have some inaccuracies due to uncertainty or the results might be slightly biased based on our explanation in the beginning.

Additionally, the parameters we used for our approach might change if tools record data and create conceptual groups over a longer period of time to yield results with high accuracy. The parameter configuration used for our evaluation potentially overfits with the collected data and might have been optimized for the collected data and should be investigated further in future work. We also tried different parameter values for approaches from previous work we compared our approach against and chose the ones that resulted in the highest accuracy. In the future, more data should be collected and separate data sets for training and testing should be used to get more accurate values for these parameters.

The study was executed on a provided machine running macOS. Some of the participants (P4, P7) were not familiar with the operating system and its usage which might have impacted their usual workflow. We tried to mitigate this risk by providing a short tutorial about the usage. Additionally, we allowed participants to use any application or programming language they wanted so that they could work in an environment they are familiar with.

7.3 Construct Validity

It is hard to measure how well conceptual groups can describe how users think of the way applications and resources should be organized for the projects they are working on. It might sometimes be hard for participants to decide whether a specific resource is actually relevant to a specific project or not. Therefore, there is a threat to the construct validity. We tried to mitigate this threat by providing a definition of conceptual groups and validating the detected patterns with participants afterwards as well.

Chapter 8

Discussion

We have shown that our approach can detect conceptual groups with an accuracy higher than previous work. In this chapter we discuss how the algorithm for detecting conceptual groups might be used in approaches to support users in their work. We thereby focus on a prototype that we developed, design recommendations that we identified and further future work in the area.

8.1 Supporting Professionals with Conceptual Groups

Conceptual groups can be used as basis for tools that support professional workflows. In the following, we present and discuss the prototype that we built to support professionals as well as feedback we received from users that were presented with screenshots of the prototype.

8.1.1 Tool Prototype

To help users in switching between different applications and projects, we developed a prototype tool that uses the conceptual group approach to detect the groups. It then uses these groups to show users their conceptual groups and make switching within and between them easier. Figure 8.1 depicts a screenshot of our prototype.

Users could use the tool as an addition to the application switcher (`cmd` + `tab`), for example if they wish to switch between tasks instead of just applications. For this the tool also allows to open all the applications that are part of a specific group. It can also be used to declutter the current workspace by allowing to close all applications and documents of a specific conceptual group.

The tool is implemented in Swift and runs on macOS. Users can open the tool using the key combination `cmd` + `fn` + `tab`.



Figure 8.1: Prototype of our tool for displaying and interacting with conceptual groups.

Conceptual groups are shown as boxes containing applications and related resources. Users can switch between applications and resources within these groups as well as between groups quickly using key combinations. Groups are updated periodically, e.g., every two minutes in the background, and applications are ordered based on their most recent usage. Since the generated conceptual groups might not entirely match the users perception, it is possible for users to delete or add groups, resources, and applications, as well as drag and drop applications and resources between groups.

8.2 Prototype Feedback

Start with a sentence saying (in case it's true) that "overall participants perceived the prototype predominantly positive, e.g. "This would...". At the same time, participants also mentioned several ideas for extending the current work, specifically with respect to the interactivity of the approach, the abstraction and visualization of it, and the use of it to enhance other navigation tools.

We showcased our tool to ask for feedback by showing screenshots and overall participants perceived the prototype predominantly positive, e.g. *"This would actually be really helpful!"* (P4). At the same time, participants also mentioned several ideas for extending the current work, specifically with respect to the interactivity of the approach, the abstraction and visualisation of it, and the use of it to enhance other navigation tools.

Interacting with Conceptual Groups. Several participants commented on the need for interacting with the conceptual groups and expressed interest in having indicated the ability to name groups (P3) and for the tool *"to break [resources] down more fine-grained"* (P2). In our demo, most of the conceptual groups had applications that had only very few different resources and some of the visited tabs were filtered out as they were considered as not relevant. Some participants were concerned whether the conceptual groups could be detected accurately and appreciated that it is also possible to manually adapt the generated groups (P1, P2).

Participants also expressed interest in having the ability to save conceptual groups and to later restore them as well as to close all the applications and documents that are part of a conceptual group that has been completed or suspended (P2, P3, P4, P7, P8, P9, P10). This would not only allow users to get back to working on a project quickly but also to share their conceptual groups with others.

Conceptual Groups Summaries and Visualisations. Different ways of representing the conceptual groups were suggested, such as displaying the current state of the actual windows instead of just showing application icons and the resource titles or to use an alignment similar to Exposé on macOS (P1). Another concern that was mentioned is that conceptual groups might grow very large over time, hence it is important to retain only the applications and resources that are actually relevant to that group. Ideally, tools should summarise resource usages and or provide search functionality.

Conceptual groups could also indicate *"[...] daily goals [...]"* (P4) and help users to maintain focus on their current tasks (P3). It might also be useful to show activity statistics or even add some gamification to help users stay focused.

Using Conceptual Groups to Enhance Current Navigation Tools. Navigational tools, such as

the application switcher could implement the approach and only show applications or windows of the current conceptual group or learn commonly repeating patterns to order the windows accordingly so that “[...] *alt + tab* would switch to the right window even if it wasn’t the most recent window [...]” (P3). It might also be useful to “[...] *have alt + tab between conceptual groups.*” (P6). In the case of Exposé on macOS “[...] *it could blur some of the applications that are not related.*” (P5) or group windows by their conceptual groups (P2). Additionally, window tiling managers could be combined with conceptual groups and “[...] *learn the layout and associate that with the context or the conceptual group.*” (P3).

8.2.1 Accuracy Improvements

Our approach currently considers frequent patterns for determining conceptual groups. Since it often takes some time for frequent patterns to evolve, our approach is not always able to detect or update conceptual groups in real time. By considering additional properties the accuracy might increase. For instance, instead of just analysing the window title, semantically analysing the whole content of a window to group windows together might yield more accurate results as window titles often consist only of very few words.

Further improvements could be achieved by using eye tracking to determine which windows are actually looked at and used by users. Currently, our data collection approach assumes that windows that are in focus are actively used. However, often users open multiple windows on the same or several screens and arrange them next to each other. While working they might look at some window, eg. to read through documentation or source code without putting that window into focus. So although that window is actively used our approach would not recognize the usage.

Certain actions such as copy, cut and paste could be strong indicators that the user is still working within the same conceptual group. It could be likely that text gets usually copied and pasted between windows that should belong to the same conceptual group. Taking these input actions into account might further increase accuracy.

8.3 Future Work

This section discusses future work related to improving the accuracy of our approach, different applications and possible field studies to gather more general observations about conceptual groups.

8.3.1 Field Study

We intend to further test our approach in a field study to confirm that our approach also works outside the lab environment. Instead of providing tasks to users, in this study participants will resume working on their own tasks. The tool used in our lab study would still run in the background, collect data, and ask for conceptual groups from time to time. The study will run over several days providing insight into how accurate our approach works in a more real-world setting, how conceptual groups develop and change over time and how many conceptual groups workers typically have. Additionally, it would be interesting to compare different workers in their perceptions of conceptual groups. We will try to run this study with mainly software developers working in industry as opposed to only graduate students since the working styles in academia and industry are potentially quite different.

8.3.2 Applications

Section 8.1 provides a wide range of different applications of our approach and ideas for tools that could integrate the approach. In the future, we are planning to develop prototypes for the ideas and perform user studies for some of these tools to determine how helpful and usable they are.

Chapter 9

Conclusion

Professionals use a lot of applications and resources during their work day for the projects, tasks and activities they work on and have to switch between them a lot. This thesis presents an approach to automatically generate conceptual groups from recorded desktop interaction data. To obtain these groups our approach analyses recurring patterns that emerge when switching between different applications and windows.

We conducted a lab study with 11 participants to collect ground truth data for conceptual groups and to evaluate our approach. Results show that frequent patterns can be used to discern and detect conceptual groups with an accuracy of 48%, however, to achieve an even higher accuracy it might be necessary to consider additional factors. The accuracy of our approach also exceeds the accuracy of previous similar approaches by up to 50%.

Based on the approach, we developed a prototype that detects conceptual groups and supports professionals in their navigation between these groups. In an interview with participants about conceptual groups and our prototype, we gained more insight into how participants perceive their conceptual groups in their daily work life and received many suggestions for potential applications. We believe that these future applications of conceptual groups could help users in organizing their workspace and could make workflows and task switches more efficient.

Bibliography

- [1] C. Abela, C. Staff, and S. Handschuh. Online activity graph for document importance and association. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 191–194. ACM, 2011. → pages 1, 5
- [2] C. Abela, C. Staff, and S. Handschuh. Automatic task-cluster generation based on document switching and revisitation. In *UMAP Workshops*, 2015. → page 29
- [3] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM, 1993. → pages 10, 21
- [4] L. Bannon, A. Cypher, S. Greenspan, and M. L. Monty. Evaluation and analysis of users’ activity organization. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 54–57. ACM, 1983. → pages 1, 4
- [5] D. Barreau and B. A. Nardi. Finding and reminding: file organization from the desktop. *ACM SigChi Bulletin*, 27(3):39–43, 1995. → page 4
- [6] M. S. Bernstein, J. Shrager, and T. Winograd. Taskposé: exploring fluid boundaries in an associative window visualization. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 231–234. ACM, 2008. → page 6
- [7] O. Brdiczka, N. M. Su, and J. B. Begole. Temporal task footprinting: identifying routine tasks by their temporal patterns. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 281–284. ACM, 2010. → pages 5, 11
- [8] S. Costache, J. Gaugaz, E. Ioannou, and C. Niederée. Detecting contexts on the desktop using bayesian networks. 2010. → page 5
- [9] A. N. Dragunov, T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, and J. L. Herlocker. Tasktracer: a desktop environment to support multi-tasking knowledge workers. In *Proceedings of the 10th international conference on Intelligent user interfaces*, pages 75–82. ACM, 2005. → page 5

- [10] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996. → page 22
- [11] V. M. González and G. Mark. Constant, constant, multi-tasking craziness: managing multiple working spheres. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 113–120. ACM, 2004. → pages 1, 4
- [12] M. S. Magnusson. Discovering hidden time patterns in behavior: T-patterns and their detection. *Behavior Research Methods, Instruments, & Computers*, 32(1):93–110, 2000. → page 5
- [13] G. Mark, V. M. Gonzalez, and J. Harris. No task left behind?: examining the nature of fragmented work. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–330. ACM, 2005. → page 13
- [14] G. Mark, S. T. Iqbal, M. Czerwinski, and P. Johns. Bored mondays and focused afternoons: the rhythm of attention and online activity in the workplace. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3025–3034. ACM, 2014. → pages 1, 10
- [15] A. N. Meyer, T. Fritz, G. C. Murphy, and T. Zimmermann. Software developers’ perceptions of productivity. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 19–29. ACM, 2014. → pages 1, 10
- [16] A. N. Meyer, L. E. Barton, G. C. Murphy, T. Zimmermann, and T. Fritz. The work life of developers: Activities, switches and perceived productivity. *IEEE Transactions on Software Engineering*, 43(12):1178–1193, 2017. → page 11
- [17] H. T. Mirza, L. Chen, G. Chen, I. Hussain, and X. He. Switch detector: an activity spotting system for desktop. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2285–2288. ACM, 2011. → page 5
- [18] Y. Miyata and D. A. Norman. Psychological issues in support of multiple activities. *User centered system design: New perspectives on human-computer interaction*, pages 265–284, 1986. → pages 1, 4
- [19] R. Morteo, V. M. González, J. Favela, and G. Mark. Sphere juggler: fast context retrieval in support of working spheres. In *Proceedings of the Fifth Mexican International Conference in Computer Science, 2004. ENC 2004.*, pages 361–367. IEEE, 2004. → pages 4, 5
- [20] R. Nair, S. Voids, and E. D. Mynatt. Frequency-based detection of task switches. In *Proceedings of the 19th British HCI Group Annual Conference*, volume 2, pages 94–99, 2005. → page 5

- [21] S. Nijssen and J. N. Kok. The gaston tool for frequent subgraph mining. *Electronic Notes in Theoretical Computer Science*, 127(1):77–87, 2005. → page 5
- [22] N. Oliver, G. Smith, C. Thakkar, and A. C. Surendran. Swish: semantic analysis of window titles and switching history. In *Proceedings of the 11th international conference on Intelligent user interfaces*, pages 194–201. ACM, 2006. → pages 1, 5, 29
- [23] N. Oliver, M. Czerwinski, G. Smith, and K. Roomp. Relalttab: assisting users in switching windows. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 385–388. ACM, 2008. → page 6
- [24] T. Rattenbury and J. Canny. Caad: an automatic task support system. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 687–696. ACM, 2007. → pages 1, 5
- [25] J. Shen, E. Fitzhenry, and T. G. Dietterich. Discovering frequent work procedures from resource connections. In *Proceedings of the 14th international conference on Intelligent user interfaces*, pages 277–286. ACM, 2009. → pages 5, 11
- [26] J. Shen, J. Irvine, X. Bao, M. Goodman, S. Kolibaba, A. Tran, F. Carl, B. Kirschner, S. Stumpf, and T. G. Dietterich. Detecting and correcting user activity switches: algorithms and interfaces. In *Proceedings of the 14th international conference on Intelligent user interfaces*, pages 117–126. ACM, 2009. → page 5
- [27] G. Smith, P. Baudisch, G. Robertson, M. Czerwinski, B. Meyers, D. Robbins, and D. Andrews. Groupbar: The taskbar evolved. In *Proceedings of OZCHI*, volume 3, page 10, 2003. → pages 1, 5
- [28] C. Tashman and W. K. Edwards. Windowscape: Lessons learned from a task-centric window manager. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 19(1):8, 2012. → page 6

Appendix A

Pilot Study Details

A.1 Recruiting Participants

After obtaining approval for conducting the study from the UBC Ethic's Board, we first tested our study on two graduate students for 3 days. After this test period we made minor adjustments to the study pop-up and added a timeline of recently used applications and resources. To recruit participants for our pilot we used mailing lists and personal contacts at UBC. We sent the following invitation via email:

Subject: Invitation to participate in study on detecting user workflows

Hi,

Have you ever been curious about how often you perform the same steps over and over again while working on your computer? Wouldn't it be nice to have tool support to make these workflows more efficient?

Under the supervision of Professor Reid Holmes, I, Anna Scholtz, am conducting a study to gain insight into workflows software developers perform while working on their computer. I would be delighted to invite you to participate in this study.

This will be an in situ study, conducted over a period of two weeks. After a short instruction session, we will ask you to install a tool that will monitor all computer interactions in the background. After that you can resume working while running the tool in the background. Approximately every 20 minutes, a popup will appear asking you about your two most-recent workflows. A brief interview session, in which we will

ask you questions related to your indicated workflows, will conclude the study.

To be eligible, you must be working with macOS and use your machine for more than one hour per day.

If you are interested in participating, or if you have any questions, please reply to this email.

Participant	Age	Gender	Which best describes your primary work area?	Which of the following best describes your role?	Which of the following statements describes your current position best?	For how long have you been using your current computer/desktop setup?
P1	25	Female	Research	Individual Contributor	Junior	3.5
P2	29	Male	Research	Individual Contributor	Other	3
P3	24	Male	Research	Individual Contributor	Junior	7
P4	25	Male	Research	Individual Contributor	Junior	2
P5	26	Male	Research	Individual Contributor	Junior	6

Table A.1: Pilot study participant metadata.

A.2 Setup and Introduction Session

If interested participants were eligible, then we scheduled an introduction session in which we explained the purpose of the study and installed the monitoring tool. We followed the following script for this session:

Welcome to the study. The study is about detecting user workflows. We consider a workflow as a sequence of steps and actions performed as part of a bigger task to achieve a certain goal. Workflows might be repeated multiple times for a given task. An example could be: (1) interacting with IDE editor, (2) web browser StackOverflow, (3) interacting with IDE editor, (4) running test case, (5) committing changes in Terminal/shell to repository.

For the study, we have to install a program that collects the following data:

- *active applications, background applications: name, title, window position*
- *aggregated input (no clear text input): number of scroll events, cursor distance, keyboard presses within 10 seconds*
 - *we are **not** recording each input event to prevent sniffing of passwords or personal information.*
- *Idle times*
- *data entered into popup*

All recorded information is stored in a database that you can navigate to using our tool. You can click on the program icon in the status bar, then click on "Preferences" and then click on "Open data folder" which will open the directory containing the database. The database file can be edited and viewed using common database browser applications like "DB Browser for SQLite". You can remove or modify data entries that you do not want to share at any time.

During the study you resume your usual work. If you don't want to track your actions for some time, either quit application or disable trackers by checking the option in the menubar of our monitoring application. Remember to start the application again and enable study mode when you want to resume recording.

Approximately every 20min a popup will appear.

[Show screenshot of popup]

You are asked you to indicate one to three of your most-recent workflows. For each workflow indicate the steps by adding more boxes and specifying applications and documents you used during that step. You can also choose to not answer the popup by leaving all fields empty and clicking "Submit".

The study will end either after 10 working days or once you submitted 100 popup responses. For each response we will compensate you with 50cents. In total, you can get up to \$50 of compensation. After the study you will be asked to complete a final survey.

Do you have any questions?

If you have any concerns or questions during the study, then please let me know.

[Setup notifications]

[Show SQL tool for modifying data entries]

After obtaining the participants consent using the consent form shown in Figure A.2 and answering remaining questions, we started the study.



The University of British Columbia
Department of Computer Science

201-2366 Main Mall
Vancouver BC
Canada V6T 1Z4

Consent Form

User Workflow Detection

Principal Investigator

[Redacted]

Co-Investigators

[Redacted]

Purpose

The overall purpose of this research is to support software development professionals by providing valuable insights into their work patterns and developing tools to support recurring workflows. To accomplish this objective, we are investigating methods that automatically detect recurring workflows in user activity data which is collected while software development professional are working at their computer.

Study Procedure

This study is an in situ study, performed over a period two weeks. The study is composed of three parts:

- (a) a 20-minute long introduction session in which we will explain the study to you, ask for your consent and install a tool which will monitor all computer interactions in the background,
 - (b) a period of up to 10 workdays in which you are asked to continue working as usual and will be prompted a popup every 20 minutes to provide self-reports about your current workflows while the monitoring tool is running in the background, and
 - (c) a 15-minute follow-up survey, in which we will ask you about demographics and more generally about the workflows and workflow switches you performed.
- Each self-report will take approximately 2 minutes to answer. During an average working day, we expect that answering all popups will take less than 30 minutes in total.

Note: we consider a workflow as a sequence of steps and actions performed as part of a bigger task to achieve a certain goal. Workflows might be repeated multiple times for a given task.

If you agree to participate, we will install the monitoring tool on your computer. Note that only software development professionals working with macOS and using their machine for more than four hours per day are eligible to participate. The installed tool will collect the following data:

- Name, title and window position of active and background applications
- Aggregated input data, such as number of scroll events, cursor movements, number of keyboard presses and number of clicks. We are not recording individual key strokes but only the number of keys pressed every 10 seconds. This prevents recording of passwords or other private information.
- Idle times
- Self-reported workflow samples

The data will be stored locally on your device. At the end of the study our tool will provide you instructions how this data should be sent to us. During the introductions session we will show you how recorded data that you do not want to share can be removed or censored.

Known Risks

One risk is that potentially private or confidential data will be recorded. We are mitigating this risk by showing the location of where collected data is stored. You are allowed to remove or censor data entries that you do not want to share at any time. Additionally, the monitoring program does not capture each key stroke which prevents recording of passwords or other private information. Also, you can terminate the study at any point in time without providing any reason. Otherwise, the risks involved in this study are minimal and are those commonly associated with the use of computers, such as potential eye or wrist strain.

Potential Benefits

We do not expect there to be any potential benefits from participating in this study. However, you may find it interesting to reflect on your activity patterns while working on your computer. Direct benefits can arise if our findings lead to tools that can automatically identify current workflows and assist in providing information, such as related applications, documents or websites relevant for the current workflow.

Compensation

We use a micro-payment approach and will compensate you with 50 cents for every self-report you submit, with a maximum compensation of CDN \$50 per person (100 submitted self-reports). The study will end either after you submitted 100 self-reports or after 10 workdays. Note that you can withdraw from the study at any time without a reason and will receive a prorated reimbursement.

Data, Storage & Confidentiality

All data (survey responses and monitoring data) will be saved on password-protected and encrypted devices of the researchers directly involved or in locked university filing

cabinets in rooms of the University of British Columbia. All data will be anonymized before and will not contain any identifying information.

You will be identified by numbers or pseudonyms in any internal or academic research publication or presentation. If we choose to use some of your comments, they will be attributed to a participant number or a pseudonym. At no point in time will your employer have access to the identifying information. The data will be stored for five years, after which it will be permanently deleted.

Use of the Data

Data collected will be used for analysis and may also be used for class project presentations and other research presentations. This project forms the basis of a thesis research project and may be submitted as a research publication in the future.

Contact for information about the study

If you have any questions about or desire further information with respect to the study,

[REDACTED]

Who can you contact if you have complaints or concerns about the study?

If you have any concerns or complaints about your rights as a research participant and/or your experiences while participating in this study, contact the Research Participant Complaint Line in the UBC Office of Research Ethics at 604-822-8598 or if long distance e-mail RSIL@ors.ubc.ca or call toll free 1-877-822-8598.

Consent

Your participation in this study is entirely voluntary. You are free to withdraw your participation at any point during the study, without needing to provide any reason. You can disable the monitoring software yourself at any time during the study. Any information you contribute up to your withdrawal will be retained and used in this study, unless you request otherwise.

With your signature on this form, you confirm the following statements:

- I am at least 18 years old.
- I had enough time to make the decision to participate and I agree to the participation.

Name of Participant

Signature of Participant

Location and Date

Figure A.1: Pilot study consent form.

A.3 Final Survey

After finishing the study, participants were asked to complete a survey as shown in Figure A.3. For Q8 we determined frequently occurring patterns using frequent pattern analysis and updated the survey for each participant based on their frequent workflows.

Q0. Thank you again for participating in our study!

Your data will help us a lot in our research on automating workflows and we hope you enjoyed the participation. Finally, we would like to ask you a few more questions and would appreciate your feedback.

Q1. How old are you?

24

Q2.
What is your gender?

Male

Q3.
Which best describes your primary work area?

- ☒ Research
- ☐ Development
- ☐ Project Management
- ☐ Testing
- ☐ Other Engineer
- ☐ Other Non-Engineer

Q4.
Which of the following best describes your role?

- ☒ Individual Contributor
- ☐ Lead
- ☐ Manager
- ☐ Other

Q5. Which of the following statements describes your current position best?

- ☐ Senior
- ☒ Junior
- ☐ Other

Q6. For how long have you been using your current computer/desktop setup?

7 years

Q7. What is your definition of *workflow*?

Sequence of actions I do while completing a task

Q8.

For the following workflows, could you please rate how often they occurred during your work?

	Never	Rarely	Occasionally	Frequently	Very Frequently	Always
iTerm2 → Preview → Google Chrome	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Discord → iTerm2 → Google Chrome	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Mail → Preview → Google Chrome	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Xcode → Safari → Terminal	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Google Chrome ("Messenger") → Google Chrome ("YouTube") → Google Chrome ("Messenger")	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
iTerm2 ("1. fg") → Google Chrome ("Messenger")	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

Q9.

Can you think of any way/approach that could support you performing some/multiple/all workflows and if so how? For example, would a tool that automatically groups windows/applications by workflow in the task bar or on your screen be of use to you?

While it would be nice to have some low level tasks automatized, most of the action in my workflows requires to look and read the screen, therefore there isn't much that could be doable to automaize that.

Q10. Are there any parts or whole workflows that you would like to be automated (regardless of how realistic it may be) and if so, which ones and why?

Not really. Most of my workflow involves one or two tools, and I use shortcuts to quickly jump through them

Figure A.2: Pilot study final survey.

Appendix B

Ground Truth Study Details

B.1 Recruiting Participants

After obtaining approval for conducting the study from the UBC Ethic’s Board, we first tested our study on two graduate students for 3 days. After this test period we made minor adjustments to the study pop-up and the frequency of how often it task switches and conceptual group reviews should occur. To recruit participants for our study we used mailing lists and personal contacts at UBC. We sent the following invitation via email:

Invitation to participate in the study “Detecting Conceptual Groups”

Hi,

Have you ever been curious about how often you perform the same steps over and over again while working at your computer? Or what applications and documents you usually use for your different high-level tasks? Wouldn’t it be nice to have tool support to make switching between tasks more efficient?

Under the supervision of Professor Thomas Fritz, I, Anna Scholtz, am conducting a study to gain insight into workflows software developers perform and their conceptual groups while working on their computer. I would be delighted to invite you to participate in this study.

This will be a lab study, conducted over a period of approximately 2 hours. After a short instruction session, we will ask you to install a tool that will monitor all computer interactions in the background. You can either use your own computer if it runs

macOS or we can provide you with a MacBook Air. You are asked to work on different tasks while running the tool in the background. These tasks range from implementing a simple software system to planning a vacation trip. Approximately every 20 minutes, a popup will appear asking you to review your conceptual groups that have been detected. A brief survey, in which we will ask you about your demographics and questions related to your indicated workflows, will conclude the study.

For your participation in this study, we will compensate you with a 20\$ Amazon gift card.

If you are interested in participating, please reply to this email. We will then follow up with obtaining your consent and scheduling a study session.

If you have any questions, feel free to contact me at any time.

B.2 Setup and Introduction Session

If interested participants were eligible, then we scheduled an introduction session in which we explained the purpose of the study and installed the monitoring tool and asked for their consent using the consent form shown in Figure B.2. We followed the following script for this session:

Welcome to the study. The study is about detecting conceptual groups and grouping applications and used resources that belong to one conceptual group. We consider a conceptual groups as groups of applications and resources, such as files or websites. So conceptual groups contain all applications and resources necessary to achieve a certain goal or to complete tasks that have a shared goal. These goals could for example be developing a software system or planning a vacation and could be composed of other sub-tasks.

For the study we provide you with a MacBook Air which runs a program in the background that collects the following data:

- *Active applications: name, title, window position*
- *Used resources: file paths, URLs*
- *Key combinations: cmd + tab, switching window*
- *Idle times*
- *Generated groups*

- *Data entered into popups*

During the study, you will work on 3 different tasks which will be described in detail in the provided files:

- 1. Task 1 is about writing a simplified blockchain implementation in a language of your choosing. We provide a step-by-step tutorial in Python which you can follow.*
- 2. Task 2 is about creating a UML class diagram for a raytracer written in Python. The source code can be found in the provided files.*
- 3. Task 3 is about planning a trip to a destination of your choosing. You will fill out a provided spreadsheet template asking about sights, food and accommodation for your trip. Additionally, please create a packing list of things you would take on the trip.*

You are not expected to finish the tasks in the allocated time.

A popup will ask you to switch to the next task from time to time. If you reached task 3 then continue with task 1 again.

Approximately every 20 minutes a popup will appear asking you to review your conceptual groups. You can open the conceptual group view by clicking on the notification. The overview allows you to create new groups, delete groups, add and delete applications and resources from groups and let's you drag and drop applications between groups.

[Show screenshot of tool]

The study will end after 90 minutes and will be compensated with a 20\$ Amazon gift card.

All recorded information is stored in a database that you can navigate to using our tool. You can click on the program icon in the status bar, then click on

"Preferences" and then click on "Open data folder" which will open the directory containing the database. The database file can be edited and viewed using common database browser applications like "DB Browser for SQLite". You can remove or modify data entries that you do not want to share at any time.

Do you have any questions?

If you have any concerns or questions during the study, then please let me know.



The University of British Columbia
Department of Computer Science

201-2366 Main Mall
Vancouver BC
Canada V6T 1Z4

Consent Form

Detection of Conceptual Groups

Principal Investigator

[Redacted]

Co-Investigators

[Redacted]

Purpose

The overall purpose of this research is to support software development professionals by providing valuable insights into their work patterns and developing tools to support recurring workflows. To accomplish this objective, we are investigating methods that automatically detect recurring workflows and conceptual groups in user activity data which is collected while software developers are working at their computer.

Study Procedure

This study is a lab study, performed over a period of approximately 2 hours. The study is composed of three parts:

- (a) a 5-minute long introduction session in which we will explain the study to you, ask for your consent and install a tool which will show groupings of applications and resources that are conceptually related,
- (b) a period of up to 90 minutes in which you are asked to work on three different provided tasks. The tasks are composed of a software implementation, a software planning and a trip planning task. A popup is prompted approximately every 20 minutes asking to provide self-reports about your conceptual groups. You will also get asked to change to different tasks via a popup.

(c) A 15-minute follow-up survey, in which we will ask you about demographics and more generally about your conceptual groups.

Each self-report will take less than 2 minutes to answer.

If you agree to participate, we will either install the tool on your computer or provide you with a MacBook Air that has the tool already installed. Note that you can only use your own device if it is running macOS. The installed tool will collect the following data:

- Name, title and window position of active and background applications
- Name, path and titles of used files and visited URLs
- Groups the installed tool generates
- Tool settings
- Idle times and task switches
- Self-reports

At the end of the session we will show you how recorded data that you do not want to share can be removed or censored.

Known Risks

One risk is that potentially private or confidential data will be recorded. We are mitigating this risk by showing the location of where collected data is stored. You are allowed to remove or censor data entries that you do not want to share at any time. Additionally, the monitoring program does not capture any keystrokes which prevents recording of passwords or other private information. Also, you can terminate the study at any point in time without providing any reason.

Potential Benefits

We do not expect there to be any potential benefits from participating in this study. However, you may find it interesting to reflect on your activity patterns while working on your computer. The tool might have some direct benefit since it assists participants in providing information, such as related applications, documents or websites relevant for the current workflow.

Compensation

We will compensate you with up to CDN \$20 in form of a gift certificate for your participation. Note that you can withdraw from the study at any time without a reason and will receive a prorated reimbursement.

Data, Storage & Confidentiality

All data (survey responses and monitoring data) will be saved on password-protected and encrypted devices of the researchers directly involved or in locked university filing cabinets in rooms of the University of British Columbia. All data will be anonymised before and will not contain any identifying information.

You will be identified by numbers or pseudonyms in any internal or academic research publication or presentation. If we choose to use some of your comments, they will be attributed to a participant number or a pseudonym. At no point in time will your employer have access to the identifying information. The data will be stored for five years, after which it will be permanently deleted.

Use of the Data

Data collected will be used for analysis and may also be used for class project presentations and other research presentations. This project forms the basis of a thesis research project and may be submitted as a research publication in the future.

Contact for information about the study

If you have any questions about or desire further information with respect to the study,



Who can you contact if you have complaints or concerns about the study?

If you have any concerns or complaints about your rights as a research participant and/or your experiences while participating in this study, contact the Research Participant Complaint Line in the UBC Office of Research Ethics at 604-822-8598 or if long distance e-mail RSIL@ors.ubc.ca or call toll free 1-877-822-8598.

Consent

Your participation in this study is entirely voluntary. You are free to withdraw your participation at any point during the study, without needing to provide any reason. You can disable the monitoring software yourself at any time during the study. Any information you contribute up to your withdrawal will be retained and used in this study, unless you request otherwise.

With your signature on this form, you confirm the following statements:

- I am at least 18 years old.
- I had enough time to make the decision to participate and I agree to the participation.

Name of Participant

Signature of Participant

Location and Date

Figure B.1: Ground truth study consent form.

B.3 Study Execution

We provided participants with 3 folders that contained descriptions of the tasks they were asked to work on and additional files. The ordering of the tasks was random for each participant.

B.3.1 Task 1 - Let's Go Travel

The description for the travel planning task is depicted in Figure B.3.1.

Let's Go Travel

This task is about planning a trip to a destination of your choosing. Please do a little research about activities, food, flights as well as accommodations for your trip and fill out the `travel-planner` spreadsheet. Try to stay within a budget of \$4000.

Please create a packing list for the things you want to take on the trip.

Figure B.2: Travel planning task description.

We provided a spreadsheet template shown in Figure B.3 in which participants could enter information.

TRANSPORTATION							
Carrier	Flight #	Date	From	Departure Time	To	Arrival Time	Total Cost
Trenz Airlines	1623	Oct 25, 2018	Lorem International	7:56 AM	Dolor Airport	9:15 AM	\$1220.00
							\$1220.00

LODGING							
Accommodations	Date	Address	Days	Website/Link	Rating	How to get there?	Total Cost
Lorem Hotel	Oct 25, 2018	1234 First Street, Anytown, State ZIP	2	https://www.lipsum.com/	3.7/5	Taxi	\$800.00
							\$800.00

Figure B.3: Spreadsheet template for the travel planning task.

B.3.2 Task 2 - Step by Step Blockchain

The description and step-by-step tutorial for the task asking for a simplified blockchain implementation is shown in Figure B.3.2.

Step by Step Blockchain

This task is about writing a simplified blockchain implementation. The following steps will guide you through the process. You can choose any programming language for your own implementation. The implementation will be slightly different from the reference implementation and the tutorial. After each step you will be asked to use the version control system [git](#) to record your changes.

A blockchain is an immutable, sequential chain of records called *blocks*. They can contain transactions, files or any kind of data. Our implementation will be based on transactions. The blocks are chained together using hashes. Blockchains can have a wide variety of applications such as for crypto currencies, smart contracts or other financial services.

In the following we'll first create a basic implementation of a blockchain and then add a small API that can be used to interact with the implemented blockchain.

Step 0 - Setup

Please create a new git repository in the task folder `/task-1`. You can use any git client of your choosing. The following instructions will be based on using git in the command line:

Open the command line, switch to the current working directory and

run `git init`.

Create a new project folder inside `/task-1` for your blockchain implementation and setup your project in your preferred programming environment.

Once everything is set up, create your initial commit:

You can first review changes: `git status`

It often makes sense to exclude certain file (eg. binaries) that should not be tracked. For this, create a new file `.gitignore` and add the paths of the files to be excluded.

To add and commit changed files run `git add` and `git commit` and provide a descriptive commit message.

Step 1 - Blocks

Blockchains consist of blocks that look like this:

```
block = {
  'index': 1, # Index of this block
  'timestamp': 1506057125.900785, # Creation time
  # It might make sense to create a separate data structure for transactions
  'transactions': [ # List of transactions
    {
      'sender': "8527147fe1f5426f9dd545de4b27ee00",
      'recipient': "a77f5cdfa2934df3954a5c7c7da5df1f
```

```

",
    'amount': 5,
}
],
'proof': 324984774000, # Proof (more on that later)
'previous_hash': "2cf24dba5fb0a30e26e83b2ac5b9e29e1b16
1e5c1fa7425e73043362938b9824" # Hash of the previous block
}

```

Let's first create a data structure to represent blocks. Note that the blocks themselves will be immutable which means that they cannot be changed otherwise `previous_hash` of subsequent blocks would be incorrect.

For this you might also need to implement a data structure representing transactions.

Commit your changes.

Step 2 - Blockchain

Next, create a blockchain data structure that will manage all the blocks. An exemplary blueprint for this class could look like:

```

class Blockchain(object):
    def __init__(self):
        self.chain = []

```

```

        self.current_transactions = []

    def new_block(self):
        # Creates a new Block and adds it to the chain
        pass

    def new_transaction(self):
        # Adds a new transaction to the transactions list
        pass

    @staticmethod
    def hash(block):
        # Hashes a Block
        pass

    @property
    def last_block(self):
        # Returns the last Block in the chain
        pass

```

Commit your changes.

Step 3 - Blockchain Initialization and Creating Blocks

When the blockchain is first instantiated a *genesis block* - a block with no predecessors - needs to be created. This block needs to have a *proof*

which is the result of *mining* and will be described a little later.

Before creating the genesis block, the blockchain could use a general method for creating a new block and a method to calculate the SHA-256 hash of a block:

```
class Blockchain(object):
    def new_block(self, proof, previous_hash=None):
        """
        Create a new Block in the Blockchain
        :param proof: <int> The proof given by the Proof of
        Work algorithm
        :param previous_hash: (Optional) <str> Hash of pre
        vious Block
        :return: <dict> New Block
        """

        # Create a new instance of block data structure
        block = {
            'index': len(self.chain) + 1,
            'timestamp': time(),
            'transactions': self.current_transactions,
            'proof': proof,
            'previous_hash': previous_hash or self.hash(se
            lf.chain[-1]),
        }

        # Reset the current list of transactions
```



```

        self.current_transactions = []

        self.chain.append(block)

        return block

    def hash(block):
        """
        Creates a SHA-256 hash of a Block
        :param block: <dict> Block
        :return: <str>
        """

        # We must make sure that the Dictionary is Ordered
        # , or we'll have inconsistent hashes
        block_string = json.dumps(block, sort_keys=True).encode()

        return hashlib.sha256(block_string).hexdigest()

    # [...]

```

The genesis block can now be created as follows when instantiating a new `Blockchain`:

```

class Blockchain(object):
    def __init__(self):
        self.current_transactions = []

```

```

self.chain = []

# Create the genesis block
self.new_block(previous_hash=1, proof=100)

# [...]

```

Commit your changes.

Step 4 - Transactions

Next, blockchains need a way to add transactions to blocks:

```

class Blockchain(object):
    # [...]

    def new_transaction(self, sender, recipient, amount):
        """
        Creates a new transaction to go into the next mined Block

        :param sender: <str> Address of the Sender
        :param recipient: <str> Address of the Recipient
        :param amount: <int> Amount
        :return: <int> The index of the Block that will hold this transaction
        """

```

```

        self.current_transactions.append({
            'sender': sender,
            'recipient': recipient,
            'amount': amount,
        })

        return self.last_block['index'] + 1

    # Helper to return the last mined block of the chain
    @property
    def last_block(self):
        return self.chain[-1]

```

Note that the function `new_transaction` returns the index of the next block to be mined that will contain the added transaction.

Commit your changes.

Next we will dive deeper into how blocks are created, forged and mined.

Step 5 - Proof of Work

The *proof of work* is how new blocks are *mined*. The proof of work is a number that should be computationally difficult to find but easy to verify. The rule we will implement in the following is: **Find a number p that when hashed with the previous block's solution a hash with 4 leading 0s is produced.**

```

class Blockchain(object):
    # [...]

    def proof_of_work(self, last_proof):
        """
        Simple Proof of Work Algorithm:
        - Find a number p' such that hash(pp') contains le
        ading 4 zeroes, where p is the previous p'
        - p is the previous proof, and p' is the new proof
        :param last_proof: <int>
        :return: <int>
        """

        # At this point we brute-force until we get a vali
        d solution
        p = 0
        while self.valid_proof(last_proof, p) is False:
            p += 1

        return p

    @staticmethod
    def valid_proof(last_proof, proof):
        """
        Validates the Proof: Does hash(last_proof, proof)
        contain 4 leading zeroes?
        :param last_proof: <int> Previous Proof

```

```

:param proof: <int> Current Proof
:return: <bool> True if correct, False if not.
"""

guess = f'{last_proof}{proof}'.encode()
guess_hash = hashlib.sha256(guess).hexdigest()
return guess_hash[:4] == "0000"

```

Miners get a small reward (eg. 1 coin) that will be added as a new transaction into the chain. More about this will be explained later.

Commit your changes.

Step 6 - Blockchain API

Next we'll create a small API to use the previously created blockchain over the web using HTTP requests.

You can use any HTTP framework you'd like for this part in the following example code *Flask* is used.

```

class Blockchain(object):
    # [...]

    # Instantiate our Node
    app = Flask(__name__)

```

```

# Generate a globally unique address for this node
node_identifier = str(uuid4()).replace('-', '')

# Instantiate the Blockchain
blockchain = Blockchain()

# The API endpoints

@app.route('/mine', methods=['GET'])
def mine():
    return "We'll mine a new Block"

@app.route('/transactions/new', methods=['POST'])
def new_transaction():
    return "We'll add a new transaction"

@app.route('/chain', methods=['GET'])
def full_chain():
    response = {
        'chain': blockchain.chain,
        'length': len(blockchain.chain),
    }
    return jsonify(response), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)

```

This creates the following endpoints:

- `GET /mine` : mines a new block
- `POST /transactions/new` : creates a new transaction
- `GET /chain` : returns the blockchain as json

Here, we run the server on port `5000` . Feel free to use any server and port configuration you prefer.

Commit your changes.

Step 7 - Transaction Endpoint

We already have the logic implemented for creating a new transaction.

So we just need to connect it to our new `/transactions/new` endpoint:

```
# [...]  
@app.route('/transactions/new', methods=['POST'])  
def new_transaction():  
    values = request.get_json()  
  
    # Check that the required fields are in the POST'ed data  
    required = ['sender', 'recipient', 'amount']  
    if not all(k in values for k in required):  
        return 'Missing values', 400  
  
    # Create a new Transaction
```

```

        index = blockchain.new_transaction(values['sender'], v
        alues['recipient'], values['amount'])

        response = {'message': f'Transaction will be added to
        Block {index}'}
        return jsonify(response), 201
    # [...]

```

Commit your changes.

Step 8 - Mining Endpoint

The mining endpoint does the following 3 things:

- Calculate the proof of work
- Add 1 coin as reward for the miner as transaction to the chain
- Forge the new block by adding it to the chain

```

# [...]

@app.route('/mine', methods=['GET'])
def mine():
    # We run the proof of work algorithm to get the next p
    roof...

    last_block = blockchain.last_block
    last_proof = last_block['proof']
    proof = blockchain.proof_of_work(last_proof)

```



```

    # We must receive a reward for finding the proof.
    # The sender is "0" to signify that this node has mine
    d a new coin.
    blockchain.new_transaction(
        sender="0",
        recipient=node_identifier,
        amount=1,
    )

    # Forge the new Block by adding it to the chain
    previous_hash = blockchain.hash(last_block)
    block = blockchain.new_block(proof, previous_hash)

    response = {
        'message': "New Block Forged",
        'index': block['index'],
        'transactions': block['transactions'],
        'proof': block['proof'],
        'previous_hash': block['previous_hash'],
    }
    return jsonify(response), 200

# [...]

```

Commit your changes.

Step 9 - Interacting with the

Blockchain

Now we can manually test and interact with the created blockchain.

First start the server that is executing the blockchain API:

```
$ python blockchain.py

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Next we can run some **GET** and **POST** requests. One way to run these requests is either by using **curl** on the command line or by using a GUI tool, like **postman**.

We can execute the following requests:

- **GET http://localhost:5000/mine**
- **POST http://localhost:5000/transactions/new** with a body containing our transaction structure:

```
$ curl -X POST -H "Content-Type: application/json" -d '{
  "sender": "d4ee26eee15148ee92c6cd394edd974e",
  "recipient": "someone-other-address",
  "amount": 5
}' "http://localhost:5000/transactions/new"
```

- mine a few blocks to get a more impressive chain

- `GET http://localhost:5000/chain`

Step 10 - File-sharing Blockchain

Any data can be used in a blockchain and transformed to transactions metadata. In the case of file sharing, blockchains would remove the need of having a central store of the files. In this step you are asked to extend your blockchain implementation so that transactions can be used for file sharing and can contain files.

For now all the participants in the chain would receive a copy of that metadata and therefore the files. You can look into approaches to make this more secure and ensure that only certain participants can access the files.

This step is not part of the reference implementation.

Please commit your changes.

Next Steps

We now have a basic blockchain implementation. Right now our implementation is not decentralized. If you still have time, you can implement the *Consensus Algorithm* by following the guide starting at Step 4 [here](#).

This tutorial is based on <https://hackernoon.com/learn-blockchains-by-building-one-117428612f46>

Figure B.4: Blockchain implementation task description.

B.3.3 Task 3 - Raytracer Documentation

Figure B.3.3 shows the task description for the raytracer documentation task. We also downloaded the code files for the raytracer from <https://github.com/martinchristen/pyRT> and stored them in the directory we provided to participants.

Raytracer Documentation

This task is about creating a [UML class diagram](#) for a raytracer written in Python. Raytracers are used in computer graphics to render a scene and generate images. Raytracers generate the images by tracing the path of light as pixels on the image plane. This technique creates very real looking images and is used in the production of animation movies. For this task you *do not* have to understand the math behind raytracers (but you are welcome to try following it while creating the UML class diagram).

You can find the source code for the raytracer in [/pyRT/pyrt](#) . You can also look at some rendered examples in [/pyRT/examples](#) .

UML class diagrams are often used in software documentations. You can use any tool you'd like for creating the diagram. Some tools that work well are:

- [yEd](#)
- the free online editor [draw.io](#)
- any graphics editor such as [Inkscape](#) or [Open Office Draw](#)

You can find the UML class diagram specification online. A short cheat sheet can be found here: <https://docs.microsoft.com/en-gb/visualstudio/modeling/uml-class-diagrams-reference?view=vs-2015>

Please create a simplified UML class diagram which contains all

relevant classes and relevant methods of these classes. Please export the class diagram to PDF, PNG or JPEG and save it in the `/task-2` directory.

The source code is from the repository
<https://github.com/martinchristen/pyRT>.

Figure B.5: Raytracer documentation task description.

B.4 Final Interview

During the final interview, we asked participants the following questions:

- 1. How do you usually switch between applications and different windows?*
- 2. Do you currently group applications and resources in some way?*
- 3. What is your definition of conceptual group?*
- 4. Do you experience any problems when grouping them?*
- 5. Imagine the groups could be detected with perfect accuracy, do you think they would help you in your workflow? How and when would they be particularly helpful?*
- 6. Would you prefer a certain way of displaying the groups?*
- 7. After showing prototype: Would you prefer a different way of displaying the groups? Would you use a similar or improved version of a tool for this? Helpful when switching between tasks or applications of one task?*

All interviews were recorded and later transcribed.