Master Thesis

August 17, 2023

TaskSnap

Semi-Automatic Task Context Capturing & Task Resumption Support for Software Developers

Remy Egloff

of Wettingen, Switzerland (17-705-823)

supervised by

Prof. Dr. Thomas Fritz
Dr. André Meyer
Roy Rutishauser

University of Zurich^UZH

HASEL

Master Thesis

TaskSnap

Semi-Automatic Task Context Capturing & Task Resumption Support for Software Developers

Remy Egloff

University of Zurich^UZH

HASEL

Master Thesis

Author: Remy Egloff, remy.egloff@uzh.ch

Project period: 17.02.2023 - 17.08.2023

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

Acknowledgements

I want to thank all members of the Human Aspects of Software Engineering Lab (HASEL) at the University of Zurich for their support, great advice, and fruitful discussions during the course of this thesis. Especially, I would like to thank Dr. André Meyer, Roy Rutishauser, and Prof. Dr. Thomas Fritz for their excellent supervision. By providing continuous feedback, they helped to shape the approach presented in this thesis. In addition, many thanks to the participants of the pilot study. Testing the assumptions and design decisions made while developing an approach is vital to creating a valuable product. Therefore, learning about participants' work routines and getting feedback regarding the developed approach was insightful.

Abstract

The workday of software developers is highly fragmented, as many developers work on multiple tasks per day and frequently switch between them, for example, to help co-workers or when being stuck. Frequent task switches introduce time overhead, as the user must always capture and restore the tasks' working context (e.g., applications, documents, folders) and mental context (e.g., task knowledge, goals, intentions). Previous work focused on supporting users in restoring their working context, for instance, by keeping track of task-related documents or web pages. However, these approaches frequently do not help users to re-establish their mental task context and operate fully automated, which can lead to restoring task-unrelated artifacts. In addition, existing approaches are generally not targeting software developers by not displaying source code related information. To overcome these shortcomings, we propose an approach that facilitates task context capturing and resumption for software developers and data scientists by allowing the user to semi-automatically create a snapshot of a task's associated working and mental context at any time. Later, when resuming a task, all information stored in a snapshot can be restored. A two-week pilot study with six participants showed that the approach fitted well into existing workflows, supported users in capturing their working and mental context, and saved them time when resuming a task. Users mainly created snapshots when having enough time, for example, at the end of the workday to reflect on the day and detach from work. Creating snapshots during instant task switches was less common, as participants did not encounter these situations frequently during the pilot study, likely because they were part-time developers. In addition, participants curated snapshots by providing thorough descriptions of their intent on how a task should be continued and frequently restored them within 24 hours.

Zusammenfassung

Die Arbeitstage von Software-Entwickler:innen sind stark fragmentiert, da viele Entwickler:innen täglich an mehreren Aufgaben arbeiten und häufig zwischen diesen wechseln, zum Beispiel um Mitarbeitenden zu helfen oder wenn sie selbst nicht mehr weiterkommen. Das häufige Wechseln von Aufgaben benötigt Zeit, da der:die Nutzer:in den Arbeitskontext (z.B. Applikationen, Dokumente, Ordner) sowie den mentalen Kontext (z.B. Aufgabenwissen, Ziele, Absichten) festhalten und wiederherstellen muss. Frühere Arbeiten fokussierten darauf, dem:der Nutzer:in beim Wiederherstellen des Arbeitskontexts zu helfen. Zum Beispiel, indem aufgabenbezogene Dokumente oder Webseiten erfasst werden. Jedoch unterstützen diese Ansätze den:die Nutzer:in häufig nicht dabei, den mentalen Kontext wiederherzustellen und die Ansätze funktionieren vollautomatisiert, was zum Öffnen von aufgabenfremden Artefakten führen kann. Ausserdem richten sich bestehende Ansätze im Allgemeinen nicht an Software-Entwickler:innen, da keine auf den Quellcode bezogenen Informationen dargestellt werden. Um diese Schwachstellen zu beseitigen, präsentieren wir einen Ansatz, der Software-Entwickler:innen und Data Scientists dabei hilft, den Aufgabenkontext festzuhalten und wiederherzustellen, indem der:die Nutzer:in die Möglichkeit hat, halbautomatisiert und zu jeder Zeit eine Momentaufnahme des Arbeitskontexts und des mentalen Kontexts zu erstellen. Später, bei der Wiederaufnahme einer Aufgabe, können alle Informationen, die in der Momentaufnahme gespeichert sind, wiederhergestellt werden. Eine zweiwöchige Pilotstudie mit sechs Teilnehmenden konnte zeigen, dass der Ansatz gut in existierende Arbeitsabläufe passte, dass Nutzer:innen im Festhalten des Arbeits- und des mentalen Kontexts unterstützt wurden und sie Zeit beim Wiederaufnehmen einer Aufgabe sparen konnten. Nutzer:innen erstellten Momentaufnahmen vor allem, wenn sie genug Zeit hatten, beispielsweise am Ende des Arbeitstages, um über den Tag zu reflektieren und sich von der Arbeit zu lösen. Das Erstellen von Momentaufnahmen während schnellen Aufgabenwechseln war weniger üblich, da die Teilnehmenden diesen Situationen während der Pilotstudie nicht häufig begegneten, wahrscheinlich weil sie Teilzeit-Entwickler:innen waren. Des Weiteren kuratierten Teilnehmende Momentaufnahmen, indem sie ausführliche Beschreibungen über ihre Absicht, wie eine Aufgabe fortgeführt werden soll, verfassten. Momentaufnahmen wurden häufig innerhalb von 24 Stunden wiederhergestellt.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software developers usually work on multiple tasks during a day and frequently switch between them [MBM$^+$17], for example, to help co-workers with their questions, switch to another task when being stuck, or to start dealing with a new issue [MFMZ14]. These context switches can be time-intensive, as developers need to restore a task's working context, including all associated artifacts (e.g., applications, documents, and folders). Moreover, the mental task context needs to be captured and re-established, which comprises task knowledge, goals, and intentions related to a task [PD10]. One way to reduce the fragmentation of work is to reduce external interruptions at in-opportune moments [ZCM$^+$17] or support users in staying focused on a task [RKF$^+$20, PRM$^+$20]. However, in reality, interruptions are not completely avoidable, and developers themselves might often need or want to switch between tasks. In both cases, developers could benefit from support with capturing the current working and mental task context and help them to restore the contexts later.

Previous research tried to support task context capturing and resumption by allowing users to relate recently used artifacts to tasks. *Mylyn* by Kersten & Murphy [KM06] collects and highlights task-related source code files within the IDE. Other approaches, such as *TaskTracer* by Dragunov et al. [DDJ$^+$05], generalize this approach to operating system level and provide the possibility to restore all artifacts related to a task automatically [SBR$^+$03, HL22]. Usually, these approaches either work fully automated, which can result in restoring task-unrelated artifacts, or the user must manually select artifacts, which can be cumbersome. In addition, while existing approaches help to capture and restore the working task context, support for the mental context is limited [PD10]. Approaches that only try to support restoring the mental context exist [PD10, WKM$^+$18, RTH17, SFKD22], for example, to help with work detachment and reattachment [WKM$^+$18]. However, existing approaches are not designed to support users in dealing with interruptions and seldom explicitly target software developers, for example, by not including source code related information as visual cues. Further, to the best of our knowledge, no existing approach explicitly tries to support restoring working <u>and</u> mental task context. Related work is discussed in more detail in Chapter 2.

In this thesis, we propose a new approach that supports software developers and data scientists in task context capturing and resumption by semi-automatically creating snapshots of a task's associated working and mental context on operating system level. Snapshots can be created at any time, for example, when being interrupted or at the end of the workday. Subsequently, users can curate them by selecting relevant artifacts and writing down task-related information as soon as they have time. Later, the approach helps users to re-identify and restore a snapshot to support task resumption. Chapter 3 presents the core concepts of the approach.

Our proposed approach is centered around two main questions: On the one hand, little is known about how software developers and data scientists (henceforth referred to as "developers") keep track of task-related information and how they deal with switching and resuming tasks. Thus, we want to understand better:

- RQ1: How do developers capture their working and mental task context, and how do they manage task resumption today?

On the other hand, we want to find out how we can support developers in these activities without negatively affecting their existing workflows. Therefore, we want to investigate:

- RQ2: How should a semi-automated task context snapshot tool to capture and resume the working and mental task context (a) be designed to be seamlessly integrated into developers' existing workflows and (b) what is its impact?

To answer our research questions, we implemented the proposed approach with TaskSnap, a semi-automated software to create, curate and restore snapshots of a task's context at any time during work. With the TaskSnap tool, the user can instantly create a snapshot by pressing a physical or virtual button and then curate it immediately or postpone the curation in inopportune moments. When curating, the user can select relevant artifacts (applications, files, folders, and web pages) based on a pre-selected list, summarize just finished actions, express intents, and finally clean up no longer needed artifacts. Later, artifacts captured by a snapshot can be automatically reopened to restore the working context of a task, and information is displayed that should help users to restore their mental task context. In Chapter 4, the TaskSnap tool is described in more detail.

To evaluate the developed approach, we conducted a two-week explorative pilot study in the field with six developers. After an initial survey to learn about participants' existing task context capturing and resumption routines, a five-day baseline phase was performed. In the baseline phase, the TaskSnap tool was running in the background to collect information about the artifacts the user typically keeps open. Then, in an intervention phase that lasted for five workdays, all features of the TaskSnap tool were available, and participants were asked to try to incorporate the tool into their existing workflows. In both study phases, experience sampling was used to get insights about the impact of task switches performed by the user and to collect more details about created snapshots. The pilot study ended with a final interview to get qualitative feedback. The study method is presented in Chapter 5.

The results of the pilot study show that participants have varying existing strategies to preserve their working and mental task context, while most participants apply note-taking to capture their mental context. Overall, users reported being very satisfied with the approach, that it reduced the fear of forgetting to persist artifacts, and saved them some time when restoring a task's context. On average, 1.24 snapshots were created per day and user, and participants created snapshots mainly when having enough time, for instance, at the end of the workday. Creating snapshots during instant task switches was less common, as the participants did not encounter these situations frequently. Moreover, while users reported that the approach helped them to declutter their workspace, the number of open artifacts did not generally decrease between the two study phases. All results of the pilot study are provided in Chapter 6.

We discuss the main findings from the pilot study and elaborate on potential directions of future work in Chapter 7. Finally, potential threats and limitations are discussed in Chapter 8 and Chapter 9 concludes this thesis by summarizing the main results.

The main contributions of this work are the following:

- We **introduce a novel approach** that supports developers in capturing and resuming tasks by allowing them to semi-automatically create snapshots of their task context at any time.

- We provide **initial insights** into using such an approach from a **two-week pilot study in the field** with six developers, including computer interaction data that show users' artifact opening and closing behavior.
- We present **findings** that contribute to our understanding of **how developers capture their working and mental task context today** based on data collected through experience sampling, a survey, and interviews.

Chapter 2

Related Work

Previous research has discussed various topics regarding task switching, capturing a task's context, and task resumption. Section 2.1 gives an overview of findings regarding the fragmentation of developers' workdays to elaborate on the need for task context capturing and resumption support. As described in the introduction, the context of a task consists of working context and mental context. Literature about capturing and restoring the working context of a task is presented in Section 2.2, while Section 2.3 discusses research regarding capturing and restoring the mental context.

The term *task* is not unambiguously defined in related work. In this thesis, we define a task as "a usually assigned [to a person] piece of work often to be finished within a certain time" [MW03] comprising of "a set of interrelated events, which share a common motive (or goal)" [GM04].

2.1 Fragmentation of Developers' Workdays

The workday of knowledge workers is highly fragmented. Many work on multiple tasks per day and are frequently switching between them [MBM+17, GM04]. Research found that knowledge workers spend only between 6.2 and 13.2 minutes on a task before switching [GM04, MFMZ14, MSZ+22]. Similar results were found for software developers, who work on two to ten tasks a day and switch between them more than thirteen times per hour on average [MFMZ14]. Also within the bounds of a task, many context switches happen. Meyer et al. [MBM+17] found that software developers work only between 0.3 and 2.0 minutes before switching from one activity (e.g., editing code, reading documents, or internet browsing) to another. Frequently changing contexts can be problematic for developer productivity, as it introduces time overhead to recall and restore task-related information and increases errors and loss of knowledge [PD10, KM06, PR11]. Further, frequent switches make it difficult to remember what someone worked on at which time [SFM21]. A primary reason for task switching is interruptions. These can emerge from external sources such as planned meetings, co-workers that ask for help, or noise in the office, but interruptions can also be self-initiated when being stuck [MBM+17] or leaving the desk to interact with a team member [GM04]. Previous work estimated that internal interruptions occur approximately as often as external interruptions over a workday [GM04, CHW04] and are even more disruptive than the latter [KBVVT16]. While the number of external interruptions can be reduced by increasing the awareness about the productivity impact interruptions have [SBL98] or reducing the number of interruptions at work with technical aids [ZCM+17, RKF+20, PRM+20], many interruptions are difficult for developers to avoid. Some are urgent and cannot be postponed, or the developer even wants to handle the disruption immediately [SBL98]. Further, it is important to note that not all interruptions are harmful. Some are required for collaboration [RTH17] or support problem solving [SBL98]. Therefore, means to handle interruptions immediately when they occur are valuable. The approach proposed in this work aims to help developers to deal with interruptions right when they occur by supporting them with task context capturing and resumption and, by doing so, making existing workflows more efficient.

2.2 Capturing & Restoring Working Context

Developers apply different strategies to keep the cost of interruptions and task resumption as low as possible. As refinding artifacts, such as applications, files, emails, or web pages, that are relevant for a task can be challenging and time-consuming, a common strategy is to leave application windows open, even though they are no longer needed for the task the user switched to [HL22, PRM$^+$20]. In a survey by Hu & Won Lee [HL22], 45% of the participating computer users reported applying this strategy. By following this approach, application windows might stay open for a long time, which affects a computer's performance and leads to clutter [HL22]. Especially as developers frequently use multiple applications to complete a task [Maa09, MFMZ14]. Another often applied approach is manually organizing documents according to tasks by creating project folders. However, this approach only works within one application type, for example, in the file system or the browser, but not across applications. Further, issues arise when a file is related to multiple projects [Kap03].

Tool Support to Manually Capture Working Context. Technical aids can help users to organize task-related artifacts manually by grouping, for example, by using virtual desktops [CH86, RHC$^+$04, SBR$^+$03]. *GroupBar* by Smith et al. [SBR$^+$03] extends the functionality of the Microsoft Windows taskbar by allowing the user to group windows together, for example, according to tasks. A similar approach is chosen in *Scalable Fabric* by Robertson et al. [RHC$^+$04], in which resized windows can be grouped at the edge of the computer screen. However, a big issue of these dedicated task spaces is time overhead. Artifacts have to be manually arranged, updated, and in the end, cleaned up [Kap03]. Further, it is usually faster to reuse an existing application window in a task-unrelated group than to open a new one in the correct context [DDJ$^+$05]. This overhead could be the reason why manual approaches like the presented ones are not prevalently used [Kap03].

Automated Approaches to Capture & Restore Working Context. Previous work proposed approaches that automate parts of the context-capturing and task-resumption process. Kersten & Murphy [KM06] developed a plug-in for the Eclipse IDE called *Mylyn* that allows users to define tasks and later highlights artifacts frequently used while working on the selected task. Another example is the browser extension *OneTab*[1] that lets users save, group, and relaunch browser tabs. While these approaches work within a single application, like the IDE or browser, *TaskTracer* by Dragunov et al. [DDJ$^+$05] works on operating system level. In their approach, artifacts (such as files, emails, and web pages) the user interacts with are automatically associated with the currently active task. Then, after a task switch, *TaskTracer* can restore all associated artifacts automatically. Related, the commercially available application *Rewind*[2] captures and analyzes every activity made on the computer and allows users later to search through the recordings to refind information. Screen recording or screenshot-based approaches for task identification and resumption were also successfully applied in previous research [HL20, SM07]. In *Scrapbook* by Hu & Won Lee [HL22], the user can take a screenshot, and the tool automatically stores references to

[1]`https://www.one-tab.com`, verified 04.07.2023
[2]`https://www.rewind.ai`, verified 04.07.2023

all application windows visible in the selected screen area. Next, the user can curate captured artifacts by manually deselecting resources irrelevant to a task and providing a general description. Later, the user can browse a gallery view of created screenshots to relaunch a set of windows. A field study showed that participants saw value in the approach, and screenshots taken by users contained various types of applications. Users also created screenshots as visual reminders of tasks. However, participants raised concerns that personal data could appear in screenshots and further stated that the tool should also support capturing and restoring web pages.

Shortcomings of Existing Approaches. The presented approaches have some limitations we address with the approach proposed in this thesis. Existing solutions either require the user to manually curate artifacts, which can be time-consuming, or operate fully automated, which can lead to capturing and restoring task-unrelated artifacts. For this reason, in our approach, artifacts are curated using a semi-automated process by leaving the final decision on which artifacts are relevant with the user. Further, besides capturing a task's working context, not all presented approaches can automatically reopen artifacts, which we include, and in contrast to *Mylyn* [KM06] or *OneTab*[3], we are not focusing on tasks within a specific application, like the IDE or browser, but operate on operation system level. *Scrapbook* [HL22] is probably the closest related to our approach. In contrast to this approach, we do not rely on screenshots to visualize tasks. Source code often looks visually repetitive [PR12], which makes screenshots not well applicable to software development tasks that we want to support. In addition, as developers spend much time in the browser [MBM$^+$17], we further include web pages as task artifacts, which is missing in *Scrapbook* [HL22].

2.3 Capturing & Restoring Mental Context

Software development tasks require creating complex mental models that must be restored when a task is continued. The so-called *mental context* comprises knowledge and thoughts about a task, open issues, next steps, and goals. Research has shown that the mental task context fades quickly from memory when being interrupted [RTH17] and that restoring it can be challenging [PD10, IH07]. Restoring the mental task context can take around 15 minutes, which makes it hard to work efficiently if too many interruptions occur [SBL98]. The approaches presented in Section 2.2 have in common that they only focus on restoring the user's working task context. While they help to keep the workspace organized, it is unclear whether they also support restoring the mental context [PD10]. Generally, only a few prior approaches have explicitly focused on helping to restore this task context type and were evaluated for their effectiveness in supporting programming activities [RTH17]. For this reason, the approach we present in this work goes beyond simply capturing and restoring task artifacts by providing means to support capturing and restoring the mental task context of developers.

Existing Strategies of Developers. Developers use different strategies to capture their mental task context and reduce the time needed to resume interrupted tasks. While they frequently take physical or digital notes [PD10, RTH17, GM04], developers also leave in-line code comments or deliberately write code that will throw errors to guide them back to the position where they left off [RTH17]. A limitation of passive reminders such as in-line "TODO" comments and external notes is that they do not seek the user's attention when resuming a task and can get forgotten. Writing code that throws errors is also not ideal, as it prevents switching to a task in the same

[3]`https://www.one-tab.com`, verified 04.07.2023

codebase [PR12]. These limitations can be overcome easily by digital aids like the one presented in this thesis.

Theoretical Background. Altmann & Trafton [AT02] propose with the *goal-activation model* a theoretical background for interruption recovery. According to their model, each goal has a degree of mental activation, whereby the goal with the highest activation governs human behavior (see Figure 2.1). The activation of goals decays over time, and new goals become more active. Therefore, to successfully resume a task, the activations of its affiliated goals have to be increased again. This can be achieved by encountering mental or physical cues that are associatively linked to a particular goal. In the domain of software development, such visual cues can be application windows, tabs, method names, or in-line comments [PD10]. It is important, though, that besides being present at the time of goal retrieval, cues are also present before goal suspension, for example, when switching a task. Further, according to the model, the rehearsal of the current goal before suspension can increase its baseline activation level, which will help to make it the dominant goal again later. In the context of interruptions, this time window right before goal suspension, when an alert is present, but the user has not yet switched the task, is called interruption lag. Altmann & Trafton state that this time window is critical for the success of task resumption, as associative cues can be formed and goals rehearsed [TABM03].

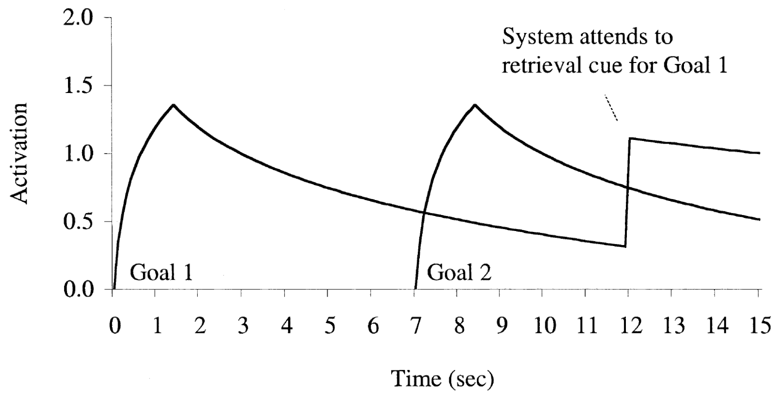


Figure 2.1: Goal-activation model by Altmann & Trafton [AT02].

Cues That Support Task Resumption. Visual cues can trigger recognition and can hence support restoring the mental task context [RTH17]. Different visual cues can be helpful for developers. A common approach is quickly writing down relevant information on a piece of paper and placing it somewhere close by to get reminded about the issue [BVKKS08]. The position of file tabs and scroll bars in the IDE can also help developers to reestablish a mental model by pointing to a specific code segment. However, the spatial position of these elements is very unstable, and tabs frequently just display a file's name, which only provides limited associativity [PR12, KMCA06]. If the user anticipates that an interruption will only last for a short time, it is also a common strategy to leave a specific part of an artifact visible [PD10]. Further, several approaches, *Scrapbook* presented in Section 2.2 for instance, use screenshots to trigger associative memory [HL22, HL20, RTH17, SM07].

Parnin & DeLine [PD10] asked 414 software developers to gather feedback about possible future tools that could support users in task resumption by providing helpful visual cues. The three best-rated ideas were (1) providing a view that displays recent code changes, (2) providing a natural language summary of recent actions, and (3) taking and visualizing images of places recently edited or navigated to. Following the survey results, the authors implemented the idea of a code change history visualized as a timeline. They stated positive user feedback and improved task performance compared to simple note-taking. These findings suggest that digital tools that make use of visual cues can help users to resume tasks. Based on the findings, we decided to integrate visual cues in our approach that trigger recognition and help users to restore their mental task context, but we do not make use of screenshots, as they are not suitable for depicting many software development tasks [PR12]. Instead, we want to use source code related information to target the needs of developers. Further, we integrate the not yet implemented idea suggested by Parnin & DeLine [PD10] of providing users with a natural language summary of recent actions.

Preparation Before Task Suspension. Giving a user already a few seconds to rehearse a task before dealing with an interruption positively impacts task resumption performance [TABM03, SDNL19]. The interruption lag could either be used to state a goal that should be pursued after the interruption is handled or to gather information about the current task's state needed to resume it later. The authors of the *goal-activation model* refer to these two strategies as *prospective goal encoding* and *retrospective rehearsal* [TABM03]. Literature about digital tools that could support users during the interruption lag is sparse. By following the model proposed by Altmann & Trafton [TABM03], Williams et al. [WKM$^+$18] investigated how structured dialogues that reflect on work-related tasks can be used to help knowledge workers to detach from and reattach to work. By interacting with a chatbot, participants of a user study were asked at the end of each day what they were working on during that day (*retrospective rehearsal*) and about their goals for the next day (*prospective goal encoding*). In the morning, the chatbot asked questions for goal confirmation and goal priming. The authors' results show that participating in these dialogues made users feel more productive in the first hour of the workday. Hence, reattachment to work was supported by their approach. In contrast to their work, our primary goal is not to support users in work detachment but with interruptions immediately when they occur. While the authors stated that work detachment and reattachment shares similarities to task interruption and resumption, their approach was not tested for this use case, as they focused on supporting work detachment.

Chapter 3

Approach

To answer our research questions, we developed a semi-automated approach that supports developers in capturing and resuming tasks. This is achieved by allowing the user at any time to instantly *create* a snapshot of the current working and mental task context on operating system level, for example, when being interrupted or at the end of the workday. Subsequently, the user can *curate* the snapshot as soon as they have time by selecting relevant artifacts from a pre-selected list and providing additional textual information about the task. Later, the approach helps users to re-identify and *restore* a snapshot to facilitate task resumption. This chapter discusses the key considerations of the approach, including the foundation on just-in-time snapshots (Section 3.1), making the snapshot creation and curation semi-automated (Section 3.2), and how restoring a task's context is supported (Section 3.3).

3.1 Snapshot Based Approach

Automatically identifying the task context and context switches of developers is challenging [MMV+01], among other things, because developers frequently switch between tasks [MBM+17, GM04] and a context switch not necessarily involves changing the application [MFMZ14]. Even if semantic information about work artifacts is considered, the accuracy and applicability of approaches are limited [SFM21]. For this reason, in our developed approach, the user must manually indicate when a task is switched, and a snapshot of the current context will be created. This procedure has the advantage of capturing only those task contexts that are valuable for the user to preserve. Some tasks might be finished or not planned to be continued for other reasons; hence, capturing them is unnecessary. Relying on snapshots instead of continuously building up a task context further circumvents the issue of task evolution, which could make assigning artifacts to tasks even more challenging. Artifacts can also evolve: A document could be created to pursue task A but continued the next day in the context of task B [BSW08]. By applying a snapshot approach, we do not have to deal with this possible artifact evolution, as we are only interested in the task context at the time the snapshot is created. In addition, including all items accessed while working on a task likely results in information overload [Kap03]. This can be avoided by only considering artifacts that are open when the snapshot is created. Lastly, many existing approaches require the user to define new tasks upfront [KM06, DDJ+05, SFM21, Kap03], which could be cumbersome for users or be forgotten. A snapshot approach avoids this issue.

3.2 Semi-Automated Snapshot Creation & Curation

Snapshot Creation. In our approach, snapshots can be created at any point in time to instantly support the user in capturing a task's context. When a snapshot is created, references to all artifacts open at this point are automatically stored. Previous work sometimes focused only on artifacts within a specific application type [KM06], but as developers use four to five different tools on average while working on a task [Maa09], our approach works on operating system level to capture a task in its entirety. Considered artifacts are open applications and their associated open files, including the titles of application windows, as they may contain valuable information describing the captured task [SFM21]. Further, open web pages and file system folders are considered as developers spend significant time in the browser and file system [MBM^{+}17]. Moreover, information from the version control system is stored when creating a snapshot to cover developer-specific needs. Information from the version control system that could be valuable to capture is the last commit message or the branch the user was working on.

Curating Working Context. While all open artifacts are captured automatically by the approach when creating a snapshot, not all artifacts might be task-related. Therefore, snapshots must be *curated*, meaning to decide which artifacts are actually relevant. The presence of task-unrelated artifacts can have many reasons. Users might already have stopped working on the task that a snapshot should capture, artifacts from previous tasks could still be present (see Section 2.2), or the user opened unrelated emails or web pages. Automatically deciding which artifacts are relevant would be ideal but likely not perfectly accurate. Manual selection would be highly accurate but possibly too time-consuming. Therefore, we decided to apply a semi-automated snapshot curation process, meaning that the user receives a list of all captured artifacts that is already preselected based on insights drawn from the artifacts' interaction history. Still, the user makes the final decision on which artifacts belong to a task and should be kept. Besides saving the user some time, the pre-selection can also be valuable to not overwhelm them when many artifacts are present in their workspace. This is an issue the authors of *Scrapbook* reported. As users might have many web pages open, the *Scrapbook*-prototype does not support capturing this artifact type. To solve this issue, the authors propose to add "an intelligent algorithm that can evaluate the relevance of inactive tabs" [HL22]. Our approach includes this suggestion by estimating relevance based on the artifacts' interaction history. A further advantage of the semi-automated approach is that the user can manually deselect artifacts that were task-related but are of no relevance when continuing the task again. Detecting and deselecting these artifacts automatically would not be feasible with a fully-automated approach.

As a snapshot might be created during a task switch that has to be performed quickly, we cannot be sure that the user has time to curate it immediately. Therefore, curating a snapshot is possible at any point in time: Right after its creation, after postponing it, or when restoring the snapshot. Then, as soon as the user had time to curate the snapshot, all associated artifacts can be automatically closed to reduce workspace clutter.

Curating Mental Context. Based on the *goal-activation model* presented in Section 2.3, our approach provides means to semi-automatically capture the mental task context of a user. The authors of the model state two strategies to support context preservation: *retrospective rehearsal* and *prospective goal encoding*. We address *retrospective rehearsal* in our approach by prompting the user to think about recently performed actions and encouraging them to write them down as a short summary when curating a snapshot. To achieve this, the user is asked to answer "Now what was I doing?". However, summarizing the user's last performed actions might not be enough to restore the mental task context, as the summary does not convey further steps needed to complete a task [RTH17]. Therefore, we address *prospective goal encoding* by encouraging the user in the curation step to state their intent on how the task should be continued. Here, the question "Now what was I about to do?" is asked. These questions correspond to the ones proposed by the authors of the *goal-activation model* [TABM03]. While answering the two questions could also be done in a separate, dedicated note-taking application, switching to another application introduces additional navigational and cognitive costs [BVKKS08]. This is why the user can note down information regarding the two concepts in our approach in the same place where the snapshot's artifacts are curated. Further, by including this functionality directly into the approach, the two texts can automatically be pre-filled with information already gathered about a task, which follows an idea proposed by previous work [PD10]. The user can read the pre-written texts to start reflecting and then modify or rewrite them. This semi-automated approach reduces the time needed to curate a snapshot, which could be crucial as the user may only have little time to rehearse a task after being interrupted and required to switch tasks. Further, users are generally overconfident in their knowledge and ability to remember information [LFP82], leading to the impression that some task details do not need to be written down [BVKKS08]. Providing a semi-automatically generated summary counteracts this effect.

Besides the texts that summarize recently performed actions and intents, the user can give the snapshot a descriptive title in the curation step that can also capture parts of the mental context. This title can later also be helpful to re-identify a snapshot before restoring it.

3.3 Restoring Working & Mental Context

To support developers in resuming tasks, context snapshots can be restored. The approach provides an overview of existing snapshots and helps users re-identify the snapshot they want to restore by using visual cues, like displaying a snapshot's title, associated artifacts, and provided texts regarding recently made actions and stated intent. We explicitly decided against using screenshots at this point, because they might not be suitable for visualizing software development tasks, as source code looks visually repetitive [PR12]. Then, when restoring a snapshot, all artifacts of a task's working context are automatically reopened. Further, following the *goal-activation-model*, the written texts capturing the user's mental task context are automatically displayed to activate the task's associated goals and help to reestablish the mental model. Automatically showing the texts can also be beneficial as even if a user noted down information, the information is not always taken into account when resuming a task [BVKKS08].

Restoring the mental context is supported by further, more subtle means. As mentioned earlier, when storing references to task-related artifacts, the window titles of applications and information from the version control system are preserved, as they may contain valuable information to get back into a task. To start rebuilding the mental task context, the window titles and the captured data from the version control system are displayed when inspecting a snapshot before restoring it.

Chapter 4

Implementation

Based on the approach presented in the chapter before, we implemented a technology probe called *TaskSnap*. The TaskSnap tool is a multi-platform desktop application that mainly runs in the background and is connected to the user's browser and the IDE Visual Studio Code using custom-built extensions. Section 4.2 gives a more detailed overview of the different components. At any point, the user can instantly create a snapshot of the current task context that contains currently open applications, files, folders, and web pages by clicking a physical or virtual button (Section 4.1.1). After a snapshot is created, the user can curate it as soon as they have time by setting a descriptive title and deciding which artifacts belong to the working context of a task. The user is further prompted to write down recently performed actions and future intents to support capturing the mental task context (Section 4.1.2). To make the curation step as efficient as possible, it is semi-automated. The list of artifacts is already pre-selected based on an algorithm that considers the artifacts' interaction history (Section 4.3), and the texts regarding recent actions and intents are pre-filled with an automatically generated draft (Section 4.4). After curation, the user can automatically close all task-related artifacts to reduce workspace clutter. Later, the user can inspect, adjust, and manage snapshots from a list view where all existing snapshots are visualized. The user can further select a snapshot to restore it, which will automatically reopen all selected artifacts, and the texts that capture the mental context are displayed (Section 4.1.3). Noteworthy, all data that is automatically retrieved from the system or provided by the user in these steps is only stored locally to preserve the user's privacy.

4.1 User Workflow

The user workflow to create, curate, and restore a snapshot is described in the following three sections. To visually illustrate the different steps, images are provided that depict a snapshot of an imaginary front-end development task.

4.1.1 Snapshot Creation

At any point during the workday, the user can instantly capture the currently active task context, for example, when switching tasks. By pressing a physical button, using a customizable shortcut, or clicking the application icon in the macOS tray or Windows taskbar, a snapshot is created, and the TaskSnap tool collects references to all currently open artifacts (see Section 4.2 for how artifacts are collected). In parallel, a camera shutter sound is played, the physical button flashes (if connected), and a small pop-up at the border of the screen appears (depicted in Figure 4.1).

In this window, the user can decide how to proceed with the just-created snapshot. A text field is provided to give the snapshot a title quickly. Users can utilize this instantly assigned title to choose a more descriptive one later when more time is available [RTH17]. The user can further delete the snapshot again if it was created by accident and has the option to curate it immediately, which will open a more detailed overview of the snapshot (the *Snapshot Curation Window*, see Section 4.1.2). Snapshot curation can also be postponed by selecting a time, after which the tool prompts the user to curate by showing the *Snapshot Curation Window*. The user might work on the same task regularly. For this reason, the user can further amend the created snapshot to an existing one. Lastly, the user can choose to close all artifacts that were automatically associated with the task (see Section 4.3 for more information about the automatic pre-selection of artifacts).

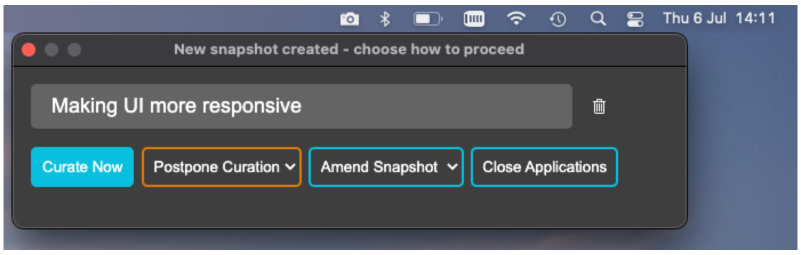


Figure 4.1: Window that appears when a snapshot is created.

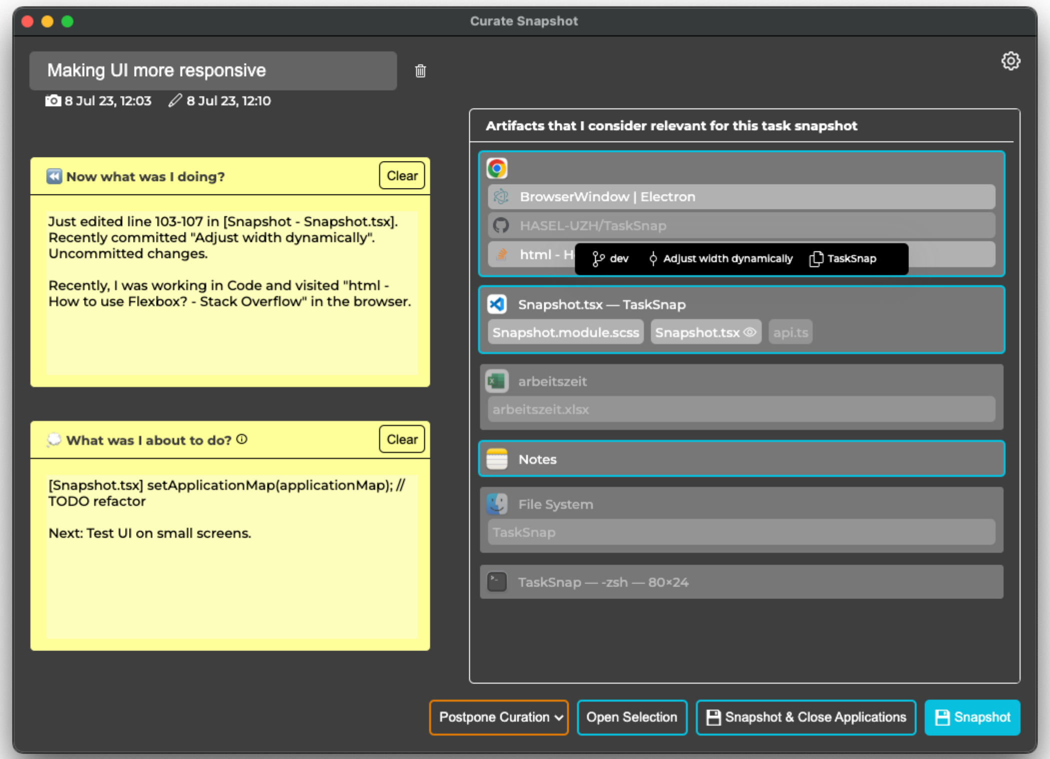
4.1.2 Snapshot Curation

The user can curate a snapshot whenever it fits them best by selecting relevant artifacts from a pre-selected list and summarizing recently performed actions and future intents. To do so, a dedicated window is provided where the user can inspect and update the content of an existing snapshot. A screenshot of the *Snapshot Curation Window* is shown in Figure 4.2. At the top, the snapshot's title is displayed and can be edited, and the snapshot's creation and last edit date are visible. On the left, the two editable texts that should help users to capture and later restore the mental context of a task are displayed. On the right, the working context of a task is depicted by showing the pre-selected list of the snapshot's associated artifacts. The following paragraphs explain the curation step in detail.

Curating Mental Context. As motivated in the approach in Section 3.3, the TaskSnap tool provides two text fields to support capturing the user's mental task context. The first text field has the title "Now what was I doing?" and should prompt the user to reflect on actions recently performed and write them down. For example, a user could provide information about which code section was just edited or which web search was recently conducted. The second text field starts with the question "Now what was I about to do?" and should encourage users to write about their current intent on how the captured task should be continued. The titles of the two sections are chosen to match the questions proposed by Altmann & Trafton [AT02]. Both text fields are automatically pre-filled to spark ideas and save the user some time. The first text field is filled with information about recently edited code in the IDE, just-visited web pages, and the application that was recently used the most. To the second text field, edited lines of code that include TODO messages are automatically added. The automatic text generation is presented in more detail in Section 4.4. The user can adapt and extend the generated texts or start from scratch by clicking the "Clear" button in the top right corner. As physical Post-It notes are still a common tool to capture information about a task and open issues [PD10, BVKKS08], the two text fields are visualized in the TaskSnap tool in a way to resemble Post-It notes.

Curating Working Context. As not all open artifacts may be relevant, the user can curate the task's working context. To accomplish this, the user is provided with a list of all open artifacts grouped by application and can select or deselect items by clicking on the respective entries. To save the user some time, curating the working context is semi-automated, meaning that the list of artifacts is already pre-selected (see Section 4.3). Deselected artifacts are grayed out and will not be opened when restoring the task snapshot later.

To support the user in identifying artifacts, each application's name, icon, and window title are depicted. When additional artifacts, such as web pages, folders, or files, are associated with an application, they are listed together. For web pages, the page's title is shown alongside its Favicon, and an eye icon indicates the web page that was active when the user created the snapshot. The same approach was chosen for files open in the IDE. The browser and IDE always appear at the top of the list, as we have richer information available for these applications through the extensions. All other applications are sorted by relevance (see Section 4.3 for how relevance is calculated). The same sorting is applied for artifacts associated with one application.


Curate Snapshot

Making UI more responsive

8 Jul 23, 12:03 8 Jul 23, 12:10

Now what was I doing? Clear

Just edited line 103-107 in [Snapshot - Snapshot.tsx].
Recently committed "Adjust width dynamically".
Uncommitted changes.

Recently, I was working in Code and visited "html - How to use Flexbox? - Stack Overflow" in the browser.

What was I about to do? Clear

[Snapshot.tsx] setApplicationMap(applicationMap); // TODO refactor

Next: Test UI on small screens.

Artifacts that I consider relevant for this task snapshot

BrowserWindow | Electron
HASEL-UZH/TaskSnap
html - H
dev Adjust width dynamically TaskSnap
Snapshot.tsx — TaskSnap
Snapshot.module.scss Snapshot.tsx api.ts
arbeitszeit
arbeitszeit.xlsx
Notes
File System
TaskSnap
TaskSnap — -zsh — 80×24

Postpone Curation Open Selection Snapshot & Close Applications Snapshot

Figure 4.2: Snapshot Curation Window. Hovering over the Visual Studio Code entry shows the checked-out branch of the active Git project, the last commit message, and the workspace's name.

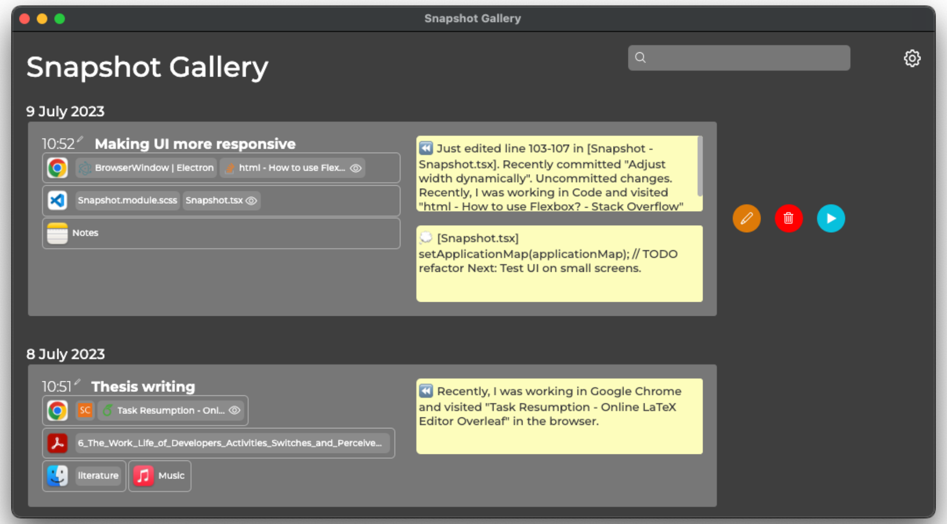
Further Options When Curating. At the bottom of the *Snapshot Curation Window*, the user is provided with further options. They can postpone the curation or open all selected artifacts, which might be helpful when curating a snapshot right before resuming a task. In addition, the user can save changes and close all selected artifacts to clean up the workspace. However, there might be applications that are relevant for a task but undesirable to close, for example, a mail client, the browser, or team collaboration software that are useful for every task. For this reason, the user can specify applications that will never be closed by the TaskSnap tool on the settings page. The settings page of the TaskSnap tool can be accessed by clicking the gear icon in the top right corner of the *Snapshot Curation Window*.

4.1.3 Snapshot Resumption

When resuming a task, a snapshot can be identified, inspected, and restored from the *Snapshot Gallery*, which provides an overview of all created snapshots. In a view with a similar purpose in *Scrapbook* [HL22], screenshots were used to visualize captured tasks. However, as source code is often visually repetitive [PR12] and users of *Scrapbook* raised the concern that personal data could be visible on the displayed screenshots [HL22], we decided to use a different visualization approach in the TaskSnap tool. Research has shown that humans rely on temporal and semantic associations to recall memories [Tul04]. Therefore, we visualized existing snapshots along a timeline. The user further has the option to filter the shown snapshots with a text-based search. The *Snapshot Gallery* is depicted in Figure 4.3.

For each snapshot, a preview is displayed that should help users to re-identify a task context. The preview contains the snapshot's title and creation or last edit date. Further, all artifacts associated with the snapshot that were selected during snapshot curation are listed. The sorting of artifacts corresponds to the one present when curating the snapshot, and clicking on an artifact opens it automatically. To make the preview as compact as possible, not all available information is visible by default. For example, only the Favicon is shown for web pages that were not active when the snapshot was created. The user can expand a snapshot preview to see more detailed information by clicking, as depicted for the first snapshot in Figure 4.3. Hovering over an artifact reveals even more helpful information. By hovering over an application icon, the window title is shown. Placing the cursor above an IDE entry will display the current branch, last commit message, and workspace name, and hovering over a file reveals the file path. On the right of the artifacts, the texts that capture and support restoring the mental task context are shown. The summary of just performed actions before creating the snapshot is always displayed. The user's intent is only shown when the snapshot preview is expanded. Also in this view, the two sections are visualized in a way to resemble Post-It notes. By hovering over a snapshot preview, buttons for possible further actions are displayed. The user has the option to start curating a snapshot, delete it with all its associated data, or restore it.

When the user chooses to restore a snapshot, a separate window pops up that displays the summary of recent actions and task intent to help the user to re-establish the mental task context (visible in Figure 4.4). After the window is loaded, the TaskSnap tool automatically opens the snapshot's associated artifacts that were selected during the curation step. By doing so, the user has time to read the window's content while artifacts are opening, which can take some time.


Snapshot Gallery

Snapshot Gallery

9 July 2023

10:52 **Making UI more responsive**

BrowserWindow | Electron · html - How to use Flex...

Snapshot.module.scss · Snapshot.tsx

Notes

Just edited line 103-107 in [Snapshot - Snapshot.tsx]. Recently committed "Adjust width dynamically". Uncommitted changes. Recently, I was working in Code and visited "html - How to use Flexbox? - Stack Overflow"

[Snapshot.tsx] setApplicationMap(applicationMap); // TODO refactor Next: Test UI on small screens.

8 July 2023

10:51 **Thesis writing**

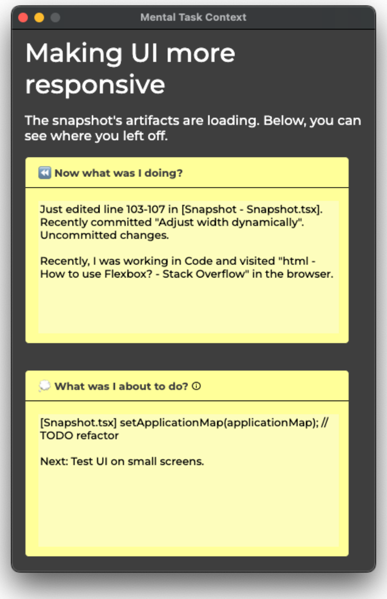
SC · Task Resumption - Onl...

6_The_Work_Life_of_Developers_Activities_Switches_and_Perceive...

literature · Music

Recently, I was working in Google Chrome and visited "Task Resumption - Online LaTeX Editor Overleaf" in the browser.

Figure 4.3: Snapshot Gallery. The entry "Making UI more responsive" was expanded by clicking on it. By hovering over the entry, the user triggered the appearance of buttons for curating, deleting, and restoring the snapshot.


Mental Task Context

Making UI more responsive

The snapshot's artifacts are loading. Below, you can see where you left off.

Now what was I doing?

Just edited line 103-107 in [Snapshot - Snapshot.tsx]. Recently committed "Adjust width dynamically". Uncommitted changes.

Recently, I was working in Code and visited "html - How to use Flexbox? - Stack Overflow" in the browser.

What was I about to do?

[Snapshot.tsx] setApplicationMap(applicationMap); // TODO refactor

Next: Test UI on small screens.

Figure 4.4: Mental Context Window that appears when restoring a snapshot.

4.2 Artifact Capturing

To capture all artifacts belonging to a task, the TaskSnap tool consists of three software components: A desktop application written in TypeScript using the Electron framework[1] that runs on macOS and Windows, a browser extension for Chromium[2]-based browsers and *Mozilla Firefox*, and an extension for the IDE Visual Studio Code. The desktop application is the only component of the tool the user interacts with and is the primary data source to collect system-wide artifacts. It is closely attached to a local database, where data is stored exclusively to preserve the user's privacy. Moreover, the desktop application communicates with the operating system to get, open, and close artifacts. More information is provided in Section 4.2.1. The desktop application is connected to an extension for the browser that captures the user's open web pages and which page is currently active (see Section 4.2.2). Similarly, an extension for the IDE Visual Studio Code is connected to retrieve the user's open files, information about recently made edits, and Git-related data (Section 4.2.3). The other way around, the communication channels to the extensions are used to send open or close requests for web pages or files open in the IDE. We created an extension for these two application types, as developers spend much of their time in browsers and IDEs [MBM+17]. Hence, more detailed information from these contexts could be valuable. Optionally, the user can connect a physical button via USB to their computer to create a snapshot with a button press. Currently, only a single product is supported, the *Luxafor Mute Button*[3]. However, more products could be easily added in the future. Figure 4.5 depicts the overarching architecture.

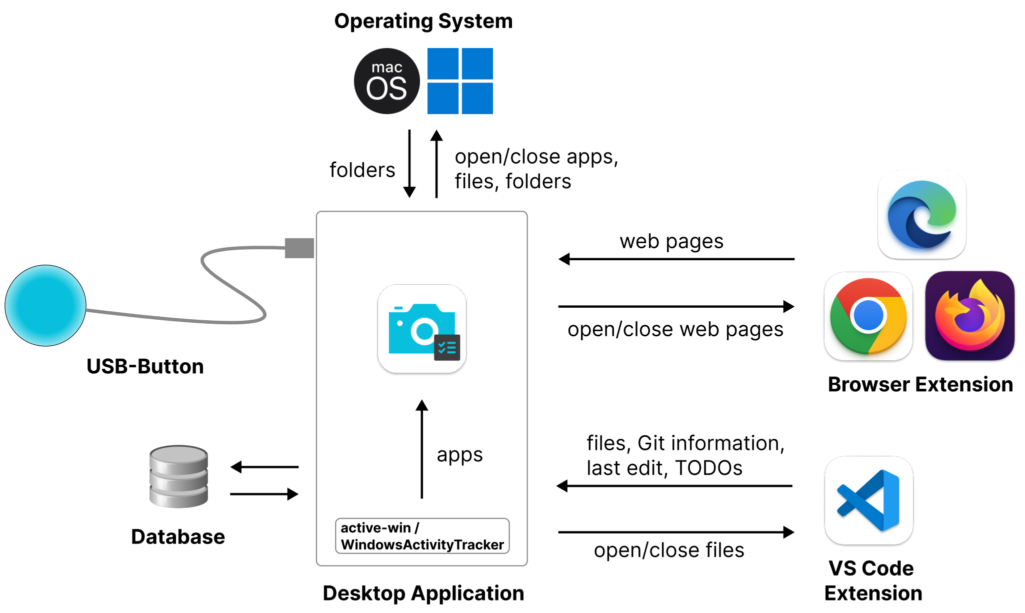


Figure 4.5: Architecture overview.

[1] https://www.electronjs.org, verified 06.07.2023
[2] https://www.chromium.org/Home/, verified 06.07.2023
[3] https://luxafor.com/product/luxafor-mute-button, verified 06.07.2023

4.2.1 System-Wide Artifact Capturing

The desktop application collects system-wide information regarding currently open applications, files, and folders. To capture artifacts, the basic idea is to collect the set of open application windows, including their window titles, and then retrieve artifacts based on this information. To achieve this, the npm package *active-win*[4] is used. The package provides an API endpoint that returns metadata about all open application windows, including information about each window's owning process. The owning process is the application the window belongs to, and the API returns the owner's name, process ID, and path. At the time a snapshot is created, the TaskSnap tool calls the endpoint, and for each retrieved application, its icon is fetched by using the npm package *extract-file-icon*[5]. Then, all gathered information is stored in the local SQLite database.

Capturing Minimized Applications. A shortcoming of *active-win* is that minimized applications and applications on separate virtual desktops are not captured. While testing the first versions of the TaskSnap tool, it became apparent that without considering minimized applications, many task-related artifacts are missed. To overcome this limitation, we decided to also take application windows into account that were in focus within a time window of twelve minutes before creating a snapshot. This period was chosen as it corresponds to the mean time knowledge workers spend on a task before switching according to González & Mark [GM04]. To keep track of focused application windows, the *WindowsActivityTracker*[6] module is used. The module queries the currently focused application window every second and fires an event in case the window changes. Our tool then stores the gathered metadata about the window to the local SQLite database. When a snapshot is created, the database is then queried for entries added during the last twelve minutes, and the retrieved entries are merged with the information about currently open applications.

Capturing Files. For some applications, the user might be more interested in the files that were edited with this application than the application itself, for example, when using a text editor. It turned out that from an implementation point of view, it is difficult to get the path of a file that is edited through an application from a window instance, as accessing files is not the window's responsibility. One way to solve this issue is to handle each application individually, as implemented in *Scrapbook* [HL22]. However, this has the disadvantage that the set of supported applications has to be defined in advance. To make our approach work with a wide range of applications, we came up with the following solution: On macOS, the TaskSnap tool makes use of the Unix command *lsof* ("List open files") to get the paths of files accessed by currently running applications. As a wrapper of this command, the npm package *list-open-files*[7] is used. From the API of *active-win*, we get the process IDs of open applications. Then, using *list-open-files*, all files accessed by these processes can be retrieved. A process usually accesses many helper files not directly relevant to the user. Therefore, the list of accessed files has to be filtered. We decided only to keep files with a name that is included in the window title of an application or vice versa. While this approach leads to some false positives, according to our initial testing, relevant files are usually found by this heuristic. The retrieved file paths are then stored in the local database with a link to their associated application.

A different approach had to be implemented for Windows, as the *lsof* command is not provided by this operating system. However, Windows maintains a designated folder that keeps track of the files the user has recently accessed. The TaskSnap tool queries the entries of this

[4]`https://github.com/sindresorhus/active-win`, verified 11.07.2023
[5]`https://github.com/sentialx/extract-file-icon`, verified 11.07.2023
[6]`https://github.com/HASEL-UZH/PA.WindowsActivityTracker`, verified 04.08.2023
[7]`https://github.com/grantila/list-open-files`, verified 11.07.2023

folder to get a list of file paths. As this process is relatively slow, only files accessed during the current day are considered. The resulting list of paths is then matched with applications by considering the applications' window titles as described before. Finally, the paths are persisted in the local database.

Capturing Folders. As software developers use the file system frequently [MBM+17], knowing which folders are opened in the Finder on macOS and in the Explorer on Windows could be valuable to capture the working context of a task. To retrieve open folders, a dedicated PowerShell script is executed on Windows when a snapshot is created. For macOS, AppleScript code was written that provides the same functionality. The extracted folder paths are then stored in the SQLite database and linked to the application entries of the Finder or Explorer.

Opening and Closing Artifacts. Once the paths to artifacts are captured, artifacts are opened and closed using shell commands that are executed by the TaskSnap tool. If files were captured, the tool ensures they will be reopened in the same application as used when creating the snapshot.

4.2.2 Web Page Capturing

To know which web pages are open when creating a snapshot, the TaskSnap tool uses a browser extension initially developed for an approach by Rutishauser et al. [RMHF23]. Whenever an event happens in the browser, like switching a tab or bringing a browser window to focus, metadata about open web pages are sent to the TaskSnap desktop application over a WebSocket connection. The sent metadata contains the web pages' URLs, titles, Favicon URLs, and information about which web page is currently active. In the desktop application, the list of open web pages is cached in memory, and the currently active web page's URL is hashed for privacy reasons and stored in the local database. Keeping the history of active URLs is needed to calculate the relevance of web pages (see Section 4.3). In addition, when creating a snapshot, the latest list of cached web pages is persisted in the database.

The browser extension is also used to automatically close single or multiple web pages again when utilizing the artifact closing functionality of the TaskSnap tool. The extension provides an API that takes a list of URLs as input and closes the corresponding pages if present. Similarly, the extension is used to reopen web pages when restoring a snapshot. As the data received by the extension further contain the browser's name, multiple supported browsers can be used in parallel while using the TaskSnap tool. As of now, *Google Chrome*, *Microsoft Edge*, and *Mozilla Firefox* are supported.

4.2.3 Development Activity Capturing

An extension for Visual Studio Code is used to easily access files opened in the IDE, access Git-related information, and recent code edits. In contrast to the browser extension, it was built specifically for the TaskSnap tool. While the extension is running, it keeps track of recently made code edits. For each edit-event fired by Visual Studio Code, the method that encapsulates the changed code is retrieved by calling the document symbol provider[8]. Then, an object containing details about the edit is added to a queue that caches all edits of the last 60 seconds. When the user creates a snapshot, the extension receives an event from the TaskSnap desktop application. It then aggregates the queue's entries to get the range of recently edited lines within a single method. Further, when the snapshot event is received, the branch and last commit of the currently active

[8] `https://code.visualstudio.com/api/references/commands`, verified 11.07.2023

Git project are fetched by using the Git extension API[9], the workspace name and workspace path are accessed, and a list of currently open files is retrieved. In addition, the uncommitted lines of code accessed through the Git Diff are analyzed to extract all lines that contain a "TODO" keyword, and a flag is set that indicates if uncommitted changes are present in the currently open codebase. All this information is then returned to the TaskSnap desktop application using the WebSocket connection. Similar to the browser extension, the extension for Visual Studio Code provides dedicated endpoints to open and close files.

4.3 Pre-Selection of Artifacts for Curation Step

An automated pre-selection of artifacts was developed to avoid overwhelming users with a long list of items and save them time when curating a snapshot. To pre-select artifacts, the TaskSnap tool calculates the relevance of each element based on a Frequency-Duration-Age (FDA) model proposed by Maalej et al. [MER17]. We also considered using the more well-known degree-of-interest (DOI) model by Kersten & Murphy [KM06] that was successfully implemented in *Mylyn*. However, this model does not incorporate recency determined by the time an interaction with an artifact happened but based on the artifact's relative position in a queue of interaction events. In our case, this position might have remained the same for a long time if the user stayed within a single application, as we do not have access to detailed interaction data for every application type. Therefore, we chose a different approach.

Frequency-Duration-Age (FDA) model. The FDA model is based on the intuitive assumption that "the more and the longer we interact with 'objects' [...], the more relevant they become to the current context. The older our interactions with these objects are, the less important they become for our context" [Maa10]. Hence, to calculate the relevance score of an artifact, the access frequency, access duration, and time since the last access is considered. The more frequently an artifact is accessed and the longer it is in focus, the higher its relevance should be. The more time passes since the artifact was last accessed, the lower should the relevance score be. Following this intuition, Formula 4.1 is used in the TaskSnap tool to calculate relevance. The formula corresponds to the one proposed by Maalej et al. [MER17]. However, their formula is stated in a more generalized manner and allows the inclusion of a richer set of interaction data than we access with the TaskSnap tool, like the execution of saving or copying commands.

For a task t, we define the relevance of an artifact a as:

$$Rel(a,t) = \frac{Freq(a,t) \cdot Dur(a,t)}{Age(a,t)} \tag{4.1}$$

To define the terms used below, we have to introduce the following constants:

- $\boldsymbol{F}$ is the total number of artifact accesses while working on task t.
- $\boldsymbol{D}$ is the total time worked on task t in seconds.
- $\boldsymbol{A}$ is the summed-up elapsed time in seconds since each artifact was last accessed.

These constants are used for normalization purposes. As we do not know when a task t was started in a snapshot-based approach, we always consider interactions with artifacts that happened since the previous snapshot was created. Or, if no previous snapshot exists, the time

[9] `https://github.com/microsoft/vscode/tree/main/extensions/git`, verified 11.07.2023

elapsed since the tool was installed is considered. This heuristic has some shortcomings, as users likely do not create snapshots for all their tasks. An alternative would be to introduce an upper time limit for the task's assumed length. However, we decided against this idea to ensure that also long-lasting tasks are well-supported.

By using the constants defined for normalization, we define:

$$Freq(a,t) = \frac{number\ of\ accesses\ to\ \boldsymbol{a}\ while\ working\ on\ \boldsymbol{t}}{F} \tag{4.2}$$

$$Dur(a,t) = \frac{summed\ up\ interaction\ time\ with\ \boldsymbol{a}\ while\ working\ on\ \boldsymbol{t}}{D} \tag{4.3}$$

$$Age(a,t) = \frac{time\ elapsed\ since\ last\ access\ to\ \boldsymbol{a}}{A} \tag{4.4}$$

Handling of Idle State. Generally, it is assumed that an interaction with an artifact always lasts until another artifact is accessed (as proposed by Maalej et al. [MER17]) or until the computer's lock screen gets active. However, the case has to be handled that the user is not actively using the computer while having an artifact in focus, for example, when discussing something with a co-worker or after going for a coffee break without locking the computer. To address this issue, the TaskSnap tool monitors the time the computer is idle, which is the time neither keystrokes nor mouse movements are detected. As proposed by the authors of the FDA relevance model, a threshold of five minutes was chosen, after which an interaction with an artifact is automatically marked as finished due to idleness.

Threshold for Pre-Selection. When creating a task snapshot, the above-described constants are calculated, and the relevance score of each open artifact is determined based on the stated formulas. Then, a threshold is set that determines what relevance score is needed to pre-select an artifact. In our approach, all artifacts with a relevance score larger than 50% of the artifact with the highest score are pre-selected. This strategy is a basic filtering approach used in data analytics and machine learning [CKD21]. It has to be tested, though, if it fits into our context. One advantage of this adaptive threshold is that at least one artifact is guaranteed to be pre-selected.

4.4 Semi-Automated Summarization of Recent Actions and Intent

To save users some time and prompt reflection, a semi-automated approach to capture the user's mental task context is applied. The two texts regarding recently performed actions ("Now what was I doing?") and the user's intent ("Now what was I about to do?") are automatically pre-filled with a few sentences based on heuristics we defined while developing the approach. Then, the user can adapt or rewrite the texts when curating a snapshot. In the following, we present how the two texts are produced.

4.4.1 Summarization of Recent Actions

The automatically created summary of recently performed actions is generated based on information from the IDE, the recently accessed applications, and open web pages. As we are interested in activities conducted recently, we only consider events that happened within a time window of $T_1 = 12min$ before creating the snapshot. This time window was chosen based on the observation by González & Mark [GM04] that knowledge workers spend on average only twelve minutes on a task before switching. As we do not know when a task has started in a snapshot-based approach like TaskSnap, this time window ensures that an action likely belongs to the present task when the snapshot is created. The summary is structured into two paragraphs: One focusing on recent actions within the IDE that might be especially useful for developers and a second with more general information.

IDE-Specific Summary. The summary starts with information about recent development activity. From the IDE, the last performed code edit is included in the text if it was made within the time window T_1. If just a single line was changed, the content of this line is added to the summary alongside the line number, the name of the edited method, and the file name. When multiple lines of the same method are changed within a short period ($T_2 = 60s$) and hence likely belong to the same change, the edited line range is added. However, the lines' content is omitted to not make the text too lengthy. Besides the information about recently made edits, the last commit message is appended to the summary if the commit happened within the time window T_1. Further, a sentence is added if uncommitted changes are present in the active Git project.

General Summary. The second paragraph provides information about the user's recently used applications. Again making use of the time window $T_1 = 12min$, the application that was in focus for the longest during this period is determined and added to the summary. We also considered showing the last accessed application but decided against it as users switch very frequently between artifacts [MBM$^+$17]. Therefore, the last accessed application might be task unrelated, for example, when the user quickly checked the calendar or mail client before creating a snapshot. In addition, the most recent activity in the browser is contained in the snapshot's summary by adding the title of the last accessed web page, given that it was accessed within the time window T_1. Knowing what information was looked up on the internet last could be a valuable cue to resume a task, as software developers heavily use the browser [MBM$^+$17].

4.4.2 Capturing Task Intent

The text regarding the user's intent on how a captured task should be continued is also automatically pre-filled. Automatically sensing the user's intent is challenging or even impossible. However, we can leverage that many developers actively leave TODO comments in source code to know where to continue a task and to get reminded about open issues [PD10, RTH17]. As we can access the codebase used while working on a task through the IDE extension, we can use written TODO statements to pre-fill the text containing the user's intent. When the user creates a snapshot, the TaskSnap tool extracts all lines of code containing TODO comments from the uncommitted source code of the active Git project. Only uncommitted code is considered to focus on open issues related to the current task and not issues generally present in the codebase. The collected lines of code are then concatenated to a single text, analogously to the summary of recently performed actions. Including recently written TODOs into the TaskSnap tool has a further advantage for the user: Even though modern IDEs typically provide dedicated views to inspect all TODO comments found in a codebase, these issues are not actively brought to the user's attention again and hence can get forgotten or are not considered at the right time [PR12]. By presenting the user’s intent when restoring a snapshot, we ensure that the comments are brought to attention when continuing to work on a task.

Chapter 5

Pilot Study Method

We conducted an explorative pilot study with six developers in the field to answer our research questions. Our intention behind the performed study was, on the one hand, to get insights into existing strategies of developers regarding how they capture task-related information and how they manage task switching and resumption. On the other hand, we wanted to find out how participants used the TaskSnap tool and what impact it had on their work, especially concerning the number of artifacts they kept open. The pilot study spanned approximately ten workdays and contained two main study phases: A baseline phase in which the tool was running in the background to collect data, and an intervention phase, in which the complete set of features offered by the TaskSnap tool was accessible. This chapter presents the study procedure in more detail (Section 5.1) and describes the study participants (Section 5.2). Then, we discuss the collected data types (Section 5.3) before presenting how data was analyzed (Section 5.4).

5.1 Procedure

The pilot study with six participants spanned over ten workdays. After recruiting participants, the study started with a survey to get insights about existing task context capturing strategies participants apply. Then, after an onboarding meeting, a five-day baseline phase was performed in which the TaskSnap tool was mainly running in the background to collect information about the number of artifacts participants typically keep open. Next, the study phase was changed together with the participants, and a five-day intervention phase started, in which all features of the TaskSnap tool were available. Finally, an interview was held to get qualitative feedback about how the participants used the tool and if they see value in it. In the following paragraphs, the involved steps of the study are described in more detail. An overview of the procedure is outlined in Figure 5.1.

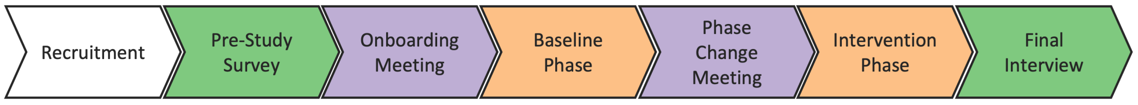


Figure 5.1: Study procedure.

Recruitment. Participants were recruited from our personal and professional networks. We were seeking software developers and data scientists that utilized the IDE Visual Studio Code regularly and worked on either macOS or Windows, as only these platforms are currently supported by the TaskSnap tool. An email containing the consent form and link to the pre-study survey was then sent to people who were interested in participating and met the criteria. In addition, an appointment for the onboarding meeting was scheduled over email.

Pre-Study Survey. The study started with a pre-study online survey. With this survey, we aimed to gather insights on the existing task context capturing and resumption routines of participants, determine whether they already employ strategies to facilitate these activities, and evaluate if there is a need for an approach like TaskSnap. Further, participants were asked to provide basic demographic information. The questions asked in the pre-study survey can be found in the Appendix A.1.

Onboarding Meeting. In an onboarding meeting held online, the goals and procedure of the study were explained, and participants had the opportunity to ask remaining questions regarding the study. If there were no more questions, participants were asked to install the TaskSnap desktop application and the two required extensions for the browser and Visual Studio Code on their work computer. One of the researchers then ensured that the tool was working correctly on the participant's machine. In addition, a meeting to change the study phase was scheduled.

Baseline Phase. Participants were asked to continue working as usual while the TaskSnap tool was running in the background for five workdays, meaning that users were not yet able to create task context snapshots in this study phase. In the evenings, participants were prompted to fill out a short end-of-day questionnaire to reflect on the impact of task switches made during that day and how they captured information about the interrupted tasks. The questions of the end-of-day questionnaire are listed in Appendix A.2. This study phase was further used to collect information about the number of artifacts participants typically keep open and their behavior regarding closing artifacts.

Meeting to Change Study Phase. After five workdays, a meeting was held in which one of the researchers performed the necessary steps to change the study phase in the TaskSnap tool together with the participants. In addition, the features of the TaskSnap tool were explained and demonstrated. Particularly, participants were instructed on how they can create, curate and restore snapshots. Finally, they created their first snapshot to see if the tool worked correctly on their machine.

Intervention Phase. For the next five workdays, while the full functionality of the TaskSnap tool was enabled, participants were asked to continue working as usual while trying to incorporate the tool into their workflows. This means participants could create snapshots when switching the task and restore them later. At the end of each workday, participants were prompted to fill out a short questionnaire. On the one hand, this questionnaire contained the same questions regarding the impact of task switches made during that day as asked in the baseline phase. On the other hand, questions about snapshots created throughout the day were asked. Moreover, after restoring a snapshot, the user was prompted to fill out a separate questionnaire, in which they were asked how they re-identified the restored snapshot in the *Snapshot Gallery*. The questions of this questionnaire can be found in Appendix A.3. In this study phase, in addition to the number of open artifacts, pseudonymized interaction data with the TaskSnap tool was collected.

Final Interview. After the main part of the study, a 30-minute semi-structured online interview was conducted with each participant. Participants were asked about their experience with the TaskSnap tool, the scenarios in which they created snapshots, and whether the tool supported restoring the working and mental context of a task. A list of the used guiding interview questions can be found in Appendix A.4. In the end, we asked participants to export, review, and send the locally stored data to one of the researchers.

5.2 Participants

Six participants (five male, one female) were recruited using the researchers' personal and professional networks. The age of participants was between 23 and 27 (M=25.17, SD=1.21). All participants worked in software development, data science, or a related field, and they reported having between one and three years of professional software development experience (M=1.67, SD=0.94). Two stated to be (computer science) students, two were software engineers, one was a mechanical engineer, and one participant described their job title as "PhD student/Research assistant". Henceforth, we will continue to refer to them as "developers". All participants reported to be individual contributors. Asked about their operating system, four answered using Windows and two macOS during the study.

The availability and work schedule of participants introduced minor deviations to the study's planned time span of ten workdays. Participants used the TaskSnap tool for four to six days in the baseline phase (M=4.83, SD=0.69) and between four and five and a half days in the intervention phase (M=4.92, SD=0.45).

5.3 Data Collection

During the pilot study, quantitative and qualitative data were collected. Among others, the collected data contained created snapshots, a data stream capturing the number of open artifacts, experience sampling results, and survey and interview answers. All data captured by the TaskSnap tool was stored in a local database, and participants could inspect, review, and voluntarily upload it at the end of the study. All participants decided to share their data with the researchers. The collected quantitative and qualitative data are presented in more detail in the following.

5.3.1 Quantitative Data Collection

For the TaskSnap tool to work and to get insights into the usage and impact of the approach, quantitative data was collected. The data types presented subsequently were gathered by the tool without further user interaction.

Context Snapshot Data. The TaskSnap tool stored the context snapshots created by the user in a local database. A context snapshot contained the name, path, and window title of all applications that were open at the time the snapshot was created and their associated files or folders. Further, from the IDE, open code files, Git-related data, and information about the most recent code change belonged to a context snapshot. The complete set of persisted IDE data is described in Section 4.2.3. Moreover, a list of open web pages and a textual summary of recently performed actions and the user's intent was part of a snapshot.

Continuous Data Stream to Determine the Relevance of Artifacts. As a data source for the automatic pre-selection (see Section 4.3), the active Visual Studio Code file, web page, and application window were continuously stored to the local database. Based on this history of active artifacts, relevance could then be calculated. URLs coming from the browser were always hashed to preserve privacy. Besides using this continuous data stream to calculate the relevance of artifacts, it was accessed to add currently minimized applications to a task snapshot (see Section 4.2.1, where we describe how open artifacts were retrieved).

Continuous Data Stream for Quantitative Analysis. To be able to analyze the impact of the developed approach on workspace clutter, the TaskSnap tool ran an algorithm that collects the number of open artifacts at an interval of $T = 60s$. This frequency was chosen to have enough data points to observe short-term changes, for example, after a snapshot was created, while not significantly increasing the performance impact of the application by sampling too frequently. Each time the algorithm was executed, the names of running applications that had open windows were stored. Minimized applications were considered as well. In addition, the list of open web pages was persisted by storing hashed URLs. The same was done for files open in the IDE and folders in the file manager.

TaskSnap Interaction Data. Interaction data with the application was collected to analyze how participants used the TaskSnap tool. Among others, it was logged when a snapshot was created, postponed, inspected, or resumed. It was also stored from where these actions were triggered. For example, if a snapshot was created by pressing the physical button, entering the shortcut, or clicking the icon in the tray or taskbar.

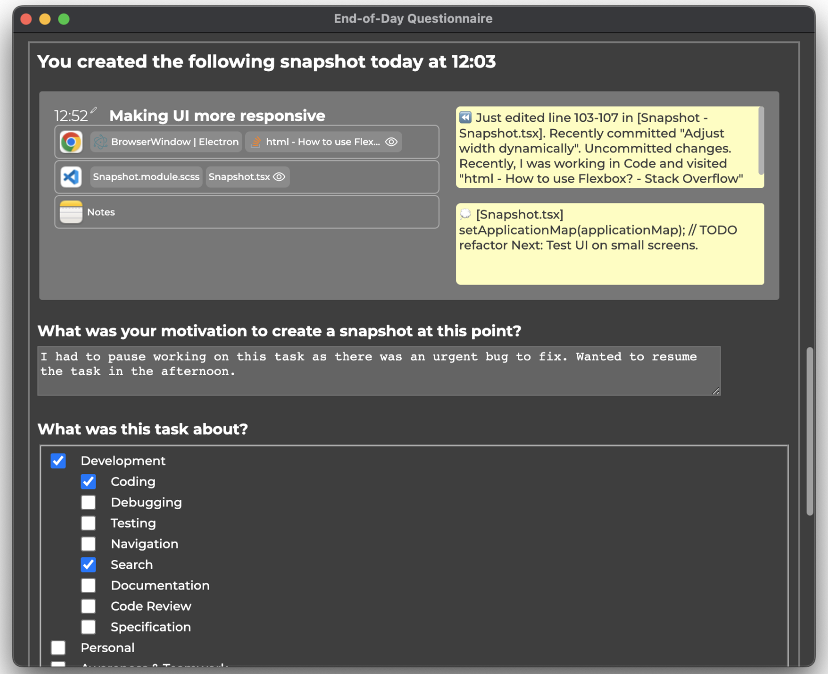
5.3.2 Qualitative Data Collection

To get qualitative insights into existing task context-capturing strategies, occurring task switches, and the usage and impact of the approach, we collected answers from a pre-study survey and final interview. However, by only collecting these answers upfront or at the end of the study period, insightful details could be missed. Therefore, we incorporated experience sampling into our study design to gather more in-the-moment feedback. The following qualitative data types were collected during the pilot study.

Experience Sampling – End-Of-Day Questionnaires. At the end of each workday, the user was prompted to answer questions regarding task switches performed during the day. Some questions were only present in one of the two phases; others were included in both study phases to compare the answers and observe a possible impact of the TaskSnap tool. The exact structure of the end-of-day questionnaires and all answer options are listed in Appendix A.2.

In the baseline phase, the user was asked how they kept track of important task-related information and if there was a situation where a tool that supports task context capturing and resumption would have been helpful. With these questions, we wanted to learn more about existing strategies of developers and if an approach like ours could support them. In the intervention phase, questions regarding created snapshots were asked. If available, two snapshots created during the day were presented, and for each snapshot, the user could provide insights about their motivation to capture a task's context and what the task was about. In addition, the user was asked if there were additional tasks not captured by a snapshot. With these questions, we wanted to learn about different use cases of our approach and situations in which it might not fit well. Figure 5.2 shows a part of the end-of-day questionnaire that was used during the intervention phase.

In both study phases, the user was asked about instant task switches that happened during the day. With this question, we wanted to learn more about the possibility of supporting this kind of task switch with our approach. Further, the user was asked how much effort it was to restore the working and mental task context during the most recent task switch and how cluttered they perceived their workspace. With these questions, we wanted to analyze the potential impact of the approach regarding productivity and workspace clutter.


End-of-Day Questionnaire

You created the following snapshot today at 12:03

12:52 Making UI more responsive

BrowserWindow | Electron   html - How to use Flex...

Snapshot.module.scss   Snapshot.tsx

Notes

Just edited line 103-107 in [Snapshot - Snapshot.tsx]. Recently committed "Adjust width dynamically". Uncommitted changes. Recently, I was working in Code and visited "html - How to use Flexbox? - Stack Overflow"

[Snapshot.tsx] setApplicationMap(applicationMap); // TODO refactor Next: Test UI on small screens.

What was your motivation to create a snapshot at this point?

```
I had to pause working on this task as there was an urgent bug to fix. Wanted to resume
the task in the afternoon.
```

What was this task about?

- [x] Development
  - [x] Coding
  - [ ] Debugging
  - [ ] Testing
  - [ ] Navigation
  - [x] Search
  - [ ] Documentation
  - [ ] Code Review
  - [ ] Specification
- [ ] Personal

Figure 5.2: End-of-day questionnaire during intervention phase with preview of a snapshot created that day.

Experience Sampling – Task Resumption Questionnaire. To get insights into how snapshots must be visualized for easy re-identification, the user was prompted to fill out a short questionnaire each time a snapshot was restored from the *Snapshot Gallery*. In this questionnaire, the user was asked how they re-identified the just restored snapshot and what additional visual cues could be helpful. The asked questions and provided answering options are listed in Appendix A.3.

Survey & Interview Answers. The conducted pre-study survey was used to gain qualitative insights into existing task context capturing, task switching, and resumption strategies of participants. Further, the answers of the final interview were considered to better understand how our proposed approach was used, how it fitted into existing workflows, and what its impact was.

5.4 Data Analysis

Interviews. The final interviews were held online using Microsoft Teams. With the consent of participants, the interviews were held in English, recorded, and later transcribed using the transcription feature of Microsoft Word. All participants agreed to this process. Using the transcripts, the participants' answers were analyzed using a coding approach based on the thematic analysis framework [BC12]. For each statement of participants, codes were iteratively defined and assigned that summarize the statement's content. Then, based on the assigned codes, connecting different answers and extracting the main themes was possible. These qualitative findings gathered from the interviews later helped us to answer our research questions. Henceforth, the abbreviations P_1 to P_6 are used to refer to the six participants.

Processing Collected Quantitative Data. The collected quantitative data was statistically analyzed using Python. As a pre-processing step, data cleaning was performed, for example, to discard snapshots created during the onboarding process or other snapshots that were clearly created for testing purposes.

While designing the pilot study, we decided that deleting a snapshot in the TaskSnap tool removes all associated data from the local database to preserve the user's privacy. However, by doing so, we know little about the reason why a user deleted a snapshot. Possibly, the snapshot was no longer needed; it was created for testing purposes, or by accident. Overall, 44 delete events were logged by the TaskSnap tool. To consider only the snapshots in the analysis that provided value at some point, we included only those snapshots that were still present when the user uploaded their local database at the end of the study.

Analyzing Likert Scale Answers. As described in the data collection section (Section 5.3.2), some questions were part of the end-of-day questionnaire in both study phases to be able to observe a potential impact of the approach. To answer these questions, the user was given 5-point Likert scales that were then mapped to the numbers one to five in the pre-processing step to compute mean answers. If the mean responses showed a trend, we further applied Welch's t-tests to statistically analyze if there was a difference between the two study phases.

Chapter 6

Results

This chapter presents the results of the performed explorative pilot study. The chapter starts with findings about the users' existing behavior and strategies regarding task context capturing and resuming before using the TaskSnap tool (Section 6.1). Then, we present how participants integrated the task context snapshot approach into their workflows (Section 6.2), before providing results about the general impact of the approach (Section 6.3). Finally, we focus on the approach's impact on workspace clutter in Section 6.4.

6.1 Developers' Existing Strategies for Task Context Capturing & Resumption

The pre-study survey allowed us to get insights into participants' existing task-switching behavior and their strategies to capture working and mental context to make task resumption easier. Participants of the pilot study only work on a few main tasks during the day and differ in the strategies they employ for task context capturing and resumption. However, they all have in common that they use separate approaches to capture the state of their working and mental context and apply note-taking using a wide variety of tools. The following paragraphs discuss these strategies, differences, and commonalities in more detail and try to answer RQ1.

Developers Focus on a Few Tasks a Day. According to the pre-study survey, developers participating in the pilot study are typically not working on more than three main tasks during a day. Two out of six participants even stated that they generally work on just one main task. This finding was supported by the data collected through the end-of-day questionnaires. There, each participant stated on at least one workday not having worked on multiple tasks. Overall, this was the case in 15 out of 45 end-of-day responses. In the final interview, P_2 gave more context by reporting *"I basically did one task for the whole week with only like small side tasks"*. It must be noted, though, that all participants were part-time developers, and hence their workflows might not be representative of full-time professionals.

Capturing & Resuming Working Context. Almost all participants actively capture their working context using technical solutions to make task resumption easier. They manually capture their most important artifacts by referencing them in the current context (e.g., adding URLs to a code file) or using a separate system (e.g., writing URLs to a local text file or online document). Not only text but also screenshots are used to document the current progress (mentioned by one out of six participants). Another common approach three out of six participants mentioned is to keep the current working context open when switching tasks, at least when the started task will likely not last long. Further applied strategies are using a dedicated virtual desktop for each task (two out of six), using different applications, browser windows, or tabs for each task, or all relevant information is kept *"only in my head"* (P_3).

Capturing & Resuming Mental Context. All participants apply note-taking to capture their mental task context while working on a task or right before task-switching. The tools in which notes are stored differ. Some write comments directly into source code with TODO statements (mentioned once), others use online or offline documents (mentioned by two out of six users), or use dedicated note-taking applications (three out of six). Participants also use project or task management tools to capture information (two out of six), for example, by writing comments to a GitHub issue or keeping track of ideas and goals for follow-up tasks in the application *Microsoft To-Do* or by creating subtasks in *Asana*. Another participant reported always trying to commit their last changes before switching to have less mental context to restore. Two out of six participants further stated that they keep track of the mental context in their heads. One reported not having a strategy, which likely also results in mentally storing important information.

Finding 1: Developers currently manage working and mental task context using separate strategies. Note-taking is applied by all participants to capture the task's context.

6.2 Integration of a Task Context Snapshot Tool Into Existing Workflows

The developed approach was well-perceived by participants and fitted seamlessly into their existing workflows by being very unobtrusive. Overall, the six participants created 42 snapshots throughout the study, on average, 1.24 per day and user (SD=1.33). Snapshots were mainly created at the end of the workday and restored within 24 hours, but also other possible use cases were identified. In the following paragraphs, more detailed information about how the approach was perceived and integrated into the participants' work is presented to answer RQ2a.

Approach Was Perceived as Unobtrusive. Participants especially liked the approach's unobtrusiveness which facilitated a seamless integration into numerous existing workflows. According to P_1: *"It fits perfectly in a sense because it never prompted me. It never wrote any notifications that would take me out of my usual workflow, and it was only when I explicitly wanted to create a snapshot, I could click the shortcut, and then I could create the snapshot"*. P_2 added: *"That's the nice thing. If I don't need it, I can just leave it be. I don't have to do anything, but if I want to use it, it's right there all the time"*. All other participants reported similar impressions. Two out of six users mentioned that a minor drawback of the unobtrusiveness of the approach is that they sometimes forgot to create snapshots in the beginning of the study when switching tasks.

Finding 2: The approach's unobtrusiveness was pointed out positively by all participants and made it fit into existing workflows. However, due to its unobtrusiveness, two out of six participants sometimes forgot to create snapshots.

6.2.1 Use Cases

While participants used the approach to capture and resume tasks for different use cases, the most common was using it at the end of the workday. Snapshots were seldom created at instant task switches. The different identified use cases are presented in the following paragraphs.

Approach Frequently Used at End of Day. Half of the participants reported having mainly used the approach at the end of the workday. The timestamps when snapshots were created support these statements. P_3 described the reason for taking snapshots at the end of the day as follows: *"I think it's really nice like at the end of the day [...] and actually it makes you think: What will I actually need tomorrow to solve the task? What's going to be relevant?"*. Another participant used the approach to detach from work by closing all work-related artifacts. Using a digital aid to detach from work can have positive effects on productivity, as analyzed more thoroughly by related work [WKM+18] (presented in Section 2.3). In contrast, one user preferred not to create snapshots at the end of the day to be able to review their work. *"I feel like it's kind of satisfying at the end of the day to be able to close everything individually and like sort of... Let it run past you, what you've done in the day and what you used"* (P_2).

Snapshots Not Created During Instant Task Switches. The approach was seldom used for instant task switches, according to the participants' answers in the final interview. They further reflected on this use case and had mixed opinions. One participant was confident that the approach could be very valuable to support instant task switches. Another participant pointed towards a potential issue: *"Sometimes when someone approached me... I could have just taken a snapshot. But it's always difficult to know how long this task switch is going to be, so if it's very short, or if it's just one question, I think I'll be fine without the snapshot, but this is not always clear beforehand"* (P_1). A third participant stated that they preferred creating snapshots when having enough time to write down thorough notes. For quicker task switches, they just wrote down keywords, which made it harder to resume the task later. Still, this user thinks the approach supports this use case as well.

Many other use cases exist. Besides using the approach when having sufficient time to switch tasks, developers used it for different use cases and could envision even more situations in which it might be supportive. In the final interview, P_5 explained that they might have used the TaskSnap tool differently than intended. Instead of creating snapshots of individual tasks, the user created snapshots for all their projects. Another participant mentioned that the tool might be valuable in preparing for meetings. *"If you have a meeting early in the morning, you can make a snapshot preparing it the evening before [...], and then restore it before the meeting"* (P_3). Another participant also brainstormed about the possibility of using a snapshot-based approach as a team right after meetings to summarize the main findings and persist things to discuss next.

Finding 3: Participants used the approach when having enough time to switch tasks or at the end of the workday to reflect on the day and detach from work. In contrast, the approach was seldom used for instant task switches.

6.2.2 Snapshot Creation, Curation & Resumption

Participants created between two and 16 snapshots, almost always curated them immediately, and provided thorough descriptions of their intent. The automatic pre-selection of artifacts was perceived as accurate but too conservative and users generally restored snapshots within 24 hours after their creation. Further, when restoring a snapshot, the snapshot visualization was perceived as useful but could be improved by supporting grouping. The snapshot creation, curation, and resumption workflow of participants is described in more detail in the following.

Developers Created One Snapshot per Day on Average. All participants used the tool by creating and not deleting between two and 16 snapshots, resulting in 42 snapshots. On average, 1.24 snapshots were created per day and user (SD=1.33). Table 6.1 displays how many snapshots were created by each participant. Half of the snapshots were triggered using the shortcut (20 out of 42), the button was used 13 times, and the respective option in the application's menu nine times. Even though the button was not used the most to create snapshots, several users pointed out that they especially liked this feature. *"I like the button. That was fun"* (P_3). However, users reported that in many situations, the other options to create snapshots are more practical, as the button blocks a USB port (mentioned by three out of six) and is cumbersome to carry along or can get forgotten when switching locations (two out of six). As the currently supported button uses touch input, two out of six participants further pointed out that the button should be more haptic.

Mostly Immediate Snapshot Curation. According to the answers of the final interview, participants had enough time when switching tasks and hence performed the curation step immediately. The collected interaction data support this finding. On average, the curation was saved for the first time 1min 25s after creation (SD=51s). P_6 pointed out that they avoided postponing the curation step to reduce clutter immediately. *"The whole point of doing snapshots is to keep a clean workspace, and if I wouldn't do it immediately before switching to another task, I would have the same situation as always"*. The collected interaction data confirm this finding, as snapshot curation was only postponed once.

Texts That Support Mental Context Were Modified. Both texts that support the user in restoring their mental context were adapted and extended by participants. Regarding the text for summarizing recent actions, for 40% of the snapshots, users adapted the pre-filled text (4 out of 42) or created it from scratch (13 out of 42). When adapting the text, usually the structure was kept, but the content changed: *"I was currently working in IntelliJ due to a java project, which had a specific configuration for IntelliJ"*, *"Recently, I was working on the general discussion part"*. Custom-written texts are generally short and do not contain a lot of details. Examples are *"Fixed visualization"* or *"Implementation of sensitivity analysis for linear MPC"*. A custom text covering the user's intent was created in 23 out of 42 snapshots. The written texts are more detailed than the summaries of previous actions. For example, users wrote *"Further investigate querying different collections and aggregation techniques to create an overview of the API requests split by plugins"* or *"The next steps would be: 1. Get an overview over participant data. 2. Write Methods, Introduction, Related Work"*.

Few Artifacts Belong To A Task. On average, 4.10 artifacts (SD=4.04) were selected in the curation step. The number of selected artifacts ranged from zero to 25 and the persisted artifacts are diverse. The browser was part of a snapshot in 81% of the cases (34 out of 42), and if present, on average, 3.54 web pages were selected (SD=2.55). Table 6.1 shows the mean number of selected artifacts and web pages for each participant. Noteworthy, only three IDE files were persisted and selected in snapshots. Possibly because several participants reported that they did not do a lot of programming work during the study or could use the IDE Visual Studio Code less frequently than expected.

Automatic Artifact Pre-Selection Was Too Conservative. The automatic pre-selection of artifacts received mixed opinions and generally selected fewer artifacts than were relevant to the task. According to the final interview, three out of six participants perceived the pre-selection as accurate, two users experienced good and bad pre-selections, and one is unsure and maybe would even prefer having the pre-selection disabled. Two out of six participants mentioned that in tendency, too few artifacts were selected. However, P_3 pointed out that this is not necessarily

Participant	No. of Snapshots	Mean No. of Artifacts	Mean No. of Web Pages
P_1	6	8.3 (SD=8.8)	8.0 (SD=3.8)
P_2	2	3.5 (SD=0.7)	3.0 (SD=0.0)
P_3	16	3.5 (SD=2.5)	2.9 (SD=1.4)
P_4	4	2.3 (SD=1.5)	1.5 (SD=0.7)
P_5	5	3.4 (SD=1.3)	3.2 (SD=1.1)
P_6	9	3.7 (SD=2.3)	2.7 (SD=1.6)

Table 6.1: Number of snapshots, mean number of artifacts, and mean number of web pages per snapshot for each participant.

bad: *"If you want to have a minimalist approach, it's probably better to have fewer pre-selected"*. The snapshot data stored by the TaskSnap tool supports this impression. Of the 193 artifacts selected by participants, 38 were pre-selected, which leads to a low recall value of 0.20. The precision of the approach is better with a score of 0.73, meaning that the majority of artifacts selected by the model were relevant for a task. A confusion matrix is provided in Figure 6.1.

While developing the approach, we defined the heuristic that all artifacts get pre-selected that have a relevance score larger than 50% of the artifact with the highest score. Lowering this percentage still leads to many false negatives, as many calculated scores lie within a small range just above zero. This is caused by the fact that the normalization constants can be disproportionately large, as all interactions since the previously created snapshot are considered. A future iteration of the pre-selection algorithm could improve this by capping the number of considered interaction events. Further, it could be tested if adding different weights to the three components of the frequency-duration-age model improves the accuracy of the pre-selection. As two out of six participants reported that interaction recency affected the pre-selection too much, a starting point could be to add a small weight to the age component of the model.

Finding 4: Snapshots were almost always curated immediately, and participants provided thorough descriptions regarding their intent. The automatic pre-selection of artifacts was perceived as accurate but is a bit too conservative.

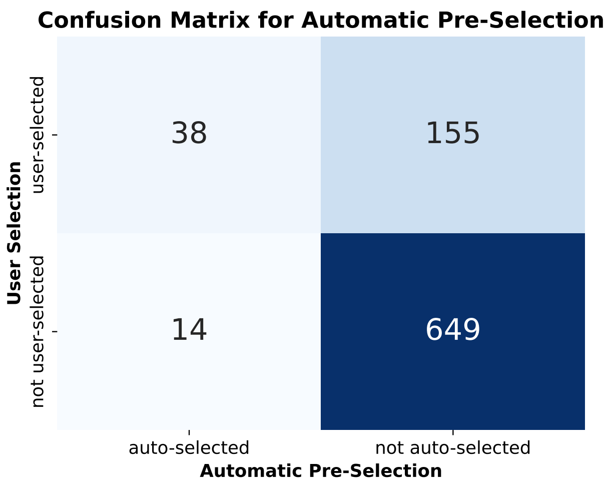


Figure 6.1: Confusion matrix comparing the artifacts selected by the user with the automatic pre-selection.

Snapshots Were Restored Quickly. Most snapshots were restored within 24 hours, and some were even restored multiple times. Based on the user interaction data, all participants restored snapshots, resulting in a total of 33 restore events. Snapshots were restored mainly from the *Snapshot Gallery* (26 out of 33 times). The option to restore from the tray or taskbar was less frequently used (5 out of 33), and users seldom restored from the curation view (2 out of 33). 57% of the snapshots (24 out of 42) were restored at least once; single snapshots were also restored up to three times. The elapsed time between creating a snapshot and resuming it differed. 82% of the snapshots (27 out of 33) were restored within 24 hours, but the maximum elapsed time is over six days. The mean elapsed time was the longest for P_5 (M=72.6hr, SD=83.8hr), likely as this user used the tool differently than initially intended (described in Section 6.2.1). The distribution of elapsed time between snapshot creation and resumption is displayed in Figure 6.2.

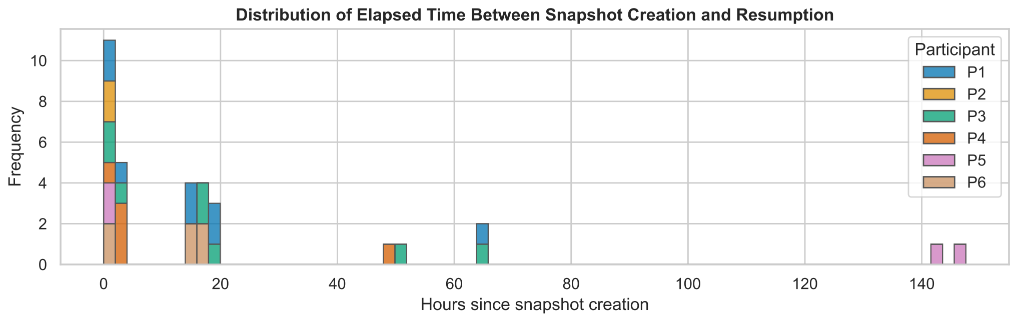


Figure 6.2: Distribution of elapsed time between snapshot creation and resumption.

Re-Identifying Snapshots Is Easy. All participants had no issues re-identifying a snapshot in the *Snapshot Gallery* and found snapshots mainly based on the title, according to the final interview. This impression is also supported by the results of the experience sampling: In 95% of the cases (22 out of 23 times), participants reported that snapshot re-identification was very easy or easy. According to the questionnaire, the title was most helpful, followed by the snapshot's position in the list. However, it has to be considered that participants generally restored the latest available snapshot (20 out of 27 times).

Generally, participants reported that the visualization approach in the *Snapshot Gallery* works well and that all necessary information is presented. According to a participant, even too much information is displayed, which makes the gallery a bit cluttered. Further, three out of six participants would like to have the option to tag or group existing snapshots to organize them better.

Finding 5: 57% of all snapshots were restored, 82% of them within 24 hours. Participants reported having mainly restored them based on the title, but as they usually resumed the latest available snapshot, the position of the snapshot in the gallery might have also played an important role.

6.3 General Impact of a Task Context Snapshot Approach

Developers reported that the approach can reduce mental clutter and supports restoring the working and mental task context. While these findings are based on qualitative insights, the quantitative data does not show significant effects, likely because only little data is available, and participants were only asked about a single task switch for each workday. The following paragraphs give more insights into these findings and try to answer RQ2b.

Reducing Mental Clutter. The proposed approach can have a positive impact on the well-being of users. Two out of six participants mentioned that closing artifacts not only cleans up the user's workspace but can also help reduce mental clutter. *"I think this closing everything from the previous task is very good because it kind of also in your mind, in your head, it closes everything that has to do with it, and you have new headspace for the new task"* (P_5). The approach can further alleviate the fear of closing artifacts and worrying about forgetting them when resuming a task (mentioned by two out of six participants). *"I don't have to actively think about: Did I store all the links that I need in some file? [...] Did I forget something? I don't need to think about that. I can just click it and it opens it"* (P_5). Additionally, two participants see a main advantage of using the approach in increased awareness. *"I think like the biggest advantage is simply that it makes you reflect on the tasks you are doing and what's important to them"* (P_3).

Faster Task Resumption. Contrary results were found regarding the impact of the approach on task resumption, as qualitatively, participants reported positive effects, but the performed experience sampling does not show this effect. All participants mentioned in the interview that they saved some time by using the approach and that it helped them to refind information. Two out of six participants pointed out that the approach increased their productivity by being able to reopen several artifacts with a single click. The same was mentioned regarding the option to automatically close artifacts: *"That was really helpful and saved me a lot of time, especially when I had a lot of windows open, I could just click the button and everything was closed"* (P_4). One participant mentioned that the approach helped them further to get focused on a new task more quickly and that they gained some time by having fewer artifacts open. *"So it helped me reduce the time I had to switch between the tabs to find the [...] ones that I were working on"* (P_1).

In the end-of-day questionnaire of both study phases, we asked participants how much time they needed to restore their working and mental task context during the most recent task switch. Participants were given a 5-point Likert scale ranging from "almost no time" to "much time" to answer the question. Based on the available data, the approach did not impact the time needed to restore the working task context (baseline: M=1.75, SD=0.58, intervention: M=1.79, SD=0.90). However, there was a non-significant decrease in the time needed to restore the mental task context. In the baseline phase, a mean value of 2.63 (SD=1.15) was reported, and in the intervention phase, a value of 2.07 (SD=0.62). Welch's t-test results in a t-value of 1.67, a p-value of 0.11, with 23.55 degrees of freedom. Based on a significance level of 5% (α = 0.05), the result of the t-test is not significant. Hence, the null hypothesis that the two means are equal cannot be rejected. However, it has to be considered that users were only asked about the impact of a single task switch and that relatively few data points were available to run the analysis (baseline: 16 reports, intervention: 14 reports).

Finding 6: A task context snapshot tool can support users' well-being by reducing mental clutter and the fear of forgetting artifacts. In addition, it can increase perceived productivity by helping users to refind and reopen artifacts quickly.

Working and Mental Task Context Are Important. Participants perceived the approach as helpful to restore working and mental task context but differed in their perception of which of the two aspects is more valuable to them. The performed tasks can also influence their preference. These findings support our initial design decision and novelty of capturing and resuming both task contexts. This is essential as users' individual preferences and differences in tasks can only be addressed by providing support for both.

Three out of six participants especially liked that the approach supports restoring the working context. Participant P_2 believes that for big tasks *"you probably also know the next day what you were working on"* and hence support to restore the working context is more important. One participant reported that they usually have no issues with getting back into individual tasks, but their cluttered desktop and browser bother them. Another participant stated that supporting the working context by showing task-related artifacts already helps to restore the mental model.

In contrast, two out of six users see more value in supporting the mental context. *"The mental state is more difficult to recover than the artifacts that you should use"* (P_1). Another participant pointed out that they did not have many artifacts open, so writing down what they did and what they wanted to do next was more important to resume a task later efficiently. Further, the observation that for two snapshots, users did not select any artifacts in the curation step and just filled in the texts to support the mental context underlines the value of supporting this context type.

Some participants pointed out that the value of the two approaches depends on the context, for instance, the time that elapses between snapshot creation and resumption (mentioned by two out of six participants).*"If I thought I would go back in like two hours, I didn't really bother writing down a lot"* (P_6). The task size and difficulty can also play a role. *"For smaller tasks, the working context is enough. For larger ones, the mental context might be more useful"* (P_6).

6.4 Impact on Workspace Clutter

Developers differ in the number of artifacts they prefer to keep open while working and have different reasons to do so. Still, all participants reported a positive impact of the proposed approach on their ability to reduce workspace clutter. While these findings are based on qualitative insights from the interviews, the number of open artifacts did not decrease for all participants between the two study phases. In the following, we present insights about the participants' typical workspace decluttering behavior and how the task context snapshot approach impacted it. This contributes to answering RQ2b.

6.4.1 Decluttering Behavior of Participants

The number of artifacts participants leave open and the reason for not closing them differ. However, the timing when closing artifacts is similar, and the same artifacts are considered most valuable by participants. The following paragraphs provide insights into these commonalities and differences.

Participants Close Artifacts When Task Is Finished. When switching a task, the timing when participants close task-related artifacts heavily depends on whether they have completed the task or intend to resume it later. In the pre-study survey, three out of six participants mentioned not closing artifacts at all until the task is finished when it is likely that the task will be resumed. Two out of six reported closing artifacts after a long time, and one after some time. The picture is different when a task is completed entirely. Then, three out of six participants close artifacts immediately, two reported "soon" and one "moderately soon".

Number of Open Artifacts Differs. Users differ in how many artifacts they leave open in the background while working. This becomes especially apparent when considering the mean number of open web pages collected during the baseline phase (see Table 6.2). P_2 and P_4 keep fewer than five pages open on average, while P_1, P_3, and P_5 have more than 19 open web pages. This finding corresponds to the results of previous work [MLNL23]. Also regarding the number of open applications, differences exist while not being that pronounced. On average, participants use 5.2 to 9.8 applications simultaneously, which also aligns with previous findings [PRM+20]. Again, P_2 has fewer applications open than other participants, indicating that some users have generally more minimalist approaches than others. These differences were pointed out by previous work as well [VJM18]. The number of open file system folders is within a smaller range (between 0.5 and 2.0 folders on average). The number of open IDE files was omitted from the analysis, as some participants used the IDE Visual Studio Code only infrequently during the study. According to the pre-study survey, the majority of participants perceive their workspace as somewhat cluttered (P_1, P_3, P_4, P_5), and only one describes their workspace as cleaned up (P_2).

Participant	Web Pages	Applications	File System Folders
P_1	19.4 (SD=12.4)	7.3 (SD=2.8)	0.8 (SD=1.0)
P_2	4.8 (SD=3.0)	5.2 (SD=1.5)	0.5 (SD=0.5)
P_3	19.7 (SD=9.9)	9.8 (SD=0.6)	1.0 (SD=0.0)
P_4	3.0 (SD=3.8)	7.5 (SD=2.0)	1.0 (SD=0.5)
P_5	26.8 (SD=10.0)	9.1 (SD=1.4)	2.0 (SD=0.1)
P_6	8.7 (SD=4.3)	7.5 (SD=3.0)	0.8 (SD=1.3)

Table 6.2: Mean number of open web pages, applications, and file system folders during baseline phase.

Several Reasons for Not Closing Artifacts Exist. Developers have various reasons for not closing currently unused artifacts, ranging from practical reasons (e.g., saving time in the short term) to psychological factors. In the pre-study survey, two out of six participants mentioned that refinding and opening artifacts introduce time overhead that they want to avoid. P_3 explained in the interview that *"[I] keep everything open because [...] you still lose time when you close stuff"*. Someone else mentioned that they usually forget to close artifacts as the artifacts do not impact the participant's work. Another user simply does not see a need to close unused applications. Two out of six users mentioned that they do not close some artifacts as this resets the artifacts' state, for example, the CLI history. Further, psychological reasons were stated by two out of six participants. P_3 reported that *"I just have a lot of stuff open, and I kind of like it when it is a bit messy"*. P_5 mentioned: *"It would make me feel guilty if I closed applications related to a task that I haven't finished yet. I know that I should be working on the task right now, but I'm not because I have other things to do"*. However, all participants agree that a cluttered workspace affects them negatively at some point, as more time is needed to switch between windows, by getting distracted by other windows, or as a high number of open artifacts can feel demotivating.

Browser Is One of the Most Valuable Artifacts to Keep Open. Apart from a programming environment, the browser and communication applications are most valuable for developers and are almost always kept open. In the final interview, all six participants mentioned that the browser is essential to include in snapshots as it is used for various activities. P_5 even reported mainly working in the browser. This finding is also visible in the collected snapshot data. In 81% of the snapshots (34 out of 42), the browser was a selected artifact. Similar answers were received in the pre-study survey, where the browser and communication applications, such as *Slack*, *Microsoft*

Teams, or a mail client were reported to be always kept open. The importance of communication apps is supported by the finding that these apps were also added to the list of applications that should never be automatically closed by all participants who used this feature (three out of three).

Finding 7: The number of artifacts participants keep open and their reasons for doing so differ. However, the reported timing when they close artifacts is similar, and participants perceive the same artifacts (and especially the browser) as valuable.

6.4.2 Impact of Task Context Snapshot Approach on Workspace Clutter

Based on the responses of participants, the approach positively impacts the users' ability to reduce workspace clutter. While this effect can be observed right after curating a snapshot, the collected data regarding the number of open artifacts during both study phases generally can not support this finding. More insights regarding these qualitative and quantitative insights are provided in the following.

Approach Reduces Perceived Clutter. All participants mentioned in the final interview that the approach made them more aware of the number of unused artifacts they keep open in the background and helped reduce workspace clutter. Participant P_4 mentioned *"I really could reduce the number of windows that were open by just closing all the ones from the previous tasks and just really opening the necessary ones for the next task"*. Another stated that they usually have issues with closing artifacts, but *"with this tool, it was way less cluttered"* (P_5). Two out of six participants also reported that the tool helped them on a practical side to declutter by using the artifact closing functionality. However, one participant pointed out that they are unsure if this is faster than closing artifacts using shortcuts.

According to the end-of-day questionnaires, where participants were asked to rate perceived clutter on a 5-point Likert scale from less to more cluttered than usual, clutter decreased non-significantly. During the baseline phase, the mean reported value was 2.83 (SD=0.70, 20 data points), and during the intervention phase 2.60 (SD=0.68, 24 data points). Applying Welch's t-test results in a t-value of 1.12 and a p-value of 0.27 with 41.00 degrees of freedom. Using a significance level of 5% (α = 0.05), the two mean values are not significantly different. However, it should be considered that only little data was available and that participating in the study itself likely affected the participants' artifact-closing behavior. Two users explicitly pointed out that already in the baseline phase, they reflected more about which artifacts are no longer needed as they knew a tool was collecting data in the background. This might have affected the results.

Snapshots Have Short-Term Impact on Number of Open Artifacts. Developers had fewer artifacts open right after they curated snapshots. This behavior is visible when inspecting the continuous data stream containing the number of open artifacts. Right after snapshot curation, sharp decreases are frequently present, as depicted in Figure 6.3. In the illustrated case, the participant used the TaskSnap tool's save-and-close feature, which was overall used for 25 out of 42 snapshots. To see what impact snapshot curation had on average, we calculated the relative change between the number of open artifacts right before and after a user curated a snapshot. Curation events for which no data was available immediately before and after were discarded, resulting in 35 considered entries. On average, the relative change was -20.6% (SD=20.3%), and the number of artifacts was reduced in 77% of the cases (27 out of 35). The boxplot in Figure 6.4 depicts the distribution of calculated relative changes. While these numbers suggest that the approach helped to reduce workspace clutter, it has to be considered that artifacts potentially would have been closed anyways at these points in time, for example, after finishing a task or at the end of the workday. Further, the decrease is only short-term. According to the data stream, the number of artifacts can increase quickly again (as visible in Figure 6.3), for example, because another task is started.

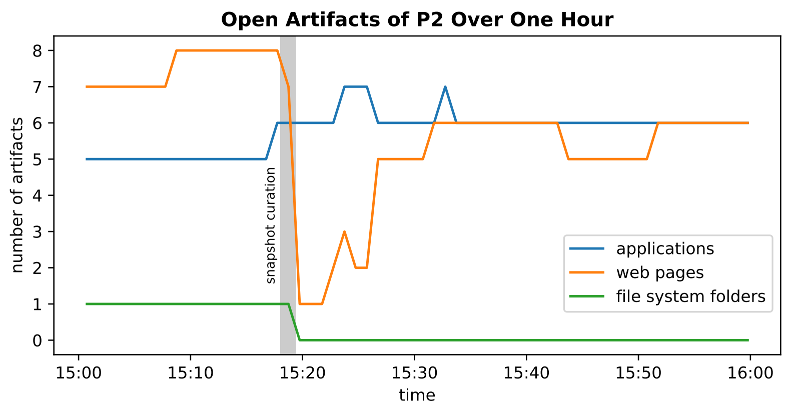


Figure 6.3: Number of open applications, web pages, and file system folders of P_2 over one hour.

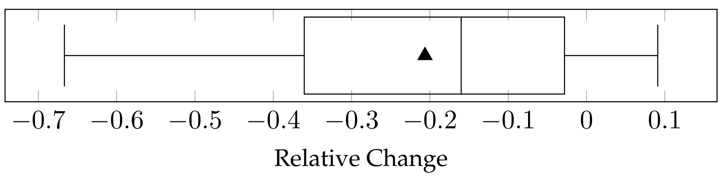


Figure 6.4: Relative change in number of open artifacts around snapshot curation.

Ambivalent Long-Term Impact on Number of Open Artifacts. Only for two out of six participants the mean number of open artifacts decreased between the study phases. For the other participants, the mean remained similar or increased. To illustrate this, the number of open artifacts for each participant and study phase is depicted as boxplots in Figure 6.5. IDE files were not considered, as some users reported that they used Visual Studio Code only infrequently. While this finding counteracts the participants' qualitative responses, it must be considered that the number of open artifacts is affected by various factors and that the presence or absence of the proposed approach is just one of them. The tasks performed during the two study phases can differ heavily, with different numbers of required artifacts. In addition, as mentioned before, two participants (P_2 and P_6) pointed out that already taking part in the baseline phase made them much more aware of no longer needed artifacts compared to their usual work practices.

Finding 8: Developers perceived that the approach helped them to reduce the number of artifacts they keep open. However, the collected quantitative data only reveals this effect in the short term.

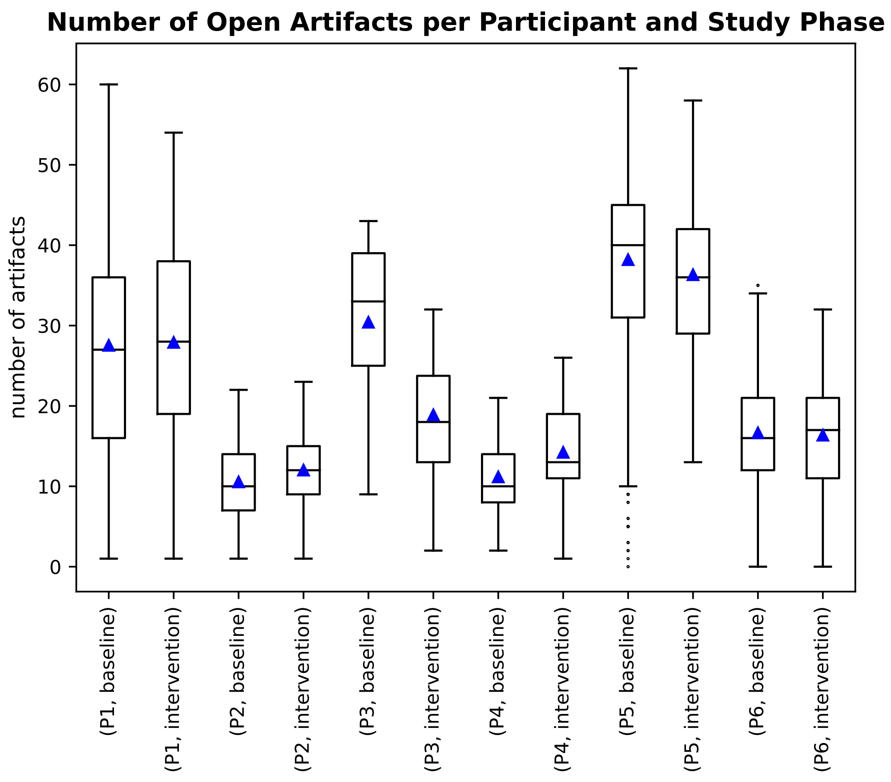


Figure 6.5: Number of open artifacts per participant and study phase.

Chapter 7

Discussion

The results of the pilot study provide first evidence regarding the acceptance and impact of a task context snapshot approach. This chapter discusses the main findings, relates them to previous research, and outlines potential future work.

Unobtrusiveness of the Approach. A main positive feedback participants mentioned regarding the proposed approach is its unobtrusiveness, allowing a seamless integration into existing workflows. At times the user sees value in creating a snapshot, they can do so, but they are not required to use the approach if it does not fit into their workflow at that moment. Further, as the curation step is semi-automated, snapshot curation can be performed quickly, and at the time it fits best. Even though the approach is unobtrusive, users created a reasonable amount of snapshots in the pilot study. These results support our initial design choice that users should not be forced to take any action when starting a task, such as specifying the task they are working on, as this can be cumbersome. Still, many related approaches require this step [KM06, DDJ+05, SFM21, Kap03]. A downside of the approach's unobtrusiveness is that the user can forget to create snapshots at suitable moments, which two participants mentioned. We hypothesize that the physical button that received much positive feedback and lies on the user's desk can be helpful in this regard. As intended actions are always stored together with a retrieval cue in memory [PR12], the physical button can act as a visual cue to create snapshots. Because participants of the pilot study were part-time developers and many had no fixed workplace setup, the button was not always connected. However, we assume that in most real-world office setups with developers, the button is always close to the user's field of view and can remind them subtly of the tool. In future work, users could be further reminded about the tool by prompting them to create a snapshot at the end of the workday as proposed by one participant: *"Unless I created a snapshot earlier [...], maybe there's like a small pop-up that says, hey, do you want to take a snapshot?"* (P_1).

Individual Preferences for Usage. The pilot study underlined that users differ regarding their existing task-switching strategies, application-closing behavior, and whether they see more value in supporting the working or mental task context. Likely, this is also related to the finding that participants used the approach for different use cases, like the one participant who created a snapshot for each higher-level project. When we developed the TaskSnap tool, some design decisions were made based on our own workflows and experience. However, these assumptions may not be ideal for all types of users. For instance, some users prefer to keep many web pages open in parallel (e.g., 26.8 on average for P_5), which can lead to clutter in the *Snapshot Curation Window* and *Snapshot Gallery*. Research has identified these different artifact preservation strategies by describing them on a scale between "minimalism" and "hoarding" [VJM18]. By now, our approach tries to address various user preferences and use cases. While a future approach should consider keeping support for working and mental task context, as there is still little research that tries to support both concepts [PD10], it could be valuable to primarily target people who have issues decluttering their workspace. The pilot study participants already stated different practical and psychological reasons for not closing artifacts. Learning more about how these concerns could be addressed and adapting the approach to their needs could be an interesting direction for future work.

Using the Approach for Instant Task Switches. We designed the approach to be valuable during instant task switches, allowing the user to quickly capture the current context and close artifacts. In contrast to this design choice, participants rarely used the approach in these situations. To some extent, this finding can be explained by the fact that users of the pilot study typically only work on a few tasks per day and, according to the final interviews and end-of-day questionnaires, do not have to switch quickly between them. As related work came to the conclusion that developers frequently switch between tasks, this was likely caused by the fact that all participants were part-time developers. However, as presented in Section 6.2.1, two users mentioned general concerns regarding the approach's applicability for instant switches. The argument of one participant that it is hard to predict sometimes how long it will take until a task is continued should be taken into account when further developing the approach. Encouraging users to postpone the curation step when instantly switching the task is not always a solution, as one user pointed out that artifacts have to be closed immediately to reduce workspace clutter. Therefore, to encourage users to create snapshots also for potentially short task switches, the effort to curate snapshots could be further reduced. For example, this could be achieved by improving the automatic pre-selection of artifacts by choosing different weights when calculating relevance or refining the automatically generated texts regarding the user's mental context (discussed in the next paragraph in more detail). Another future approach could address the need to create snapshots retrospectively, for example, at the time the user realizes that the task they switched to lasts longer than expected. When curating a snapshot, the user could be provided with a timeline to specify a time window during which an artifact must have been active in order to be listed. A timeline-based approach to refind artifacts is successfully applied by the commercial tool *Rewind*[1] and previous research [PD10, HL20]. These proposed directions of future work could improve the approach's applicability for work scenarios that contain more instant task switches. Additionally, in a future study, participants who experience more instant switches could be selected by recruiting full-time developers.

Improving Mental Task Context Summaries. The results of the pilot study show that participants see value in providing support to restore the mental task context and made use of the possibility to write down texts in this regard. While the summaries of recently performed actions were rather short, the texts about the users' intent were more extensive. Even though participants generally saw value in filling out the two texts, several users mentioned that writing useful notes can be time-consuming and that they were sometimes too lazy to provide extensive notes. *"Personally, I'm a bit a lazy writer. So I usually just entered keywords"* (P_3), *"Sometimes I was just lazy, and I was like, oh yeah, I will know what I was about to do"* (P_6). Based on this feedback it could be valuable to investigate the possibilities of extending the pre-writing texts in future work. One approach could be to add additional data sources, to get more detailed insights into recently made changes in office applications, for example. This could also support restoring the working context, as more artifacts (e.g., specific pages in a notebook) could be accessed. Another approach could be to extract more information from the already collected data. Currently, the text regarding recent actions contains which artifact was active the longest, the last accessed web page, and which

[1] `https://www.rewind.ai`, verified 30.07.2023

changes were made within the IDE. However, the combination and sequence of artifacts are not considered. These insights could be presented to the user as well. Further, semantic information from the already collected application window titles could be used, as they can contain valuable information to identify a task [SFM21]. Finally, to generate a cohesive summary based on this data, a future approach could investigate the possibility of using a locally run large language model (LLM).

Impact of the Approach Regarding Workspace Clutter. Many developers keep artifacts open when switching tasks. According to a previous study, 45% of knowledge workers regularly apply this strategy. Our pilot study came to a similar conclusion, as participants reported in the pre-study survey only closing artifacts when a task is finished, which can be undesirable. Participants reported that a task context snapshot approach helped them in this regard, but the pilot study also showed that analyzing this effect quantitatively can be difficult. Only for two out of six participants, the mean number of open artifacts decreased between the two study phases, possibly because participants worked on different tasks requiring a different amount of artifacts. A future study could be conducted in a laboratory setting to overcome this limitation. Participants could be faced with a pre-defined set of tasks, and a control and treatment group could be used to analyze the impact the presence of a task context snapshot approach has on the number of open artifacts. Possibly, also conducting a longer field study could provide more insightful quantitative answers, as this would reduce the effect of single tasks on the mean number of open artifacts in each study phase.

Application in Practice. As the results of the pilot study show, the approach is already well-perceived and provides value. While all participants reported that they could imagine continuing to use the TaskSnap tool, only one user asked if and how this is possible. Therefore, to make the approach suitable for long-term usage, some issues mentioned by participants should be addressed. Something that bothered several users was the fact that not all aspects of the working context are captured. The approach currently does not preserve the exact position of application windows, the CLI history, and the grouping of browser tabs in different windows. Moreover, to our surprise, only one participant raised privacy concerns, as the approach needs access to the browser and IDE. However, especially the access to the code base could cause concerns in a corporate environment and should be considered when improving the tool. In this regard, adding more fine-grained settings that allow users to specify which data can be accessed could be helpful. Making these proposed changes could encourage users to integrate the approach also in the long term into their workflows. In addition, after addressing these shortcomings, a larger field study with full-time professionals could be conducted to get further insights regarding the value and impact of the approach.

Chapter 8

Threats and Limitations

The primary threats and limitations of this work are that participants might have adapted their artifact closing behavior as they knew they were tracked, not necessarily representative study participants, and the short duration of the study, which resulted in having only limited data points concerning some aspects of the analysis.

Hawthorne Effect Regarding Number of Open Artifacts. Knowing that the number of open artifacts was tracked during the study could have affected how many unused artifacts were kept open by participants. Two out of six users explicitly pointed out that already participating in the study led to closing artifacts more frequently than usual. While this so-called Hawthorne Effect [MWE14] is not necessarily bad in our case, as our approach tries to support users in closing artifacts, it could have affected the quantitative study results by making the baseline less representative of the user's regular work behavior. This makes statements regarding the impact of the approach more challenging.

Participants Likely Not Representative. All participants of the pilot study were part-time developers and had only little professional software development experience (one to three years). This was the case as participants were mainly recruited through our personal networks. As the work schedules and work environment of part-time developers can differ from full-time professionals, they might not be a representative sample of the group of software developers and data scientists. Among others, this could have impacted the identified use cases of the approach. Further, as we recruited participants through our personal networks, users were likely less critical of the approach and tried to incorporate it as best as they could into their existing workflows. This could have resulted in more positive impressions than the approach would have received with randomly selected users. Moreover, it has to be noted that the set of participants lacks diversity. Only one female user was part of the study, and all developers were between 23 and 27 years old. The results show that participants differ in how they manage task context capturing and resumption and how they organize artifacts. Hence, a more diverse and larger set of participants could make results more generalizable and help identify the aspects of the approach that must be improved to support a broad range of developers.

Limited Number of End-Of-Day Responses Due To Short Study Duration. As each of the two study phases was only around five days long, the number of answered end-of-day questionnaires is limited. In both phases, the user was asked how much time they needed to restore their working and mental context during the most recent task switch and if they perceived their workspace as cluttered. We asked those questions in both study phases to compare the given answers. However, as only a few data points are available for each study phase, the validity of the insights gained from this comparison is questionable. Longer study phases or asking the user several times during the day would help to get more meaningful insights.

Chapter 9

Conclusion

The workday of software developers and data scientists is highly fragmented, as they frequently switch between multiple tasks. These task switches can be time-intensive, as the user has to rebuild their working task context (applications, documents, folders) and mental task context (task knowledge, goals, intentions). Several approaches exist that help the user in restoring the working context. However, they usually lack support to restore the mental task context or are not specifically targeting the needs of developers. This thesis introduces a semi-automated approach that supports developers in task context capturing and resumption by allowing them to create snapshots of a task's context at any point in time. Users can then curate the snapshot when it fits them best and later restore it to support task resumption. We implemented the proposed approach in a technology probe called TaskSnap and tested it in a two-week pilot study.

Participants reported that the approach fitted well into existing workflows by being unobtrusive and that it supported restoring the working and mental task context. On average, 1.24 snapshots were created per day and user, and often, the approach was used when the user had plenty of time to create and curate snapshots, like at the end of the workday. Using the approach during instant task switches was less common, likely because participants did not experience these situations often as they were part-time developers. Further, the approach reduced the fear of forgetting to persist artifacts and could reduce perceived workspace clutter. However, no general decrease in the number of open artifacts was observed during the study intervention, possibly because other factors, like the type of tasks performed, affected the number of open artifacts more heavily.

In a future approach, the user's mental task context could be automatically described more thoroughly. This could improve the approach's applicability to instant task switches, as it would further reduce the time needed to curate snapshots. In addition, after incorporating the participants' feedback, a larger field study with full-time developers could be conducted to get more representative insights regarding the approach's value and impact.

Appendix A

Appendix

A.1 Questions Asked In Pre-Study Survey

Demographics.

- What is your job title?
- What is your job role?
  - Individual contributor
  - Team Lead
  - Manager
  - Other
- What gender do you identify as?
  - Female
  - Male
  - Other
  - Prefer not to say
- What is your age in years?
- How many years of professional software development experience do you have?
- Which operating system are you using during the study?
  - Windows
  - macOS
- Please estimate on how many main work tasks you are (on average) working on per day:
  - I work on only one main work task per day
  - I work on 2-3 main work tasks per day
  - I work on more than 3 main work tasks per day
- On which types of tasks are you working regularly? (at least on a weekly basis)[1]

[1]The provided list of task types is based on findings by Meyer et al. [MSZ+22, MBBZ21]

- Development: Coding
- Development: Debugging
- Development: Testing
- Development: Navigation
- Development: Search
- Development: Documentation
- Development: Code Review
- Development: Specification
- Personal
- Awareness & Teamwork
- Administrative
- Planned Meeting
- Unplanned Meeting
- Other (input field)

Questions About Task Switching and Resumption Routines.

- Working on multiple different tasks during a day and switching between them can be challenging. How do you manage / organize your open tasks you are working on?
- During a normal workday, are there situations when you have to quickly switch from one task to another? Why? (e.g. before a planned meeting, when a bug has to be fixed immediately, or when a co-worker needs some help)
- When I switch to another task and it is likely that I will continue working on the task later, I usually close artifacts (e.g. applications, windows, files, browser tabs)...
  - Immediately
  - Soon
  - Moderately soon
  - After some time
  - After a long time
  - I don't close artifacts until the task is finished
- Once I complete a task entirely, I usually close artifacts (e.g. applications, windows, files, browser tabs) that I no longer need...
  - Immediately
  - Soon
  - Moderately soon
  - After some time

  - After a long time

- If you tend to keep applications open for a long time: What prevents you from closing applications and windows that you no longer use for the moment? (skip this question if this does not apply to you)
- Normally, my workspace contains...
  - Lots of windows that are no longer relevant
  - Several windows that are no longer relevant
  - Some windows that are no longer relevant
  - A few windows that are no longer relevant
  - No windows that are no longer relevant
- Are there apps / windows that you always keep open and (almost) never close? Which one are these and why do you keep them open?
- When considering the number of applications and windows that I usually keep open on my work computer, I think that my workspace is...
  - Cluttered
  - Somewhat cluttered
  - Neither cluttered nor cleaned up
  - Somewhat cleaned up
  - Cleaned up
- Does a cluttered workspace impact you in any way (e.g. productivity, mood, focus)? Why?
- While working on a task, how and where do you keep track of your working task context (files, code, applications, and web pages associated with the task)?
- While working on a task, how and where do you keep track of your mental task context (task knowledge, ideas, goals, and intentions of what to work on next)?
- What strategies do you have that support you in switching and resuming tasks? (for example, when you are coding and have an idea about another task, or when you are asked to fix an urgent bug, but want to finish the current task first). If you write down notes: Where do you write them down?
- Did you already try out digital tools that can support task switching and resumption? If yes, which tools did you try?

A.2 Questions Asked In End-Of-Day Questionnaire

Questions Asked In Both Study Phases.

- Did you work on multiple tasks today?
  - Yes
  - No

  If Yes:

  - Did you have to instantly switch from one task to another today? If yes, what triggered that particular task switch?
  - Briefly describe your most recent task switch
  - For your most recent task switch today, how much time did you need to restore the working context (apps, windows, files. . .) of the task?
    - almost no time
    - little time
    - moderate amount of time
    - quite a bit of time
    - much time
  - For your most recent task switch today, how much time did you need to restore the mental context (goals, plans, mental model. . .) of the task?
    - almost no time
    - little time
    - moderate amount of time
    - quite a bit of time
    - much time
- Overall, I felt that my workspace was...
  - less cluttered than usual
  - slightly less cluttered than usual
  - as cluttered as usual
  - slightly more cluttered than usual
  - more cluttered than usual
- Additional Comments

Questions Asked Only During Baseline Phase.

- Today, how did you keep track of important task information that might be needed later? (if it's the same method that you described yesterday in detail, please state so instead)
- Was there a situation today where a tool to support your task switching and resumption would have been helpful? If yes, how could it have helped you?

Questions Asked Only During Intervention Phase.

- What was your motivation to create a snapshot at this point?
- What was this task about?[2]
  - Development
    - Coding
    - Debugging
    - Testing
    - Navigation
    - Search
    - Documentation
    - Code Review
    - Specification
  - Personal
  - Awareness & Teamwork
  - Administrative
  - Planned Meeting
  - Unplanned Meeting
  - Other (input field)
- Have there been additional task switches that you did not capture by a snapshot? What was the reason for not creating one?

A.3 Questions Asked In Task Resumption Questionnaire

- How difficult was it to reidentify this task from the Snapshot Gallery?
  - very difficult
  - difficult
  - neither difficult nor easy
  - easy
  - very easy
- Which features or information helped you to reidentify the snapshot?
  - Snapshot Name
  - Timestamp
  - Search
  - Position of snapshot in list
  - Application Icons

[2]The provided list of task types is based on findings by Meyer et al. [MSZ+22, MBBZ21]

  - Listed Artifacts
  - Task summary text (What was I doing?)
  - Task intent text (What was I about to do?)
  - Git Information
  - Other (input field)

- Is there something that could have helped you better reidentify the snapshot? And if so what?

A.4 Guiding Interview Questions of Final Interview

Value and Application of the Approach

1. Ice breaker: Please talk a bit about how you've used the TaskSnap tool during the study.
2. On what kinds of tasks did you work during the last two weeks?
3. During which situations did you use the TaskSnap tool? (RQ2a)
   (a) [If not already mentioned, ask some]: Did you use the TaskSnap tool e.g. when being interrupted, before a meeting, before lunch, at the end of the day/week to clean up your workspace, to simply note down task-related information, at the end of a task, ...
   (b) [prompt, if used in this situation] How well did it support you to capture and resume the task context when being interrupted?
   (c) [prompt, if used in this situation] How well did it support you to pause a task for a longer period and resume it, e.g. after the weekend?

Snapshot Creation and Curation

4. How did your snapshot creation workflow look like? (RQ2a)
   (a) [prompt] When did you curate snapshots? Immediately after their creation, after postponing it, on resumption. . . ?
   (b) [prompt] What impacted the timing of when you curated a snapshot?

With our approach, we want to support users in preserving their working task context (apps, files, tabs...) and mental task context (current task description and next subtasks/steps in the Post-It-like views)...

Working Context

5. What do you think about how TaskSnap supports restoring the working context? Did it support you in any way and possibly save you some time? (RQ2b)
6. Was the automatic preselection of artifacts accurate and useful? Why (not)? (RQ2a)
   (a) [If not already answered] Should more / less artifacts be pre-selected?
7. In the work you did over the course of the study: What have been the most relevant artifacts to persist for your captured tasks? (RQ1)
   (a) [Ideas if the participant is unsure]: Browser tabs (GitHub, Stack Overflow), IDE files, note-taking apps, file explorer tabs...

8. Did you use the feature to clean up your workspace right after creating a snapshot? (RQ2a)

Mental Context

9. What do you think about how TaskSnap supports restoring the mental context (current task description and next subtasks/steps in the Post-It-like views)? Did it support you in any way and possibly save you some time? (RQ2b)
    (a) [If not already answered] Was it useful to see the two Post-Its when restoring a snapshot?
10. Was the pre-filled text useful? How did you adapt it / what did you add to it? (RQ2a)

Synthesis

11. How much effort did you invest in (a) manually selecting/deselecting artifacts and (b) filling the two Post-Its to support restoring the mental context? (RQ2b)
    (a) [If much time] Why was it worth spending so much time?
    (b) [If not much time] Why didn't you invest more time?
    (c) What do you think: Which of the two aspects is more valuable? Why?

Snapshot resumption and visualization

12. Do you think the TaskSnap tool supported you with switching between and resuming tasks? How? (RQ2a / RQ2b)
    (a) [prompt] Did using TaskSnap help you re-finding information when resuming a task?
13. For resuming a task/snapshot, how did you re-identify which snapshot you wanted to resume? (RQ2a)
    (a) [prompt] Based on single, very distinctive artifacts? Based on combinations of artifacts? Based on the title?
    (b) [prompt] Was there something that was missing to help you re-identify a snapshot?
    (c) [prompt] How do you like the Snapshot Gallery, and do you think it could be somehow improved?

Impact

14. Did using TaskSnap impact the number of applications/tabs/files that you kept open? (RQ2b)
15. How well did the TaskSnap tool fit into your existing workflows? (RQ2b)
    (a) Did it need much time and effort to create and curate snapshots?
    (b) [If not already answered] Did it conflict with existing workflows? (e.g. existing notebooks or task tools where task-related information is stored)
    (c) Did you adapt your task switching and resumption routine for the study?
    (d) Do you think some aspects will remain when you stop using the tool?
16. Did participating in this study make you think more about your existing task switching and resumption strategies? Did you learn something? (RQ1)

17. We were previously talking about situations in which you've created snapshots. Can you envision further situations or use cases where the TaskSnap tool could be helpful? (RQ2a)

Further Use

18. Do you have suggestions on how to improve the TaskSnap tool? (RQ2a)
    (a) something you like/dislike?
    (b) [If time left] Something we were not sure about while testing the approach internally: Was the small popup after creating a snapshot useful or unnecessary in your workflow?
19. Could you imagine continuing to use the TaskSnap tool for your daily work? (RQ2a)

Wrap-up

20. Is there anything else you want to mention regarding the TaskSnap tool or study?

Bibliography

[AT02] Erik M. Altmann and J. Gregory Trafton. Memory for goals: an activation-based model. *Cognitive Science*, 26(1):39–83, 2002.

[BC12] Virginia Braun and Victoria Clarke. *Thematic analysis.*, pages 57–71. American Psychological Association, 01 2012.

[BSW08] Michael S. Bernstein, Jeff Shrager, and Terry Winograd. Taskposé: Exploring fluid boundaries in an associative window visualization. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08, page 231–234, New York, NY, USA, 2008. Association for Computing Machinery.

[BVKKS08] Michael Bernstein, Max Van Kleek, David Karger, and M. C. Schraefel. Information scraps: How and why information eludes our personal information management tools. *ACM Trans. Inf. Syst.*, 26(4), oct 2008.

[CH86] Stuart K. Card and Austin Henderson. A multiple, virtual-workspace interface to support user task switching. *SIGCHI Bull.*, 18(4):53–59, may 1986.

[CHW04] Mary Czerwinski, Eric Horvitz, and Susan Wilhite. A diary study of task switching and interruptions. *Conference on Human Factors in Computing Systems - Proceedings*, 6, 02 2004.

[CKD21] Padraig Cunningham, Bahavathy Kathirgamanathan, and Sarah Jane Delany. Feature selection tutorial with python examples. *CoRR*, abs/2106.06437, 2021.

[DDJ$^+$05] Anton N. Dragunov, Thomas G. Dietterich, Kevin Johnsrude, Matthew McLaughlin, Lida Li, and Jonathan L. Herlocker. Tasktracer: A desktop environment to support multi-tasking knowledge workers. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, IUI '05, page 75–82, New York, NY, USA, 2005. Association for Computing Machinery.

[GM04] Victor M. González and Gloria Mark. "constant, constant, multi-tasking craziness": Managing multiple working spheres. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, page 113–120, New York, NY, USA, 2004. Association for Computing Machinery.

[HL20] Donghan Hu and Sang Won Lee. Screentrack: Using a visual history of a computer screen to retrieve documents and web pages. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[HL22] Donghan Hu and Sang Won Lee. Scrapbook: Screenshot-based bookmarks for effective digital resource curation across applications. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, UIST '22, New York, NY, USA, 2022. Association for Computing Machinery.

[IH07] Shamsi T. Iqbal and Eric Horvitz. Disruption and recovery of computing tasks: Field study, analysis, and directions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, page 677–686, New York, NY, USA, 2007. Association for Computing Machinery.

[Kap03] Victor Kaptelinin. Umea: Translating interaction histories into project contexts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, page 353–360, New York, NY, USA, 2003. Association for Computing Machinery.

[KBVVT16] Ioanna Katidioti, Jelmer Borst, Marieke Van Vugt, and Niels Taatgen. Interrupt me: External interruptions are less disruptive than self-interruptions. *Computers in Human Behavior*, 63:906–915, 06 2016.

[KM06] Mik Kersten and Gail C. Murphy. Using task context to improve programmer productivity. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, SIGSOFT '06/FSE-14, page 1–11, New York, NY, USA, 2006. Association for Computing Machinery.

[KMCA06] Amy J. Ko, Brad A. Myers, Michael J. Coblenz, and Htet Htet Aung. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Transactions on Software Engineering*, 32(12):971–987, 2006.

[LFP82] Sarah Lichtenstein, Baruch Fischhoff, and Lawrence D. Phillips. *Calibration of probabilities: The state of the art to 1980*, page 306–334. Cambridge University Press, 1982.

[Maa09] Walid Maalej. Task-first or context-first? tool integration revisited. In *Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering*, ASE '09, page 344–355, USA, 2009. IEEE Computer Society.

[Maa10] Walid Maalej. *Intention-Based Integration of Software Engineering Tools*. PhD thesis, Technical University Munich, 2010.

[MBBZ21] André N. Meyer, Earl T. Barr, Christian Bird, and Thomas Zimmermann. Today was a good day: The daily life of software developers. *IEEE Transactions on Software Engineering*, 47(5):863–880, 2021.

[MBM$^+$17] André N. Meyer, Laura E. Barton, Gail C. Murphy, Thomas Zimmermann, and Thomas Fritz. The work life of developers: Activities, switches and perceived productivity. *IEEE Transactions on Software Engineering*, 43(12):1178–1193, 2017.

[MER17] Walid Maalej, Mathias Ellmann, and Romain Robbes. Using contexts similarity to predict relationships between tasks. *J. Syst. Softw.*, 128(C):267–284, jun 2017.

[MFMZ14] André N. Meyer, Thomas Fritz, Gail C. Murphy, and Thomas Zimmermann. Software developers' perceptions of productivity. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, page 19–29, New York, NY, USA, 2014. Association for Computing Machinery.

[MLNL23] Rongjun Ma, Henrik Lassila, Leysan Nurgalieva, and Janne Lindqvist. When browsing gets cluttered: Exploring and modeling interactions of browsing clutter, browsing habits, and coping. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[MMV$^+$01] Blair MacIntyre, Elizabeth D. Mynatt, Stephen Voida, Klaus M. Hansen, Joe Tullio, and Gregory M. Corso. Support for multitasking and background awareness using interactive peripheral displays. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, page 41–50, New York, NY, USA, 2001. Association for Computing Machinery.

[MSZ$^+$22] André N. Meyer, Chris Satterfield, Manuela Züger, Katja Kevic, Gail C. Murphy, Thomas Zimmermann, and Thomas Fritz. Detecting developers' task switches and types. *IEEE Transactions on Software Engineering*, 48(1):225–240, 2022.

[MW03] Merriam-Webster. *Merriam-Webster's Collegiate Dictionary*. Merriam-Webster, 11th edition, July 2003.

[MWE14] Jim McCambridge, John Witton, and Diana R. Elbourne. Systematic review of the hawthorne effect: New concepts are needed to study research participation effects. *Journal of Clinical Epidemiology*, 67(3):267–277, March 2014.

[PD10] Chris Parnin and Robert DeLine. Evaluating cues for resuming interrupted programming tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 93–102, New York, NY, USA, 2010. Association for Computing Machinery.

[PR11] Chris Parnin and Spencer Rugaber. Resumption strategies for interrupted programming tasks. *Software Quality Journal*, 19(1):5–34, mar 2011.

[PR12] Chris Parnin and Spencer Rugaber. Programmer information needs after memory failure. In *2012 20th IEEE International Conference on Program Comprehension (ICPC)*, pages 123–132, 2012.

[PRM$^+$20] Jan Pilzer, Raphael Rosenast, André N. Meyer, Elaine M. Huang, and Thomas Fritz. Supporting software developers' focused work on window-based desktops. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[RHC$^+$04] George Robertson, Eric Horvitz, Mary Czerwinski, Patrick Baudisch, Dugald Ralph Hutchings, Brian Meyers, Daniel Robbins, and Greg Smith. Scalable fabric: Flexible task management. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, page 85–89, New York, NY, USA, 2004. Association for Computing Machinery.

[RKF$^+$20] Anastasia Ruvimova, Junhyeok Kim, Thomas Fritz, Mark Hancock, and David C. Shepherd. "transport me away": Fostering flow in open offices through virtual reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–14, New York, NY, USA, 2020. Association for Computing Machinery.

[RMHF23] Roy Adrian Rutishauser, André N. Meyer, Reid Holmes, and Thomas Fritz. Semi-automatic, inline and collaborative web page code curations. In *International Conference on Software Engineering (ICSE'23)*, Mai 2023.

[RTH17] Adam Rule, Aurélien Tabard, and Jim Hollan. Using visual histories to reconstruct the mental context of suspended activities. *Human–Computer Interaction*, 32(5-6):511–558, 2017.

[SBL98] Rini Solingen, Egon Berghout, and Frank Latum. Interrupts: Just a minute never is. *IEEE Software*, 15:97–103, 09 1998.

[SBR$^+$03] Greg Smith, Patrick Baudisch, George Robertson, Mary Czerwinski, Brian Meyers, and Daniel Robbins. Groupbar: The taskbar evolved. In *(2003) OZCHI 2003 Conference for the Computer-Human Interaction Special Interest Group of the Human Factors Society of Australia*, January 2003.

[SDNL19] Jennifer Sloane, Christopher Donkin, Ben R Newell, and Garston Liang. What's lagging in our understanding of interruptions?: Effects of interruption lags in sequential decision-making. In *CogSci*, pages 1063–1069, 2019.

[SFKD22] Christina Schneegass, Vincent Füseschi, Viktoriia Konevych, and Fiona Draxler. Investigating the use of task resumption cues to support learning in interruption-prone environments. *Multimodal Technologies and Interaction*, 6(1), 2022.

[SFM21] Chris Satterfield, Thomas Fritz, and Gail C. Murphy. Identifying and describing information seeking tasks. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, ASE '20, page 797–808, New York, NY, USA, 2021. Association for Computing Machinery.

[SM07] Izzet Safer and Gail C. Murphy. Comparing episodic and semantic interfaces for task boundary identification. In *Proceedings of the 2007 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '07, page 229–243, USA, 2007. IBM Corp.

[TABM03] J.Gregory Trafton, Erik M Altmann, Derek P Brock, and Farilee E Mintz. Preparing to resume an interrupted task: effects of prospective goal encoding and retrospective rehearsal. *International Journal of Human-Computer Studies*, 58(5):583–603, 2003.

[Tul04] Endel Tulving. Episodic memory: from mind to brain. *Annual review of psychology*, 53:1–25, 2004.

[VJM18] Francesco Vitale, Izabelle Janzen, and Joanna McGrenere. Hoarding and minimalism: Tendencies in digital data preservation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA, 2018. Association for Computing Machinery.

[WKM$^+$18] Alex C. Williams, Harmanpreet Kaur, Gloria Mark, Anne Loomis Thompson, Shamsi T. Iqbal, and Jaime Teevan. Supporting workplace detachment and reattachment with conversational intelligence. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–13, New York, NY, USA, 2018. Association for Computing Machinery.

[ZCM$^+$17] Manuela Züger, Christopher Corley, André N. Meyer, Boyang Li, Thomas Fritz, David Shepherd, Vinay Augustine, Patrick Francis, Nicholas Kraft, and Will Snipes. Reducing interruptions at work: A large-scale field study of flowlight. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 61–72, New York, NY, USA, 2017. Association for Computing Machinery.