Towards Software Development Microtasks

Yan Chen
School of Information
University of Michigan, Ann Arbor
yanchenm@umich.edu

Steve Oney
School of Information
University of Michigan, Ann Arbor
soney@umich.edu

Walter S. Lasecki
Computer Science & Engineering
and School of Information
University of Michigan, Ann Arbor
wlasecki@umich.edu

Abstract

Software development is a large, complex task that is difficult to break down into simple pieces that can be efficiently integrated into a single codebase. Developers often decompose programing tasks into classes, functions, and other constructs based on their functionality. However, solving programming tasks often requires adequate contextual information and efficient ways to aggregate the responses. In this paper, we discuss ongoing work on a system that helps developers easily (e.g. informally saying it) generate well-defined programming microtasks by enabling multimodal interactions. These tasks are flexible, schedulable, and their solutions can be efficiently integrated into the larger codebase.

Capturing Developers' Context

In prior studies, we have found that programming tasks generated during the software development process require significant added contextual information to be understood [1]. Previous work also suggested that creating tasks with enough context captured can decrease the time spent on recovery from interruption [2, 3]. Ideally, recovery time should be reduced as much as possible. This leads to our question: *How can a system automatically capture or generate task context that will minimize developers' effort?*

Automatically capturing task context will enable developers to easily define tasks that either crowds or they themselves can efficiently complete in the future. While developers have understanding that the system does not, it is inefficient for developers to spend significant time creating the most recoverable tasks. Understanding the trade off between these two processes could better help us create systems to facilitate development.

With sufficient context, either developers or crowds can complete a task, which adds more possibilities in how we allocate workforces. When a new task is created, developers can continue their current task and let the crowds help with the new task. This benefits overall productivity and keeps developers' workflows from being interrupted. If developers choose to work on a task themselves, they can quickly choose one that fits into their available time (e.g., a task that is completable before an upcoming meeting), and quickly rehydrate the task setting and context. They can then build that piece and move onto another task they have defined for themselves when they are next available. This has the potential to directly impact how developers define and complete large tasks that require multiple threads of reasoning and development, making projects more efficient and enabling more efficient context-switching and task scheduling.

Additionally, enabling responses written in different ways will support answers to different types of programming tasks. Respondents can comment, explain, or edit the code by their preferences just as how they interact with Google Docs and StackOverflow. Developers could choose to read the text responses and highlighted code for explanation and contextual reference or simply allow the responses merge to their codebases automatically without reimplementing them. This can mitigate the effort of aggregating microtasks and balance different preference of response formats that developers may have.

Ongoing Work

Our ongoing work explores how to build a system that smooths the process of assigning, rehydrating, and aggregating programming microtasks so that developers can create tasks on the fly that will be addressed when they (or a helper in the crowd) have sufficient time. The system will allow developers to create small programming tasks within their editors using natural language and automatically-captured code context.

The framework that we will create will also help to automate the microtasking process as it becomes possible to hand off either task creation or completion to machines, making software development more efficient.

Biography

Yan Chen is a 2nd year Ph.D student at the University of Michigan School of Information (UMSI). He is currently building intelligent systems for developers. In his research, he explores ways to better support developers using on-demand crowds. Steve Oney is an Assistant Professor at the UMSI. Walter S. Lasecki is an Assistant Professor of Computer Science & Engineering at Michigan.

References

[1] Y. Chen, S. Oney, and W. S. Lasecki. Towards providing on-demand expert support for software developers. In *CHI*, 2016.

[2] P. J. Guo and M. Seltzer. Burrito: Wrapping your lab notebook in computational infrastructure. In *TaPP*, 2012.

[3] J. Teevan, D. J. Liebling, and W. S. Lasecki. Selfsourcing personal tasks. In *CHI'14*, 2014.