Mercury: Empowering Programmers' Mobile Work Practices with Microproductivity

Alex C. Williams[1], **Harmanpreet Kaur**[2], **Shamsi Iqbal**[3], **Ryen W. White**[3], **Jaime Teevan**[3], **Adam Fourney**[3]

[1]University of Waterloo, [2]University of Michigan, [3]Microsoft Research
alex.williams@uwaterloo.ca, harmank@umich.edu, {shamsi,ryenw,teevan,adamfo}@microsoft.com

ABSTRACT

There has been considerable research on how software can enhance programmers' productivity within their workspace. In this paper, we instead explore how software might help programmers make productive use of their time while away from their workspace. We interviewed 10 software engineers and surveyed 78 others and found that while programmers often do work while mobile, their existing mobile work practices are primarily exploratory (e.g., capturing thoughts or performing online research). In contrast, they *want* to be doing work that is more grounded in their existing code (e.g., code review or bug triage). Based on these findings, we introduce *Mercury*, a system that guides programmers in making progress on-the-go with auto-generated microtasks derived from their source code's current state. A study of Mercury with 20 programmers revealed that they could make meaningful progress with Mercury while mobile with little effort or attention. Our findings suggest an opportunity exists to support the continuation of programming tasks across devices and help programmers resume coding upon returning to their workspace.

Author Keywords

Programming, microtask, mobile, continuation, interruption.

CCS Concepts

•**Software and its engineering** → **Integrated and visual development environments;** •**Human-centered computing** → **Mobile computing;** *Usability testing;* Lab. experiments;

INTRODUCTION

There are millions of professional programmers, and their numbers are growing significantly faster than previously predicted [11]. However, programmers are not able to fully take advantage of the added opportunities and flexibility that mobile devices offer in getting things done, due to the challenge of working across devices. From large desktops to small wearables, information workers today often use multiple devices to accomplish their work in the most productive way possible [24, 51, 43, 23], but programming presents a unique set of obstacles, such as the reliance on personalized development environments most suited for large workspaces [22], or on tasks not suitable for limited attention scenarios.

UIST '19. October 20–23, 2019, New Orleans, LA, USA

ACM. ISBN 978-1-4503-6816-2/19/10…$15.00
DOI: 10.1145/3332165.3347932

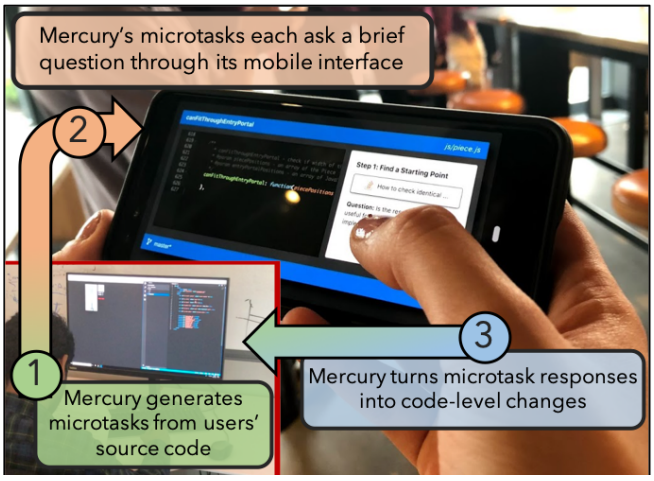


Figure 1: Mercury allows programmers to continue their work on-the-go. (1) When a user leaves their workstation, Mercury generates microtasks from their code, (2) then serves the tasks to their mobile device. These tasks are brief and require little attention. (3) Finally, Mercury integrates the user's microtask responses into the their workstation's source files.

In this paper, we explore how to facilitate programmers' mobile work practices. We conducted two pre-studies to understand how programmers currently use their mobile devices for work: a contextual inquiry with 10 software engineers, and a large-scale online survey with 78 software engineers. We find our participants already perform a myriad of programming-related tasks while mobile, many of them *exploratory* (e.g., related to capturing thoughts or conducting online research). However, they also expressed a desire to perform more mobile tasks that are *grounded* in existing code (e.g., conducting code reviews or triaging bugs), but such tasks are not yet well supported by existing mobile tools. Further, many of their tasks while mobile are intended to support the effective continuation of their work upon returning to their workstation.

Recent research suggests that microproductivity is one particularly beneficial design pattern for bringing mobility to otherwise immobile information work [23, 54]. Building on our pre-studies' findings, we developed a system, named *Mercury*, that interfaces with Visual Studio Code to facilitate mobile task completion and real-time cross-device continuation for programmers. As shown in Figure 1, Mercury orchestrates programmers' work practices by providing them with a series

of auto-generated microtasks on their mobile device based on the current state of their source code. Tasks in Mercury are designed so that they can be completed quickly without the need of much additional context, making them suitable to address during brief moments of downtime. When users complete microtasks on-the-go, Mercury calculates file changes and integrates them into the user's codebase where appropriate. From a user study with 20 participants, we find Mercury's microtask design to be an enjoyable and productive yet lightweight approach to conduct work on-the-go, and one that also aids individuals in resuming their work upon returning to their workstation. In this paper, we specifically:

- Present the notion of *exploratory* and *grounded* microtasks based on programmers' existing and desired practices.
- Introduce Mercury, a mobile programming tool that auto-generates microtasks based on programmers' existing code.
- Find that Mercury's microtasking model effectively allows programmers to continue their work on-the-go.
- Observe that engaging with programming-related microtasks via Mercury spurs users' ability to resume their work.

The remainder of the paper is structured as follows. We present related work, describe the findings of the contextual inquiry, and present the results of the online survey. We then present Mercury and its evaluation, then conclude with a discussion of considerations for designing systems that allow programmers to make progress in their work while on-the-go.

RELATED WORK

Multi-Device Use in Information Work

Increasingly, the human factors literature is examining how information workers use multiple devices to support their work [24, 16]. Karlson et al. [26] found that information workers' mobile work patterns are heavily centered around time. This research has identified a number of barriers to performing mobile work, including the poor usability of mobile web browsers [25] and issues related to resuming tasks across workstations and mobile devices [27]. Certain work tasks appear to be more appropriate for mobile interfaces than others [3, 46]. Information workers often defer engaging with a task based on the device that is available [15, 27]. However, information workers want the ability to continue tasks across devices as they otherwise have to manually transmit data between devices [24, 51]. Research has committed to understanding continuation by studying the design [17], development [45], or evaluation [9, 42, 41] of systems that support it in practice.

Prior research has examined multi-device programming in mobile contexts. Tillman et al. [56, 57] introduced TouchDevelop, a mobile programming environment powered only by touchscreen interactions on the phone. Nguyen et al. [44] explored a similar approach for on-phone debugging. More recently, Husmann et al. [22] examined pathways for supporting developers' multi-device use in stationary "ad hoc scenarios", e.g., while at a cafe. Our work complements this prior research by focusing specifically on mobile contexts in which the user may have limited attention or brief moments of downtime.

Microproductivity and Microtasks

Microproductivity, where smaller parts of a large task are completed through Microtasks, has recently emerged as a strategy for increasing the breadth of information work that can be conducted on mobile devices. Microtasks, defined as smaller tasks decomposed from a larger task into more manageable and meaningful units, present opportunities for getting useful work done in short bursts of time that is generally considered unusable [54]. Research has demonstrated the utility of microtasks by comparing them to their macrotask counterparts, showing microtasks are more resilient to interruptions [8], yield higher quality work [8], can be used to scaffold the cognitive process of maintaining and rebuilding context for complex tasks [7, 28, 50] and make tasks more engaging [19]. Prior work has explored how microtasking can improve the writing process in groups [54], from devices with small screens [43], and for individuals working on their own edits in short bursts of time [23]. Microtasks have also been used to orchestrate teams of actors for specific purposes, such as peer production [60] or scheduling meetings [12], and have been used to systematically perform taxonomy creation [10], for copy-editing [4] and to capture local knowledge [59]. Designing experiences for these scenarios has been explored [30].

Research has shown that programming can benefit from microproductivity-like practices through the lens of crowdsourcing [37], where software development tasks are decomposed into decontextualized units such as: reviewing, testing, and debugging [33, 36]. Work has shown that proper coordination of crowdwork can overcome traditional knowledge sharing challenges in software teams [34]. The literature, however, has yet to bridge the gap between crowdsourced programming, and on-the-go work for individuals. More specifically, there exists little research on how individuals can continue their own software development work across multiple devices and across different attentional states. Unlike prior crowd-powered programming systems, the tool presented here is a selfsourcing [55] tool that allows programmers to source microtasks to themselves while away from their workstation (i.e. based on their own source code). Building on prior microtask definitions, we introduce new programming microtasks designed specifically for the programming context. Speaking to its novelty, our tool enables a new mobile work practice, allowing programmers to continue their work in a low-effort fashion.

Task Resumption

Finally, as noted in the introduction, we interviewed developers and learned that their existing mobile practices were often intended to support task resumption. Past research has shown that task resumption is a general challenge for many information workers, and that a task's difficulty can double after resuming from an interruption [13]. Task resumption is especially challenging for software developers. In a study across multiple large software companies, Solingen et al. [61] observed developers typically required upward of 15 minutes to recover from an interruption, and spent an hour a day managing interruptions. Likewise, a 2006 study on programmers at Microsoft showed that 62% of the 186 respondents believe interruptions are a serious problem for their productivity [38]. Researchers have explored programmers' resumption strategies for interrupted tasks, showing that most programmers' resumption lag – the time spent mentally preparing to resume a task – lasts more than a minute [49]. As such, prior work has evaluated the utility of visual cues based on file navigation in supporting resumption, finding that programmers generally

prefer cues that visualize chronological use [29, 48, 47]. Theories of cognition hypothesize that this resumption lag can be reduced with appropriate planning and rehearsal [1, 58, 63]. Here, we hypothesize that self-sourced programming microtasks have the ancillary benefit of planning and rehearsal, and that this facilitates one's return to their primary workstation.

In summary, past work has shown that information workers routinely use multiple devices throughout the day, but that some classes of work, including programming, are difficult to perform on mobile devices. Microproductivity and self-sourcing has emerged as a technique to broaden the range of work that can be done when mobile. In this paper, we apply these learnings to the domain of software engineering by building a microproductivity tool that supports programmers' task continuation and task resumption needs.

PRE-STUDY: CONTEXTUAL INQUIRY

Before designing a system to empower programmers mobility, we need to better understand programmers' existing mobile work practices and how they complement work practices on primary work devices. We conducted a contextual inquiry [64] to address these questions.

Contextual Inquiry Methods

We recruited 10 software engineers (eight male / two female) at a large software company. Each participant was visited in their personal workspace while they were performing a programming-related task (e.g., prototyping, implementing, or debugging). Each inquiry was conducted by one researcher, and lasted approximately one hour. The researcher took written notes, and interviews were audio recorded. Participants were compensated with $10.00 for their time.

To get insights into opportunities for integration of mobile devices into the programming ecosystem, we designed an interview structure that focused on situations that take individuals out of their workplace and require them to pause their work. The researcher initiated the inquiry by explaining our interest in understanding how they work within their workspace. During the inquiry, each participant was told they would be asked to stop working and briefly chat about their mobile work practices 15 minutes after the inquiry had started. This simulated a planned interruption. Participants were interrupted to chat again 45 minutes after the inquiry had started, but were not given advanced notice of this interruption, simulating an unplanned interruption. In both cases, participants were asked to discuss their mobile work practices and their usage of artifacts around the workstation in the context of these interruptions. Participants were also asked about other scenarios that might unexpectedly take them away from their workspace.

The inquiry was concluded with a 10-minute semi-structured exit interview to better understand existing mobile work practices. Interviews began by asking participants to further discuss practices they described earlier in the inquiry, elaborating on the strengths and shortcomings of these practices in accomplishing work on-the-go. Upon concluding this phase of our study, audio recordings were transcribed. Excerpts were iteratively organized into themes following the practice of open coding and affinity diagramming [39].

Table 1: Currently practiced mobile tasks.

Task	Description
Thought Capture	Writing down or recording general thoughts and ideas related to programming tasks.
Email	Using email for a programming-related task (including emailing content to self).
Online Research	Searching or browsing the internet for information related to a programming task.
Bug Triage	Documenting and reporting on bugs.
Code Review	Reviewing or commenting on existing code.
Debugging	Fixing and testing existing code.
Programming	Creating and writing code.

Contextual Inquiry Findings

Three key themes emerged: 1) participants often engage in activities outside of their primary workspace to make progress of software development tasks using mobile devices, but they rarely interact with code; 2) their existing practices require better support for continuation of programming tasks; and, 3) because of the difficulties in task continuation across devices, they minimize what they need to resume after both mobile and non-mobile work experiences.

Understanding Mobile Work Practices

Table 1 lists details of the programming-related tasks participants reported currently performing from mobile devices. Other than Email, the two most common task types were *Online Research* and *Thought Capture*. Online Research tasks, reported by 6 participants, focus on identifying valuable directions for future programming-related tasks:

> *"It's almost like priming the pump when I start my day, but sometimes it's like I just don't know how to do this."* (P9)

Examples of online research tasks described by participants include searching for relevant Stack Overflow web pages, reading technical documentation online, and watching technical tutorial videos. Thought Capture tasks, reported by five participants, focus on opportunistically recording ideas. Examples include writing notes in a physical notebook or on their phone. Four participants reported occasionally reviewing code and tracking bug reports while on-the-go, but expressed a strong dislike for "the awful user interface" (P3). Only one participant reported debugging and programming on their phone.

Participants were excited to extend their current mobile work practices with tasks that generally enrich their source code. The most commonly desired tasks identified by four participants were code review and the ability to quickly capture thoughts that "come at the wrong time" (P5), such as while driving. One participant (P2) wanted to monitor long-running compilation processes, while another (P4) expressed an interest in using design-oriented tools on-the-go. Three participants highlighted debugging and programming as tasks they would "never" want to do on the mobile phone. Most identified poor user experience as the primary barrier behind adopting mobile tasks into their personal mobile work practices.

Understanding Cross-Device Continuation

Participants reported challenges with information transfer across devices. Email was used as the primary mechanism for transferring information, typically from their mobile device to their primary workstation:

"I emailed myself a few links last night to get them off my plate. It would've been great to have them open automatically when I arrived this morning." (P7)

Using email as a way to continue work relevant to programming adds extra steps in linking the content back to the primary coding environment. Participants did not report continuing any programming related tasks on their mobile device, primarily because there is no effective functionality for doing so.

Understanding Task Resumption

Almost all participants (8) deferred pausing their work until they came to a good break point to minimize the resumption overhead upon return:

"I'm more likely to stop where it'll take less energy for me pick back up. Otherwise, it'll take me longer to connect to the project when I come back." (P4)

Other participants described similar strategies such as "delaying lunch to continue working on the implementation" (P1) and "leaving work a few hours early because I can't finish it before the end of the day" (P7).

Though resumption of work after some time had passed is challenging, participants did not appear to leave explicit cues in their environment to help them with resumption. Participants stated they "might jot down a word or two if its extremely important" (P2). Four participants believed nothing they could do would make resumption easier, noting that "resumption sucks, but I don't think anything can be done to improve it" (P8). The other six were more optimistic. For example, P2 – who currently has no mobile work practice – said:

"If you find something that will help me keep the context alive, I'll definitely start using my phone this way." (P2)

From the contextual inquiry we see opportunities for more flexibility in task execution and easier task resumption if users are provided support to continue work in some capability while they are away from their primary workspace. This inspired us to further explore the promise of using mobile devices to complement existing programming practices.

PRE-STUDY: ONLINE SURVEY

The programmers in our Contextual Inquiry reported using their mobile devices for some tasks and described practices around task continuation and resumption. We conducted an online survey to understand these themes better and generalize them across a broader range of people.

Online Survey

We recruited 78 participants (68 male / seven female / two non-binary) by randomly sampling a company-wide employee list of individuals with job roles that regularly involve programming, including software engineers (70), electrical engineers (3), program managers (2), site reliability engineers (2), and data scientists (1). Participation was voluntary. 71 participants (91%) held at least a college degree, and had three or more years of experience in their current job role. Ownership of a mobile smartphone was the only requirement for participation.

The survey began by asking participants to identify the programming-related tasks that they currently practice while mobile from the task list shown in Table 1, including choices for "Nothing" and "Other." If participants indicated engagement in any mobile programming tasks, they were also asked to provide additional information about the last time they performed the task while mobile. The survey also asked participants to reflect on a programming project that they had not worked on for longer than a month, and estimate the amount of time they would need to feel prepared enough start making progress. Participants were asked to indicate artifacts they would utilize when resuming the task, and whether they believe the resumption overhead for the task could be reduced with proper tooling. To more concretely understand opportunities for future systems, the survey concluded by asking participants to report the utility of a system that allowed them to perform their desired work practices at their own leisure and seamlessly continue work across their devices. Utility was measured with a subset of questions of the Technology Acceptance Model [14], aimed at measuring *perceived usefulness*. A copy of the survey questionnaire is included with this publication as supplementary materials.

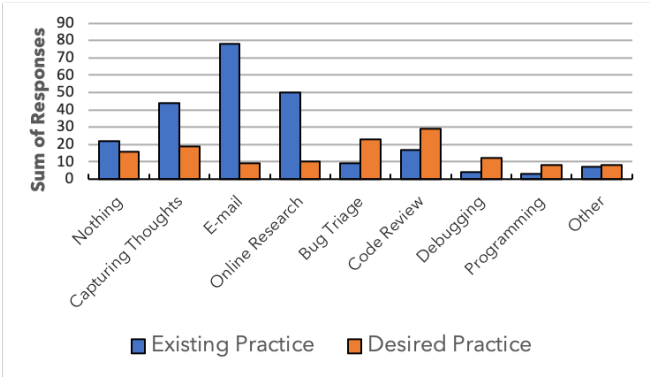


Figure 2: Histogram of existing and desired mobile practices.

Online Survey Findings

Extending our analysis of the three themes from our contextual inquiry, we find that 1) participants' existing mobile work practices are mainly exploratory while their desired work practices are more grounded; 2) continuation is primarily facilitated through email by transferring captured thoughts and online research; and 3) resumption of interrupted work is facilitated with their mobile work practices.

Understanding Mobile Work Practices

We find clear separation between the practices that respondents currently employ and those they desire. Consistent with the Contextual Inquiry, the most frequently reported tasks for existing practices were Email (78), Online Research (50), and Capturing Thoughts (43). The other four task types (and "Other") were all reported far less often, with 20 respondents saying they do no mobile tasks. Existing practices are mainly *exploratory* tasks that support ideation and planning.

In contrast, participants' desired work practices are concentrated on actionable tasks that are much more *grounded* in existing artifacts. The most frequently desired tasks included code review (29) and bug triage (23). While the remaining tasks were desired by fewer than 25% of the 78 participants, 19

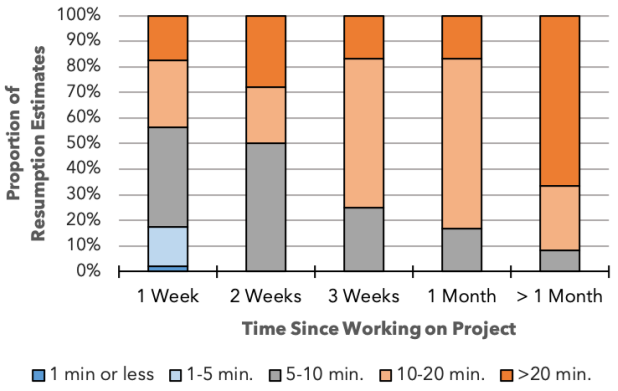


Figure 3: Percentages of resumption time estimates binned by the time passed since pausing the programming task.

participants expressed a desire for capturing thoughts on-the-go as a means for enriching existing source code. Collectively, we use exploratory and grounded tasks to describe our participants' existing and desired mobile work practices. We do not consider email as its primary use was acting as an information channel between devices.

Understanding Cross-Device Continuation

Respondents' existing practices of Thought Capture and Online Research require effective mechanisms for transferring and synchronizing data to integrate the progress made while mobile back into the primary workspace. As we found in our Contextual Inquiry, email was the most commonly used mechanism for transferring information across devices. In reflecting on a recent experience, 19 of the 43 respondents who reported Capturing Thoughts (44%) indicated they used email to transfer brief notes from their mobile device to their primary workstation. 16 respondents (37%) reported using mainstream task management software (e.g., OneNote, Wunderlist) that facilitate cross-device synchronization. The remaining 8 respondents indicated that they left the information on their phone to revisit later, but did not remember to revisit it.

Similarly, 20 of the 50 respondents who conducted Online Research (40%) indicated that they used email to transfer their researched information (e.g., URLs) to themselves. 16 respondents (32%) said they retained information in their working memory (e.g., "keep it in my brain cache" (P45)). Other less common strategies included creating browser bookmarks, writing notes on paper, and sending the information to someone else. All respondents used online research to address a particular problem on their mind. All but one used a search engine for their research.

Understanding Task Resumption

Respondents reported employing mobile work practices to also counteract the effects of pausing work on their primary workstation. For example, a common theme that emerged from respondents who used email is it acts as a mechanism for maintaining and refreshing context while on-the-go and upon returning to the workstation:

> *"It keeps me updated with progress and reduces the time to catch up when I return to my desk."* (P34)

Thought capture and performing online research similarly helps maintain context which in turn supports resumption.

Alongside their practices, we find that our participants recognized the amount of time needed to resume programming work. We asked them to estimate the amount of time it would take to resume a programming task that they last accessed: one week ago, two weeks ago, etc., up to more than a month ago. Figure 3 shows the aggregate responses. Across each time interval, participants' most frequently estimated it would take at least five minutes for them to feel prepared, and even longer for tasks paused for longer than three weeks. To that end, 69 of the 78 participants (88%) said that access to proper tooling could decrease their reported estimated resumption time, highlighting the opportunity to explore systems that help programmers resume their tasks more effectively.

Summary of Findings: Contextual Inquiry & Online Survey

Our two formative studies suggest that programmers leverage mobile devices to make progress on software development tasks, but do not write code on-the-go. Their existing mobile work practices are primarily exploratory, while their desired work practices are grounded in existing code. Email is used as the primary mechanism to continue progress across devices - where captured thoughts and online research elements are transferred from the mobile device to the workstation via email. Programmers prepare for resumption by minimizing what they need to resume and use their mobile work practices to keep context alive while away from their workstation.

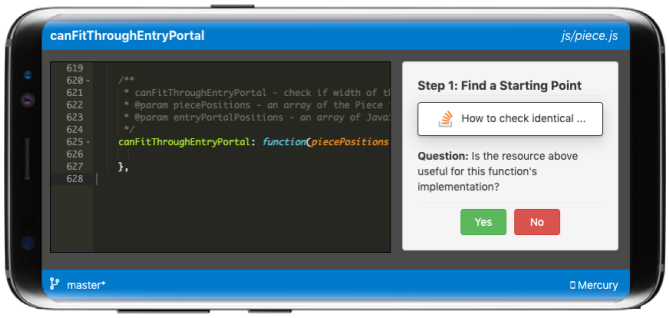
MERCURY, A MOBILE PROGRAMMING TOOL

Based on our findings from our formative studies, we designed and built *Mercury*, a microproductivity system integrated with Visual Studio Code (VSCode) that automatically generates mobile-friendly, short programming-related tasks, or *microtasks* [8], to support programmers' needs and desires for continuation. When a user decides to go mobile, Mercury uses the current state of their files to generate microtasks that can be routed to the user to make meaningful progress in their work while they are away from their workstation. Users access these microtasks from their mobile device using the Mercury mobile app (see Figure 5) and can complete their auto-generated microtasks at their own pace and leisure. Here, we detail Mercury's architecture and its approach to generating microtasks.

Microtask Generation

Mercury automatically generates microtasks based on the functions in users' source code. Functions are inherently compartmentalized to separate and scope source code, making them suitable candidates to surface in attention- and resource-constrained environments. Further, the use of function-based approaches is well-supported by prior research that has demonstrated its utility in crowdsourcing scenarios [35].

Mercury introduces two, novel selfsourcing microtasks based on the paradigms of mobile work identified in our formative studies: *exploratory microtasks* and *grounded microtasks*. To design these microtasks based on functions in users' source code, we leverage "The Function Design Recipe" [18], a six-step process used for teaching function design in software engineering curricula. Specifically, our microtasks are inspired by *function templating* (step 4), and *function testing* (step 6),



(a) Step 1 of Exploratory Microtasks ask the user to determine the relevance of a web resource for an unimplemented function.

(b) Step 2 of Exploratory Microtasks allow the user to add a brief note to add context to resources they find useful for a function's implementation.

(c) Step 1 of Grounded Microtasks ask the user to assess the behavior of a function with a particular set of function parameters.

(d) Step 2 of Grounded Microtasks allow the user to add a brief note to add context to a set of function parameters that cause the function to fail.

Figure 4: Mercury's microtasking interface supports two types of microtasks: (1) Exploratory Microtasks and (2) Grounded Microtasks. For each microtask, the interface shows the function's name (top-left), the function's origin file (top-right), the function's content (editor), and the microtask's task space (white block).

which correspond to preparing a function's implementation and reviewing a function's execution respectively.

Mercury's microtask generation procedure is powered by a custom source code parser that extracts each function's attributes, including location, name, parameters, body, and, if available, associated documentation. Importantly, the procedure relies on the presence of a function documentation string (docstring) in order to generate microtasks for a particular function. Mercury's parser was designed to specifically seek out docstrings in the JSDoc format, an industry standard already used by professional developers. We now detail the procedural aspects of generating Mercury's microtasks in depth.

Exploratory Microtasks

Exploratory microtasks (EMs) are two-step microtasks for functions with empty function bodies (e.g., function stubs). In the presence of such functions, Mercury first uses a regular expression to extract the function's description from its docstring. The description is then used as a query to Bing where the top-N web results from either a question-answering site (e.g., StackOverflow) or a documentation site (e.g., MSDN, MDN) will be converted into templated EM tasks. The first step of each EM asks users if the surfaced web resource is useful for the function's implementation. Users can tap on the resource to open the page in a modal window within Mercury's UI. Throughout this process, users have an opportunity to rate the utility of each resource (useful / not useful). Rating a resource initiates the second second step of the task, which asks users to optionally explain why the resource is useful. Upon submitting the response, the Mercury system injects the resource's URL and the user's note back into the associated function's docstring. As no convention exists for formatting URLs in source code, the resource URL was formatted to match the most commonly observed format in a recent large-scale analysis of hyperlinks in source code comments [21]. An example is shown in Figures 4a and 4b.

Grounded Microtasks

Grounded microtasks (GMs) are two-step microtasks that are generated for functions with content. When encountering such functions, Mercury will auto-generate GMs for a function by determining the type of its parameters and their purported use within the function, as documented by the function's docstring and signature. Using this information, Mercury generates a set of parameters specifically for this function to serve as a test case. Test cases are randomly selected from a list of common edge-cases, such as empty strings and null object references, per the Function Design Recipe. In the first step of each GM, users are asked to determine if the function will execute correctly with a given set of parameters (see Figure 4c). If the user indicates that the function will fail execution, they proceed to the second step of the task where they are allowed to optionally explain why the test case fails (see Figure 4d). Upon submission, Mercury injects the test-case and the optional explanation into the associated function's docstring.

Queuing, Sequencing, and Completing Microtasks

After generating microtasks, Mercury dynamically constructs a microtask queue for the user. Mercury's strategy for ordering microtask queues is based on principles of working memory

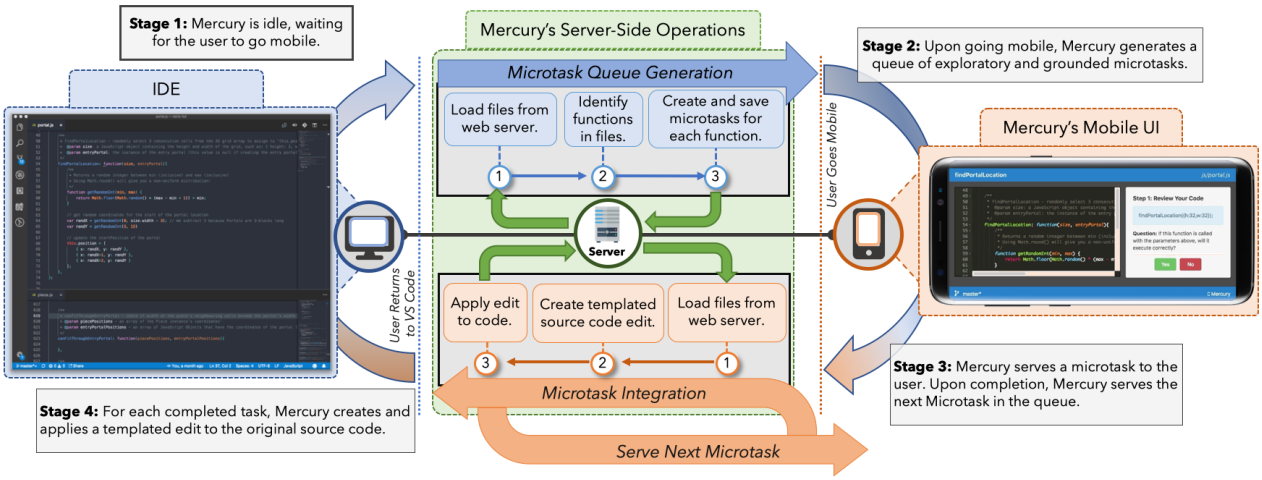


Figure 5: Mercury's architecture supports four stages of interaction.

[2]. While users are actively programming on their workstation, Mercury maintains a ranked list of functions ordered by the amount of time since being edited or seen for more than 10 seconds. When transitioning to a mobile device, Mercury uses this information to route a microtask associated with the function the user was most recently working on. Beyond the first task, Mercury uses a standard round-robin algorithm to distribute attention across the functions found in the user's workspace. Importantly, Mercury allows users complete their queued microtasks at their own pace and does not require users to exhaust their queue before returning to the workstation.

System Architecture

Mercury is composed of two primary sub-systems: 1) a MeteorJS web application that manages all web requests, serves the front-end mobile experience, and handles information synchronization between web clients and a Mongo NoSQL database; and 2) a VSCode plugin that converts the VSCode workspace into a web client that shares workspace state with the server.

Data and Synchronization Model

Mercury's data model is file-centric and based on the principles of file-based cloud storage. Upon starting VSCode, the plugin will authenticate with the user and immediately synchronize the editor's workspace files and directories with the server. Through the plugin, changes in the VSCode editor are immediately propagated and synchronized to the server and to Mercury. Similarly, any change made through Mercury's task interface will be propagated to the server and to VSCode. Alongside files, Mercury stores and synchronizes the user's mobile tasks, their state (i.e., whether or not they are at their workstation), and any interaction they have with the system.

USER STUDY: METHODS

Following established practices for evaluating cross-device systems [6] and tools to support software engineering [32], we designed and conducted a "first-use" study [20] to understand Mercury's successes and shortcomings as a tool for supporting programmers' mobile work practices.

Experimental Design

We conducted a lab study that was inspired by recent research that found developers regularly experience unplanned "short breaks" throughout their workday [40]. Specifically, these types of breaks can last upward of 15 minutes, and often yield scenarios in which individuals are forced to spend time away from their workstation. To better allow our lab study to speak to Mercury's practical utility, we adopted the temporal and unexpected nature of these breaks to frame our study design. The study required participants to work on a predefined programming task on a workstation, leave the workspace for 30 minutes with a mobile device and then return to the workstation to complete the task. The study lasted approximately 1 hour and 30 minutes and was split into three 30-minute phases:

Phase I: Starting the Task

After reading the task instructions, participants were told to work toward the implementation of the study's programming task for the next 30 minutes, and were told they would be given a mobile device to use "a new mobile experience for progammers" while they were away from the workstation. Participants worked uninterrupted during this 30-minute period.

Phase II: Going Mobile with Mercury

After 30 minutes participants were interrupted and told that they would now need to leave the room. They were given a Samsung S8 smartphone that had access to Mercury and was configured with their participant identifier to ensure synchronization. At the time of interruption one researcher administratively triggered Mercury's task generation function to simulate a seamless transition between devices. Participants were instructed to use Mercury's mobile experience during the next 30 minutes from the building's atrium and asked to return to the study room to continue their implementation after the the 30 minutes had passed.

Phase III: Returning to the Task

Upon returning to the study room, participants were told to place the smartphone face-down on their desk and continue working toward the implementation of the study's programming task. After 15 minutes of continued work, they were

told that the task was over; only one participant (P35) finished the task in this time. Participants were then given a post-study questionnaire and told that we would follow up within 24 hours to conduct a semi-structured post-study interview.

Programming Task

Participants were asked to complete a HTML5/CSS/JavaScript implementation of an enhanced version of Tetris that introduced *portals*, following prior research that has used classic arcade games as an implementation task in studies [47]. In this version, an entry portal and a corresponding exit portal automatically spawn on the Tetris game grid. When a game piece is adjacent to the entry portal, the piece's next move should transfer the adjacent pieces to the exit portal's location.

As the study task, participants were given four functions to implement in the Tetris codebase, three of which focused on portal validation and one of which focused on locating portals on the grid. All four function implementations were blank at the start of the study. If all four functions were correctly implemented, both portals would function correctly. To facilitate Mercury's integration with the codebase, all functions in the source code were documented with the JSDoc standard. Pilots of our study confirmed that the task was challenging, yet feasible. Participants were given five minutes to read through the task instructions before being allowed to begin the task.

Mercury was configured to create five microtasks for each of the task's four functions, totaling in a queue of 20 microtasks for each participant. The type of microtasks generated for each function were contingent on its "completeness". We used the number of lines in a function's body at the time of going mobile as a proxy. Grounded microtasks were created for functions whose body included more than five lines of code. Otherwise, Mercury recognized the function as incomplete, and would create exploratory microtasks for the function.

Data Collection

We collected the following data as a part of the study:

Pre-Study Questionnaire

We inquired about participants' gender, job role, and experience. We also inquired about the practices from Table 1.

Instrumentation Data

We tracked participants' actions with screen capture software, and by logging low-level events within Mercury.

Post-Study Questionnaire

Before concluding the study, we administered a questionnaire that included three validated instruments: 1) the System Usability Scale (SUS) [5] to measure Mercury's usability, 2) a 5-point reattachment questionnaire for measuring participants' ability to mentally reengage with the task [52], and 3) a 5-point PANAS-inspired scale to measure how productive, engaged, and relaxed the participant felt while they were away [62, 63].

Post-Study Interview

We conducted a 20-minute semi-structured post-study interview with each participant. The interview began by asking participants about the experience in general alongside the utility of each microtask, and transitioned into Mercury's effect on participants' ability to return to the task. Interviews concluded by inquiring about Mercury's practical utility.

Participants

20 participants (18 male / two female) were recruited by randomly sampling the same company-wide employee list of individuals with programming job roles used in both the Contextual Inquiry and the Online Survey. Job roles of those recruited include software engineer (18) and software engineering intern (2). Participants' ages included 18-24 (3), 25-34 (7), 35-44 (9), 45-55 (1), and participants' years of experience included 3-5 years (4), 5-10 years (6), and 10 or more years (10). Participants were compensated with a $50 gift card.

USER STUDY: FINDINGS

Overall, our user study results highlight how Mercury enhances mobile programming experiences. Participants found value in Mercury's interface and were able to make meaningful progress with little effort or attention. Mercury supported continuation of tasks across devices with seamless transfer of task progress, and interacting with Mercury's microtasks enabled participants to easily resume coding upon returning to their workstation. The utility of Mercury's tasks understandably varied between individual's and their unique contexts. We discuss themes from our user study and evaluation below.

Supporting Mobile Work Practices

Most participants (17 out of 20) enjoyed Mercury's microproductivity-inspired task design as it required "little attention to make progress" (P9). Participants' post-study questionnaire responses indicate that the experience allowed them to feel productive (M=3.8; SD=0.9), engaged (M=3.8; SD=0.7), and relaxed (M=4.1; SD=0.9) while mobile. The positive reception is also supported by Mercury's favorable SUS scores (M=77.5; IQR=11.8). Only two participants voiced complaints related to device constraints, stating that "the device made it difficult to read code" (P18) and that "the experience suffered from the same pitfalls as any mobile development environment" (P16). On average, participants used Mercury to complete 17 microtasks during the study. The average time per microtask was 74 seconds. No significant difference was observed between Mercury's microtask types.

Exploratory and Grounded Microtasks

The exploratory (EMs) and grounded (GMs) microtasks received positive feedback from participants, with four participants finding both to be useful, six participants liking GMs better, and seven preferring EMs instead. On average, participants identified 60% of the web resources from Exploratory Microtasks as relevant, and indicated 90% of the test-cases from Grounded Microtasks identified issues they could correct upon returning to their workspace. Only three participants found neither to be particularly helpful, yet the premise of the system was still seen as promising and beneficial:

> "*Both tasks are great ideas. They're great first-steps toward being able to mobilize myself in a new way.*" (P16)

Participants evaluated the utility of EMs on one primary characteristic: *resource relevance*. As with any online search, participants found "some references applicable and useful, but toward the end, they seemed less relevant" (P2).

Participants who found EMs useful (14 of 20) expressed sympathy for the relevance problem, highlighting that "I'd be

seeing the same noise if I did my own search" (P4) and anything more accurate would "win us the Nobel prize". The noise was not always bad: four participants recounted how EMs reoriented their understanding of a problem they were stuck on, thanks to a surprising resource.

> *"One online research task made me realize the implementation was just an array intersection problem. It kept it in my head, especially when it framed the problem for me. I knew exactly what I was going to do when I got back."* (P15)

This particular participant's experience further highlights the importance of relevance for online research as it suggests surfacing the right resource may stoke individuals' resumption. In addition to accessing online resources, six participants suggested adding support for team communication channels to leverage the expertise of teammates in various scenarios.

For GMs, participants were excited about the ability to reexamine their code in a different setting. Their appreciation of GMs were centered on the task's ability to "introduce edge cases that I didn't even think of while coding" (P15). The few participants that did not find GMs useful described the automatically generated test cases as "too simple" (P14) or "repetitive after a point" (P17). However, participants' remarks were clear that the tasks would have been useful had they surfaced test cases "of the right complexity" (P9). Eighteen participants offered explicit accounts of how GMs could be situated in their current work practice within their team:

> *"The ability to pull in reviewers and use canned comments, add voice commentary, highlight code, and look at diffs on-the-go. I think that would be a huge thing." (P4)*

Participants also noted the ease of completing these tasks using Mercury. They even suggested that Mercury could improve systems that already support mobile code review in some form (e.g., Visual Studio Team Services), to be more user-friendly: "Mercury created a mobile experience that would be generally easier and more enjoyable to use in a team setting" (P3).

The three participants who found neither GMs or EMs to be useful noted that their issue was with the specific tasks they saw, but expressed interest in an experience that would have helped them "start with algorithm design" (P10), "sketch or focus on something design-related" (P13), or "refresh my mind with creative ideas" (P1).

When Mercury Would Be Used

After using Mercury, participants had no shortage of imagining how the system could fit into their daily work practice:

> *"With Mercury, I can step away from the terminal and take a break, have a coffee, go outside. I'm not tethered to the desk as much as I would be, and I can still accomplish meaningful work."* (P4)

Participants voiced excitement in using Mercury to continue their work "when you want to productive" (P14), "when you don't really need to pay attention" (P12), and "when you have nothing better to do" (P2). 18 participants noted commutes in public transport as a key setting for continuing work:

> *"I've got 45 minutes to kill on the bus each way between home and work. If I'm still thinking about some work, the end of the workday would be great if I can eke out some additional productivity on the way home."* (P16)

Discussed by 16 participants, the second most common settings cited were brief moments that involve waiting in the workplace, such as waiting for a meeting to start, waiting in line to order lunch, and even bathroom breaks. Similarly, settings that involve waiting outside of work, such as doctors' offices, were also mentioned by participants.

Participants had mixed feelings about how Mercury might affect their work-life balance. Six expressed an interest in using Mercury as a means for capturing lingering thoughts that stem from their workday.

> *"Sometimes, you come home, and you're still attached to work. Your kids (are) trying to play with you. If Mercury is easy enough to capture a thought to let me give my kids the attention they need, I'd be excited to use it then."* (P12)

The other five bolstered the need to simply capture a quick thought as a result of the right thought coming at the wrong time (e.g., "while I'm brushing my teeth" (P11)). Conversely, three participants voiced a concern of "working 24/7." (P1). We expand on this theme later, in the Discussion section.

Supporting Cross-Device Continuation

All 20 participants liked being able to continue their work while away from the study's workstation, and, in particular, appreciated how Mercury helped them transfer information:

> *"Getting information between devices is usually the problem. Mercury kind-of helps this by handling the synchronization."* (P16)

Participants liked having mobile access to code that had recently been written on the desktop and being able to synchronize information across devices without having to remember to do particular actions (e.g., a repository commit).

Mercury's guided nature was a thematic point of discussion for each participant. Five participants expressed satisfaction with the guided aspect of Mercury's mobile experience:

> *"It was nice because I felt like it was intuitively looking for things I probably would've looked for anyway."* (P20)

Other comments in support of a fully autonomous process described Mercury's process as one that "was nice to supply guideposts", "required little input" (P16), "gamified because you didn't know what was coming next" (P15).

Participants expressed appreciation for Mercury's guided nature and ready-made, on-the-go tasks, and provided recommendations for how these could be improved with personalization:

> *"There'd be times when I want the system to autopilot me. Other times, I'm a control freak, and I want to be able to say, 'Now is the time I do this'."* (P4)

They suggested thematic pathways that would make the mobile experience more useful for them both during the study and in practice, such as the ability to tell Mercury which function to focus on while mobile, the ability to "mark a function to view directly on Mercury" (P17), and support for task navigation (e.g., skipping and revisiting). Overall, participants found it

easy to envision how Mercury could be a part of their usual work routine, and were excited to offer feedback that could help shape the system further.

Supporting Task Resumption

Reattachment questionnaire reports suggest Mercury positively affected participants' resumption processes (mean = 15.5; IQR = 3.4). 16 participants offered positive accounts of how Mercury helped them resume the Tetris task when they returned to the study workstation. When participants recounted their experience with Mercury in the post-study interview they said the experience "helped keep things available" (P9), "kept your mental process warm" (P14), and "felt like it greased my mind's wheels" (P11). In discussing how participants imagined the system's ability to help with resumption in unexpected scenarios, 11 participants highlighted that it would add comfort if participants needed to leave unexpectedly:

> *"Mercury would make me feel more comfortable if I need to walk away momentarily and come back. It would help bring down the ramp-up time time when I get back to my workstation, and I can just go and code right away."* (P12)

Five participants specifically stated their resumption with the Tetris task was facilitated not only by being able to continue the work on-the-go, but knowing the first step they would take when they returned to the study workstation. These statements were corroborated by their screen recordings in which we observed each participant referencing a source-code change made by Mercury upon their return and subsequently acting on it (e.g., copy-and-pasting a resource URL into their browser).

DISCUSSION

Our study provides insight into understanding the role of mobile programming tools in practice. Prior research targets how mobile programming can be enhanced with novel touch-based interfaces for the cumbersome nature of text entry on mobile phones [56, 57], and cross-device techniques for supporting programmers across multiple mobile devices while stationary [22]. Here, we find that a mobile work experience designed around microproductivity can not only help programmers continue their work on-the-go, but also instill comfort in pausing work unexpectedly. We also see that programmers feel like they can make meaningful progress in their work with Mercury's microtasking experience in scenarios ranging from brief moments of downtime to the daily commute. Further, we observe that engaging with programming-related tasks via Mercury spurs users' ability to resume their work.

Mercury's microtask designs were driven by the mobile tasking needs and desires observed in our formative studies. An ideal microtask is contextually self-contained, requires little effort to complete, and helps people make progress [23], and Mercury's microtasks were designed with these principles in mind. However, we find the utility of Mercury's microtasks is firmly grounded in the programmer's work context. For example, a small number of users expressed a desire for design-oriented microtasks. While we explored only two types of microtasks in our exploration of microtasked programming, a framework like Mercury allows us to design and test different experiences, providing an important first step toward empowering programmers' with microproductivity in the wild.

Our research suggests that Mercury helped kindle participants' resumption processes. In our user study, we find that giving our participants the ability to mobilize their work on-demand helps them feel "not as tethered to the desk as much as they would be" (P4). Understanding how programmers' behavior changes with this newfound comfort in moving away from their workstation is an important direction of future work. Similarly, the findings from our user study establish a frontier of future research aimed at exploring the intersection of prior and current interventions (e.g., visual cues [47, 49] for cross-device experiences) in support of programmers' productivity.

Unlike our assessment of Mercury, the majority of microtask programming research has been studied in the context of teams of "transient" developers [36, 35]. Several participants in our user study noted Mercury's potential value in team settings, while others were unsure of its ecological utility for teams with diverse information needs [31]. Exploring how social experiences and larger codebases change the utility of Mercury's mobile experience is a key direction of future research.

By enabling programmers to work from their mobile devices during free micromoments, Mercury has the potential to blur the lines between work and non-work time. More than half of our user study's participants expressed an interest in using Mercury outside of the workplace. While the overarching goal of our work is to empower programmers' mobile work practices, we recognize the threat that a mobile microproductivity system like Mercury may pose on encroaching into individuals' downtime. However, we also see how participants were able to interleave Mercury tasks with other activities. As one participant notes:

> *"When I was downstairs in the atrium, I actually felt like was still making progress even though I wasn't really paying attention."* (P12)

An important area of future work should focus on how to design software tools for task continuation support that also account for programmers' need to disconnect from work [63].

Limitations

While our study provides insight into cross-device programming support, there are a number of limitations that require further study. Mercury's user study was conducted in a lab setting. Prior research reinforces lab studies as valuable approaches to study novel systems, specifically those have cross-device components that may be challenging to reliably study in-the-wild [6]. While our lab study's design was strongly grounded in observations made in the field, further studies are needed to claim that the same observations may be seen in-the-wild or consistently over time.

Our study's evaluation of Mercury was focused on understanding the success and challenges of using a microtask programming solution for on-the-go programming. Our evaluation does not compare microproductivity tools to non-microproductivity tools for on-the-go work (e.g., CodeBeat[1]), and we make no claim about how the effectiveness of these tools may differ. However, we recognize this as a valuable area of future research both for Mercury and future programming tools that incorporate elements of microproductivity.

[1] https://codebeat.co/

Finally, our study's population consisted primarily of professional and experienced software engineers at a large technology corporation. Mercury may provide different experiences for individuals that program less frequently in their job roles or work at smaller companies. Mercury's mobile experience also relies on the presence of function documentation. We recognize that documentation practices may vary among professionals and that self-documenting code is not only common, but often promoted [53]. Future research is needed to understand how Mercury can be adapted to scenarios where documentation is significantly limited or unavailable entirely.

CONCLUSION

In this paper, we presented Mercury, a system that guides programmers in making progress on-the-go with auto-generated microtasks based on their source code's current state. We detailed how the findings from our two studies – contextual inquiry and online survey – motivated Mercury's design as a microtasking system for on-the-go programming work. In studying Mercury with 20 full-time programmers, we found that mobile work experiences designed around microproductivity can help programmers continue their work on-the-go and instill comfort in pausing work unexpectedly, all the while making meaningful progress on their work tasks. Mercury's success serves as a first step in a family of future software engineering tools that allow programmers to make progress on their work away from their primary workstation.

REFERENCES

[1] Erik M. Altmann and J. Gregory Trafton. 2002. Memory for goals: An activation-based model. *Cognitive Science* 26, 1 (2 2002), 39–83. DOI: http://dx.doi.org/10.1016/S0364-0213(01)00058-1

[2] Alan Baddeley. 2000. The episodic buffer: a new component of working memory? *Trends in Cognitive Sciences* 4, 11 (11 2000), 417–423. DOI: http://dx.doi.org/10.1016/S1364-6613(00)01538-2

[3] Patti Bao, Jeffrey Pierce, Stephen Whittaker, and Shumin Zhai. 2011. Smart phone use by non-mobile business users. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11*. ACM Press, New York, New York, USA, 445. DOI: http://dx.doi.org/10.1145/2037373.2037440

[4] Michael S. Bernstein, Greg Little, Robert C. Miller, Bjorn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. Soylent: A Word Processor with a Crowd Inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology - UIST '10*. ACM Press, New York, New York, USA, 313. DOI: http://dx.doi.org/10.1145/1866029.1866078

[5] John Brooke. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[6] Frederik Brudy, Christian Holz, Roman Radle, Chi-Jui Wu, Steven Houben, Clemens Klokmose, and Nicolai Marquardt. 2019. Cross-Device Taxonomy: Survey, Opportunities and Challenges of Interactions Spanning Across Multiple Devices. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019)*. ACM Press, New York, New York, USA. DOI: http://dx.doi.org/10.1145/3290605.3300792

[7] Carrie J. Cai, Shamsi T. Iqbal, and Jaime Teevan. 2016. Chain Reactions: The Impact of Order on Microtask Chains. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*. ACM Press, New York, New York, USA, 3143–3154. DOI: http://dx.doi.org/10.1145/2858036.2858237

[8] Justin Cheng, Jaime Teevan, Shamsi T. Iqbal, and Michael S. Bernstein. 2015. Break It Down: A Comparison of Macro- and Microtasks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*. ACM Press, New York, New York, USA, 4061–4064. DOI: http://dx.doi.org/10.1145/2702123.2702146

[9] Pei-Yu (Peggy) Chi and Yang Li. 2015. Weave: Scripting Cross-Device Wearable Interaction. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*. ACM Press, New York, New York, USA, 3923–3932. DOI: http://dx.doi.org/10.1145/2702123.2702451

[10] Lydia B. Chilton, Greg Little, Darren Edge, Daniel S. Weld, and James A. Landay. 2013. Cascade: Crowdsourcing Taxonomy Creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. ACM Press, New York, New York, USA, 1999. DOI: http://dx.doi.org/10.1145/2470654.2466265

[11] Evans Data Corporation. 2018. Global Developer Population and Demographic Study 2018, Volume 2. *Evans Data Report* (2018).

[12] Justin Cranshaw, Emad Elwany, Todd Newman, Rafal Kocielnik, Bowen Yu, Sandeep Soni, Jaime Teevan, and Andres Monroy-Hernández. 2017. Calendar.help: Designing a Workflow-Based Scheduling Agent with Humans in the Loop. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 2382–2393. DOI: http://dx.doi.org/10.1145/3025453.3025780

[13] Mary Czerwinski, Eric Horvitz, and Susan Wilhite. 2004. A diary study of task switching and interruptions. In *Proceedings of the 2004 conference on Human factors in computing systems - CHI '04*. ACM Press, New York, New York, USA, 175–182. DOI: http://dx.doi.org/10.1145/985692.985715

[14] Fred D. Davis. 1989. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly* 13, 3 (9 1989), 319. DOI: http://dx.doi.org/10.2307/249008

[15] David Dearman and Jeffery S. Pierce. 2008. It's on my other computer!: computing with multiple devices. In *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*. ACM Press, New York, New York, USA, 767. DOI: http://dx.doi.org/10.1145/1357054.1357177

[16] Linda Di Geronimo, Maria Husmann, and Moira C. Norrie. 2016. Surveying personal device ecosystems with cross-device applications in mind. In *Proceedings of the 5th ACM International Symposium on Pervasive Displays - PerDis '16*. ACM Press, New York, New York, USA, 220–227. DOI: http://dx.doi.org/10.1145/2914920.2915028

[17] Tao Dong, Elizabeth F. Churchill, and Jeffrey Nichols. 2016. Understanding the Challenges of Designing and Developing Multi-Device Experiences. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems - DIS '16*. ACM Press, New York, New York, USA, 62–72. DOI: http://dx.doi.org/10.1145/2901790.2901851

[18] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. 2001. How to Design Programs: An Introduction to Programming an d Computing. (2001).

[19] Ujwal Gadiraju and Stefan Dietze. 2017. Improving learning through achievement priming in crowdsourced information finding microtasks. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference on - LAK '17*. ACM Press, New York, New York, USA, 105–114. DOI: http://dx.doi.org/10.1145/3027385.3027402

[20] Bjorn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. 2007. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*. ACM Press, New York, New York, USA, 145. DOI:http://dx.doi.org/10.1145/1240624.1240646

[21] Hideaki Hata, Christoph Treude, Raula Gaikovina Kula, and Takashi Ishio. 2019. 9.6 Million Links in Source Code Comments: Purpose, Evolution, and Decay. *CoRR* abs/1901.0 (2019). http://arxiv.org/abs/1901.07440

[22] Maria Husmann, Alfonso Murolo, Nicolas Kick, Linda Di Geronimo, and Moira C. Norrie. 2018. Supporting out of office software development using personal devices. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI '18*. ACM Press, New York, New York, USA, 1–11. DOI: http://dx.doi.org/10.1145/3229434.3229454

[23] Shamsi Iqbal, Jaime Teevan, Dan Liebling, and Anne Loomis Thompson. 2018. Multitasking with Play Write, a Mobile Microproductivity Writing Tool (Forthcoming). In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*.

[24] Tero Jokela, Jarno Ojala, and Thomas Olsson. 2015. A Diary Study on Combining Multiple Information Devices in Everyday Activities and Tasks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*. ACM Press, New York, New York, USA, 3903–3912. DOI: http://dx.doi.org/10.1145/2702123.2702211

[25] Shaun K. Kane, Amy K. Karlson, Brian R. Meyers, Paul Johns, Andy Jacobs, and Greg Smith. 2009. Exploring Cross-Device Web Use on PCs and Mobile Devices. 722–735. DOI: http://dx.doi.org/10.1007/978-3-642-03655-2{_}79

[26] Amy K. Karlson, Shamsi T. Iqbal, Brian Meyers, Gonzalo Ramos, Kathy Lee, and John C. Tang. 2010. Mobile taskflow in context: a screenshot study of smartphone usage. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, New York, New York, USA, 2009. DOI: http://dx.doi.org/10.1145/1753326.1753631

[27] Amy K. Karlson, Brian R. Meyers, Andy Jacobs, Paul Johns, and Shaun K. Kane. 2009. Working Overtime: Patterns of Smartphone and PC Usage in the Day of an Information Worker. 398–405. DOI: http://dx.doi.org/10.1007/978-3-642-01516-8{_}27

[28] Harmanpreet Kaur, Alex C. Williams, Anne Loomis Thompson, Walter S. Lasecki, Shamsi T. Iqbal, and Jaime Teevan. 2018. Creating Better Action Plans for Writing Tasks via Vocabulary-Based Planning. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (11 2018), 1–22. DOI: http://dx.doi.org/10.1145/3274355

[29] Mik Kersten and Gail C. Murphy. 2006. 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*. ACM Press, New York, New York, USA, 1. DOI: http://dx.doi.org/10.1145/1181775.1181777

[30] A Kittur, J Nickerson, and M Bernstein. 2013. The Future of Crowd Work. *Proc. CSCW '13* (2013), 1–17. DOI:http://dx.doi.org/10.1145/2441776.2441923

[31] Andrew J. Ko, Robert DeLine, and Gina Venolia. 2007. Information Needs in Collocated Software Development Teams. In *29th International Conference on Software Engineering (ICSE'07)*. IEEE, 344–353. DOI: http://dx.doi.org/10.1109/ICSE.2007.45

[32] Andrew J. Ko, Thomas D. LaToza, and Margaret M. Burnett. 2015. A practical guide to controlled experiments of software engineering tools with human participants. *Empirical Software Engineering* 20, 1 (2 2015), 110–141. DOI: http://dx.doi.org/10.1007/s10664-013-9279-3

[33] Thomas D. LaToza, W. Ben Towne, Andre van der Hoek, and James D. Herbsleb. 2013. Crowd development. In *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 85–88. DOI: http://dx.doi.org/10.1109/CHASE.2013.6614737

[34] Thomas D. LaToza, Arturo Di Lecce, Fabio Ricci, W. Ben Towne, and Andre van der Hoek. 2015. Ask the crowd: Scaffolding coordination and knowledge sharing in microtask programming. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing*

(VL/HCC). IEEE, 23–27. DOI: http://dx.doi.org/10.1109/VLHCC.2015.7357194

[35] Thomas D. LaToza, Arturo Di Lecce, Fabio Ricci, W. Ben Towne, and Andre Van der Hoek. 2018. Microtask Programming. *IEEE Transactions on Software Engineering* (2018), 1–1. DOI: http://dx.doi.org/10.1109/TSE.2018.2823327

[36] Thomas D. LaToza, W. Ben Towne, Christian M. Adriano, and AndrÃl' van der Hoek. 2014. Microtask programming: building software with a crowd. In *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14*. ACM Press, New York, New York, USA, 43–54. DOI: http://dx.doi.org/10.1145/2642918.2647349

[37] Thomas D. LaToza and Andre van der Hoek. 2016. Crowdsourcing in Software Engineering: Models, Motivations, and Challenges. *IEEE Software* 33, 1 (1 2016), 74–80. DOI: http://dx.doi.org/10.1109/MS.2016.12

[38] Thomas D. LaToza, Gina Venolia, and Robert DeLine. 2006. Maintaining mental models: a study of developer work habits. In *Proceeding of the 28th international conference on Software engineering - ICSE '06*. ACM Press, New York, New York, USA, 492. DOI: http://dx.doi.org/10.1145/1134285.1134355

[39] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. 2017. *Research methods in human-computer interaction*. Morgan Kaufmann.

[40] Andre N. Meyer, Laura E. Barton, Gail C. Murphy, Thomas Zimmermann, and Thomas Fritz. 2017. The Work Life of Developers: Activities, Switches and Perceived Productivity. *IEEE Transactions on Software Engineering* 43, 12 (12 2017), 1178–1193. DOI: http://dx.doi.org/10.1109/TSE.2017.2656886

[41] Michael Nebeling. 2017. XDBrowser 2.0: Semi-Automatic Generation of Cross-Device Interfaces. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 4574–4584. DOI: http://dx.doi.org/10.1145/3025453.3025547

[42] Michael Nebeling and Anind K. Dey. 2016. XDBrowser: User-Defined Cross-Device Web Page Designs. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*. ACM Press, New York, New York, USA, 5494–5505. DOI: http://dx.doi.org/10.1145/2858036.2858048

[43] Michael Nebeling, Alexandra To, Anhong Guo, Adrian A. de Freitas, Jaime Teevan, Steven P. Dow, and Jeffrey P. Bigham. 2016. WearWrite: Crowd-Assisted Writing from Smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*. ACM Press, New York, New York, USA, 3834–3846. DOI: http://dx.doi.org/10.1145/2858036.2858169

[44] Tuan Anh Nguyen, Christoph Csallner, and Nikolai Tillmann. 2013. GROPG: A graphical on-phone debugger. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 1189–1192. DOI: http://dx.doi.org/10.1109/ICSE.2013.6606675

[45] Katie O'Leary, Tao Dong, Julia Katherine Haines, Michael Gilbert, Elizabeth F. Churchill, and Jeffrey Nichols. 2017. The Moving Context Kit: Designing for Context Shifts in Multi-Device Experiences. In *Proceedings of the 2017 Conference on Designing Interactive Systems - DIS '17*. ACM Press, New York, New York, USA, 309–320. DOI: http://dx.doi.org/10.1145/3064663.3064768

[46] Antti Oulasvirta and Lauri Sumari. 2007. Mobile kits and laptop trays: managing multiple devices in mobile information work. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*. ACM Press, New York, New York, USA, 1127. DOI:http://dx.doi.org/10.1145/1240624.1240795

[47] Chris Parnin and Robert DeLine. 2010. Evaluating cues for resuming interrupted programming tasks. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, New York, New York, USA, 93. DOI: http://dx.doi.org/10.1145/1753326.1753342

[48] C. Parnin and C. Gorg. Building Usage Contexts During Program Comprehension. In *14th IEEE International Conference on Program Comprehension (ICPC'06)*. IEEE, 13–22. DOI: http://dx.doi.org/10.1109/ICPC.2006.14

[49] Chris Parnin and Spencer Rugaber. 2011. Resumption strategies for interrupted programming tasks. *Software Quality Journal* 19, 1 (5 2011), 5–34. DOI: http://dx.doi.org/10.1007/s11219-010-9104-9

[50] Niloufar Salehi, Jaime Teevan, Shamsi Iqbal, and Ece Kamar. 2017. Communicating Context to the Crowd for Complex Writing Tasks. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing - CSCW '17*. ACM Press, New York, New York, USA, 1890–1901. DOI: http://dx.doi.org/10.1145/2998181.2998332

[51] Stephanie Santosa and Daniel Wigdor. 2013. A field study of multi-device workflows in distributed workspaces. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing - UbiComp '13*. ACM Press, New York, New York, USA, 63. DOI: http://dx.doi.org/10.1145/2493432.2493476

[52] Sabine Sonnentag and Jana Kühnel. 2016. Coming back to work in the morning: Psychological detachment and reattachment as predictors of work engagement. *Journal of Occupational Health Psychology* 21, 4 (2016), 379–390. DOI:http://dx.doi.org/10.1037/ocp0000020

[53] Diomidis Spinellis. 2010. Code Documentation. *IEEE Software* 27, 4 (7 2010), 18–19. DOI: http://dx.doi.org/10.1109/MS.2010.95

[54] Jaime Teevan, Shamsi T. Iqbal, and Curtis von Veh. 2016. Supporting Collaborative Writing with Microtasks.

In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*. ACM Press, New York, New York, USA, 2657–2668. DOI: http://dx.doi.org/10.1145/2858036.2858108

[55] Jaime Teevan, Daniel J. Liebling, and Walter S. Lasecki. 2014. Selfsourcing personal tasks. In *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems - CHI EA '14*. ACM Press, New York, New York, USA, 2527–2532. DOI:http://dx.doi.org/10.1145/2559206.2581181

[56] Nikolai Tillmann, Michal Moskal, Jonathan de Halleux, and Manuel Fahndrich. 2011. TouchDevelop: programming cloud-connected mobile devices via touchscreen. In *Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software - ONWARD '11*. ACM Press, New York, New York, USA, 49. DOI: http://dx.doi.org/10.1145/2048237.2048245

[57] Nikolai Tillmann, Michal Moskal, Jonathan de Halleux, Manuel Fahndrich, and Sebastian Burckhardt. 2012. TouchDevelop: app development on mobile devices. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering - FSE '12*. ACM Press, New York, New York, USA, 1. DOI:http://dx.doi.org/10.1145/2393596.2393641

[58] J. Gregory Trafton, Erik M. Altmann, Derek P. Brock, and Farilee E. Mintz. 2003. Preparing to resume an interrupted task: Effects of prospective goal encoding and retrospective rehearsal. *International Journal of Human Computer Studies* 58, 5 (5 2003), 583–603. DOI: http://dx.doi.org/10.1016/S1071-5819(03)00023-5

[59] Rajan Vaish, Keith Wyngarden, Jingshu Chen, Brandon Cheung, and Michael S. Bernstein. 2014. Twitch Crowdsourcing: Crowd Contributions in Short Bursts of Time. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, New York, New York, USA, 3645–3654. DOI: http://dx.doi.org/10.1145/2556288.2556996

[60] Melissa A. Valentine, Daniela Retelny, Alexandra To, Negar Rahmati, Tulsee Doshi, and Michael S. Bernstein. 2017. Flash Organizations: Crowdsourcing Complex Work by Structuring Crowds As Organizations. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 3523–3537. DOI: http://dx.doi.org/10.1145/3025453.3025811

[61] R. van Solingen, E. Berghout, and F. van Latum. 1998. Interrupts: just a minute never is. *IEEE Software* 15, 5 (1998), 97–103. DOI: http://dx.doi.org/10.1109/52.714843

[62] David Watson, Lee Anna Clark, and Auke Tellegen. 1988. Development and validation of brief measures of positive and negative affect: The PANAS scales. *Journal of Personality and Social Psychology* 54, 6 (1988), 1063–1070. DOI: http://dx.doi.org/10.1037/0022-3514.54.6.1063

[63] Alex C. Williams, Harmanpreet Kaur, Gloria Mark, Anne Loomis Thompson, Shamsi T. Iqbal, and Jaime Teevan. 2018. Supporting Workplace Detachment and Reattachment with Conversational Intelligence. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, New York, New York, USA, 1–13. DOI: http://dx.doi.org/10.1145/3173574.3173662

[64] Dennis Wixon, Karen Holtzblatt, and Stephen Knox. 1990. Contextual design: an emergent view of system design. In *Proceedings of the SIGCHI conference on Human factors in computing systems Empowering people - CHI '90*. ACM Press, New York, New York, USA, 329–336. DOI: http://dx.doi.org/10.1145/97243.97304