RESEARCH OVERVIEW

Programming combines creativity and logical decisions expressed through carefully designed semantics. These decisions, semantics, and creative freedom depend on the problem the programmer is solving. Whether an engineer is programming to build a software, a data scientist programming to find trends in massive data dumps, or the end user programming in Excel to manipulate data, tools and platforms play a crucial role in determining the burden and cost programmers must pay during this process. But how well do these programming tools align with the human cognitive process? And how much do programmers adapt themselves to these tools? Our understanding of the cognitive processes can reduce the programmer’s battle with the tools and give them the creative freedom to focus on what they do the best – create intelligent systems and enhance scientific understanding.

My research is centered around designing adaptive and human centered tools and interfaces that align with the human cognitive processes when solving problems. I apply principles and foundational theories about human psychology and cognition to design better tools by asking questions like how do programmers maintain information when interacting with tools and interfaces? How does the cognitive processes like biases, intentions, and memory affect problem solving ability? To answer such questions, I conduct empirical investigations grounded in theories of human behavior and cognition. My work applies qualitative and quantitative methods to (1) understand programmers’ needs to interpret complex information to make decisions, (2) identify points where current tools lack in the needed cognitive support, and (3) design interfaces through metaphors that align with human cognition.

The broader impacts of my work apply to academia and industry. Two of my papers received awards at premier ACM and IEEE conferences in HCI and Software Engineering. My research is grounded in real-world contexts through research in Microsoft Research and working with industry collaborators like PhaseChange.ai. Some design features suggested in my papers have been adapted by Deepnote [5], a new computational notebook.

As a researcher in Human-Computer Interaction, I extensively leverage findings from Psychology and Cognitive Science realm to investigate nuances of human interactions to create human-centered design. I design tools and features using intelligent algorithms that bring existing tools closer to human thinking. These designs are driven by insights from qualitative approaches like user research (surveys, interviews, field, and controlled observations) and artifact analysis on artifacts like code, GitHub metrics, computational notebooks, posts on YouTube, Stack Overflow. I also use quantitative approaches (statistical and learning models) to identify critical challenges humans face when using tools to motivate designing tools and interfaces.

RESEARCH CONTRIBUTIONS

1. Interactive Systems that preserve and recall cognitive states

Programmers often handle more than one task simultaneously or work on them collaboratively. This requires constant switching between tasks, information about the task, and the context around the information. Understanding how programmers organize information related to multiple tasks and consistently switch contexts can inform programming interfaces to identify and partition information is easily transferable chunks that reduces the cognitive effort.

![Figure 1. Patterns of task structures. Each block represents bursts of action for a specific subgoal (marked by color). For (b) grounding, once completed with a set of subgoals, developers return to initial subgoal to check progress.](image-url)
Observing developers uninterrupted, I identified six patterns of how developers structure tasks into subgoals and sequence of activities, and how they manage information [1] as shown in Figure 1. Patterns are associated with distinct ways of brain accesses and store information as shown in Figure 2. However, decisions based even on information are influenced by other cognitive processes like experiences, biases ingrained in our brain that dictate how we make sense of information. Integrating how personal cognitive processes affect making sense of information with tool design will make tools truly adaptive and user-centric. One such cognitive process is the effect of biases.

Through an observational field study of developers, I studied the extent to which biases affect programming decisions [2]. **45.72% of the developers’ actions are associated with one or more biases. And 70.07% of these bias associated actions were reversed, which makes almost 36% of all actions observed** (Figure 3). Biases have different effects, e.g., biases causing fixating on information are common when programmers are debugging. **This work received an ACM Distinguished Paper award at ICSE 2020.** Through contextual interviews with developers, I identified 14 organizational best practices and tool needs that can help deal with biases. This work is also in the process of being published at Communications of ACM Research Highlights [8].

Tools can preserve a map of the cognitive structure of information. A personalized snapshot of information allows programmers to easily recall how their brain processed information, saving them from having to orient to the tool’s information structure.

Based on the findings of how developers manage information and how biases their actions, we designed an Integrated Development Environment (IDE) to mitigate some of the challenges [3]. The IDE is based on the metaphor or arranging cards on a table as shown in Figure 4. Cards can be anything, from code to browser windows, can be stacked in clusters of related information. Relations can be drawn and annotated between cards. This reduced friction of managing information across multiple applications. In a controlled comparative lab study with 22 students [3], the card-based IDE users could complete coding tasks with higher accuracy lower cognitive load than traditional IDEs.

### 2. Mitigating dissonance between tool and cognition in data scientists:

Computational notebooks are the platform of choice for data scientists. The flexible nature of notebooks supporting code, text, and output all in one document makes a great fit for exploration. However, as data science gains more popularity, scientists’ also need to be able to collaborate, run complex costly computations, and process large scale data. Notebooks are not built to handle such activities. I ran a systematic, mixed-methods study using semi-structured interviews (n = 15), field study (n=5) and survey (n = 156) with data scientists, to create a taxonomy of nine pain points [4]. Data scientists struggle with notebooks across all stages, from setting up their analysis and data, repetitive and tortuous multi-step manual processing of data, to dealing with the dependency hell of managing code. They don’t have a reliable archival system that can store the progress of analysis, state of the outputs, and the order in which cells have been executed – a necessary price to pay for the flexibility of
notebooks for their immediate feedback. This fundamental nature of notebooks makes it challenging to replicate and adapt analysis – whether one’s own or that from another data scientist.

Figure 5. Data scientists’ report on challenges with notebooks. 156 data scientists at Microsoft from reported the important and difficult activities to perform on notebooks. Four activities (highlighted in red) that are both important and difficult – refactor code, deploy into production, explore history, and perform long-running computations.

To identify which of these challenges have significant effect on a data scientists’ workflow, we deployed a survey among 156 data scientists at Microsoft asking them to rate the difficulty and importance of each challenge on a likert scale (Figure 5). The four activities (and the associated pain points) that were both important and difficult were: viewing and exploring history, refactoring code in notebooks, scheduling long running processes for automatic execution, and easily deploying notebooks as products. The paper discusses design opportunities for these challenges. For example, to enable reuse notebooks can incorporate familiar code assistance features of the integrated development environments. Deepnote [5] implemented multiple design opportunities suggested in the paper to improve time spent in exploring data and coding.

3. Connecting communities better through social platforms:
Online presence of programmers has grown rapidly reaching millions of people within and outside the computing realm. Programmers share their identity, work, and even details of life outside work on mass media platforms like YouTube. Potentially, this creates a big shift in how general public perceives who are programmers, allowing them to identify communities with shared experience. How to leverage online presence of developers to better connect people with their communities?

Despite the effort needed to create vlogs, there are thousands of vlogs from hundreds of developers documenting their typical days at and outside work (Figure 6). These videos break (1) misconceptions among the general public about developers, and (2) barriers caused by misinformation encouraging viewers to consider a computing career. To investigate the impact of these vlogs on YouTube, I analyzed 130 vlogs with an average of 270,000 views, conducted retrospective interviews with 16 developers who vlog, and contextual analysis of 1200 comments on the vlogs.

Figure 6. Vlogs capture an intimate view of a developer’s life; where and how they work, interact with their coworkers(left), while also giving us a glimpse of what developers do outside work e.g., grabbing lunch with friends (right).

Through axial coding of these data sources, I found the vlogs and viewers comments provide information that dismantle nine types of stereotypes surrounding developers’ personal, social, and work life [7]. They reiterate that developers are more than “nerds behind computers all day”, leading a normal life, enjoying activities like most people do. Further, a contextual survey with 335 developers at Microsoft revealed that given a chance to make their own vlogs, developers will focus on capturing interpersonal activities (mentoring, networking) and time spent outside work [6] as it helps to promote diversity in computing careers.

The analysis identifies simple modification to existing social platforms than can reduce barriers from stereotypes for current and future developers [7]. Tagging videos/posts on social platforms highlighting certain communities or allowing viewers to request tags on videos/posts based on what was informative etc. can help connect people within communities. The study with vloggers also calls out values of using less conventional data sources like vlogs
and blogs to explore community engagement [6]. Vlogs can be used as a time capsule for understanding developer activities over time, and a multitude of other cross-cultural chronological experience-based studies.

RESEARCH AGENDA:

I will continue to study cognitive processes behind computing and interacting with interfaces and curate tools centered around human cognitive needs. I plan to further study how humans make sense of information, map and model underlying neural activities like memory and thoughts, facilitate social and community interaction norms, etc. Continuing my previous research projects, I will 1) investigate how cognitive process relate to external factors for computing related reasoning, 2) narrow tool-expert dissonance by designing interfaces that align with human reasoning, and 3) aim transfer implications of this work to industry and communities at large. The following research directions guide this agenda:

*Capturing and transferring analysis decision knowledge in data science:*

Data analysis and decisions that drive the analysis are invaluable for other data scientists to learn from and consult. Reusing or reproducing this knowledge is difficult [4]. How can tools document the decisions and deciding factors such that knowledge behind decisions is transferable? How can tools make analysis decisions made by data scientists more comprehensible to others? What type of interface can improve adaptability and reuse of analysis? To answer these questions, I will analyze data science artifacts and their structures. Using empirical methods like interviews, surveys, and interactions on forums like Stack Exchange, I will model how data scientists comprehend the problem space. Based on models of comprehension, adaptation, and reuse, I am interested in building and evaluating interfaces that support these cognitive actions in data scientists.

*Designing developer tools that adapt to intrinsic cognitive behavior:*

My prior work [1,3] and research on developer productivity suggest that developers manage information clusters on much smaller levels even within same tasks. How do cognitive processes like memory, learning experience, biases towards approaches affect how developers complete their tasks? Detecting and making developers aware of intrinsic cognitive behavior patterns can improve their productivity. I am interested in investigating the effect of intrinsic information processing and underlying cognitive processes on developer productivity. How can we design interventions that can adapt to every developer’s pattern of work? Through mixed method studies and human-centered design interventions, I will investigate new ideas to create developer tools that assist cognitive processes.

*Connecting developers to their communities though virtual platforms:*

Conversations among people from non-computing background [7] on YouTube videos suggested that the barrier to computing careers are not just the technical learning curve, but also a general lack of information about the people who are programmers, software engineers, and data scientists. Barriers to a more inclusive and diverse community of professionals in computing. This can be mitigated by bringing programmers and scientists from marginal communities into the general public spotlight – news and mass media platforms like Twitch, YouTube and various podcasts and social media platforms. These spaces are avenues to discover and connect with their communities. To empower individual programmers, I will explore how professionals present themselves on various media and community platforms? How does the programming community use these spaces to express opinions and discuss current and future needs? How can these platforms be better used to connect developers and scientists to communities and likeminded professionals?

BROADER IMPACT

My research papers from the projects have been published at various ACM (SIGSOFT, SIGCHI) and IEEE conferences including International Conference on Software Engineering (ICSE), Foundations of Software Engineering (FSE), Conference on Human Factors in Computing Systems (CHI), Conference on Computer Supported Cooperative Work (CSCW), ACM Transactions on Interactive Intelligent Systems (TiiS), and Visual Languages and Human Centered Computing (VL/HCC). Two of my research papers were awarded the ACM Distinguished Paper
Award (ICSE 2020), and SIGCHI Honorable Mention Award (CHI 2020). My research has also been highlights in the ACM Research Highlights and currently under publication in Communications of the ACM. My research has brought attention to underlying cognitive processes has helped build better industrial tools. Deepnote [5], a computational notebook, builds upon some of the challenges with notebooks identified in my work by integrating built-in variable explorer, and reflecting the progress of long running executions. I will continue my research through these direct connections and collaborations across like Microsoft Research and Microsoft PROSE team, Apple, Phase Change, and institutes like Iowa State University, and University of California Irvine.

REFERENCES:


