Exploring the Role of AI Assistants in Computer Science Education: Methods, Implications, and Instructor Perspectives

Tianjia Wang, Daniel Vargas Díaz, Chris Brown, Yan Chen
Department of Computer Science, Virginia Tech, USA
{wangt7, danielvargasdiaz, dcbrown, ych}@vt.edu

***Abstract*—The use of AI assistants, along with the challenges they present, has sparked significant debate within the community of computer science education. While these tools demonstrate the potential to support students' learning and instructors' teaching, they also raise concerns about enabling unethical uses by students. Previous research has suggested various strategies aimed at addressing these issues. However, they concentrate on introductory programming courses and focus on one specific type of problem.**

The present research evaluated the performance of ChatGPT, a state-of-the-art AI assistant, at solving 187 problems spanning three distinct types that were collected from six undergraduate computer science. The selected courses covered different topics and targeted different program levels. We then explored methods to modify these problems to adapt them to ChatGPT's capabilities to reduce potential misuse by students. Finally, we conducted semi-structured interviews with 11 computer science instructors. The aim was to gather their opinions on our problem modification methods, understand their perspectives on the impact of AI assistants on computer science education, and learn their strategies for adapting their courses to leverage these AI capabilities for educational improvement. The results revealed issues ranging from academic fairness to long-term impact on students' mental models. From our results, we derived design implications and recommended tools to help instructors design and create future course material that could more effectively adapt to AI assistants' capabilities.

***Index Terms*—Computer science education, Large language model, ChatGPT, Interview**

I. Introduction

Artificial intelligence (AI) is rapidly transforming the way computer science (CS) is taught and learned. Prominent AI assistants, such as ChatGPT [1] and GitHub Copilot [2], provide students with access to advanced problem-solving resources. An increasing number of researchers have shown these AI assistants outperform students when solving complex computing tasks [3], and can circumvent plagiarism detection software [4]. These advancements bring about concerns regarding cheating and the integrity of assignments and exams, as students who do not use these assistants may be at a disadvantage when learning new concepts. Furthermore, AI assistants may generate incorrect answers which could lead students to form incorrect mental models of a concept [5]. These issues present a challenge for instructors who lack the support to address the impact these AI assistants have on education.

To address these challenges, researchers and organizations have proposed various solutions. For example, studies have investigated strategies to change course materials using prompt engineering [6]. Ethan Mollick, a Professor at the University of Pennsylvania, fully embraced AI for his classes by asking students to use AI tools in various ways [5]. Some organizations have prohibited the use of these assistants [7], or developed detectors to mitigate their use [8]–[10]. However, they have not explored the AI assistants' impact across different topics and program levels of CS courses. Also, given the rapidly evolving nature and widespread accessibility of AI assistants, these practices are also not feasible in the long run [11]. Therefore, we ask ***how can we support instructors to more effectively adapt CS education (e.g., materials, and practices) to the capabilities of AI assistants to prevent students from misuses and improve students' learning experiences***.

A three-phase study was conducted to answer this question, as outlined in Fig 1. We utilized a contextual inquiry approach in executing our research [12]. First, we evaluated ChatGPT's performance in solving problem sets from six undergraduate CS courses, which encompassed a variety of problem types. Second, we explored two problem modification methods based on previous research, including adding distracting information and altering a problem's format and evaluation, to assist instructors in adapting course materials to ChatGPT's capabilities and mitigate potential misuse. Finally, we conducted semi-structured interviews with CS instructors to gauge their understanding of ChatGPT's capabilities, their opinions on our problem modification methods, and their perspectives and concerns regarding the use of AI assistants in CS education.

Our findings show mixed feelings and concerns from the interviewees on issues, such as academic fairness, for the long-term impact of AI assistants. Specifically, we found that 1) while they recognized the potential for students to exploit ChatGPT inappropriately, the majority had not yet made changes to their materials to minimize such misuse because of the lack of effective strategies or tool support; 2) instructors perceived the adapting of course materials to incorporate AI assistants as more feasible for advanced courses than introductory ones; 3) although the interviewees expressed ethical concerns about AI assistant usage, these concerns remained unchanged compared to existing issues; and 4) the false answers generated by ChatGPT could potentially mislead

RunEx: Augmenting Regular-Expression Code Search with Runtime Values

Ashley Ge Zhang
School of Information
University of Michigan
Ann Arbor, MI USA
gezh@umich.edu

Yan Chen
Computer Science Department
Virginia Tech
Blacksburg, VA USA
ych@vt.edu

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

***Abstract*—Programming instructors frequently use in-class exercises to help students reinforce concepts learned in lecture. However, identifying class-wide patterns and mistakes in students' code can be challenging, especially for large classes. Conventional code search tools are insufficient for this purpose as they are not designed for finding semantic structures underlying large students' code corpus, where the code samples are similar, relatively small, and written by novice programmers. To address this limitation, we introduce RunEx, a novel code search tool where instructors can effortlessly generate queries with minimal prior knowledge of code search and rapidly search through a large code corpus. The tool consists of two parts: 1) a syntax that augments regular expressions with runtime values, and 2) a user interface that enables instructors to construct runtime and syntax-based queries with high expressiveness and apply combined filters to code examples. Our comparison experiment shows that RunEx outperforms baseline systems with text matching alone in identifying code patterns with higher accuracy. Furthermore, RunEx features a user interface that requires minimal prior knowledge to create search queries. Through searching and analyzing students' code with runtime values at scale, our work introduces a new paradigm for understanding patterns and errors in programming education.**

***Index Terms*—code search, programming education**

I. Introduction

In large programming courses with hundreds or thousands of students, it can be difficult for instructors to provide timely and personalized feedback. Programming instructors often face the challenge of finding and understanding class-wide patterns in students' code [1]. As a result, students may not receive the support they need to succeed, and instructors may not have a clear understanding of the learning needs of their students. For example, an instructor might want to check how many students correctly adopt the concepts taught in class, track the prevalence of particular mistakes, or simply search for specific approaches among a large set of student code samples. These tasks represent instances of "code search", as outlined in the literature by Di Grazia et al [2]. Code search is a longstanding and well-studied research topic; however, the vast majority of prior work has focused on professional developers [3]–[6].

In the context of programming education, there are unique design challenges and opportunities for code search tools. For instance, in programming classes—particularly introductory ones—an instructor might conduct a search across code samples that are relatively small and self-contained, rather than in larger codebases with complex dependencies [7], [8]. This allows for the possibility of *executing* the candidate code samples and more easily searching across runtime values in novel ways. Further, whereas many code search tools are focused on finding a handful of optimal examples (e.g., finding code to reuse), an instructor's goal might be to get descriptive statistics about their class [7]–[9]. This necessitates re-designing how we display the output from code search tools. Finally, whereas most code search tools focus on finding *new* code samples that fit some criteria, instructors might want to find code samples that are similar to an existing piece of code [7]. For example, they might observe an anti-pattern—code that works but goes against the principles being taught—in one student's code and want to assess its prevalence in the class.

In this paper, we propose RunEx, a system using a novel code search approach that enables programming instructors to quickly identify and comprehend class-wide patterns in large codebases of students' code. RunEx integrates regular expressions with runtime values for enhanced functionality. Regular expressions are powerful tools for advanced text matching and manipulation. Integrating runtime values into regular expressions expands instructors' capabilities to access code execution status, enabling a wider exploration and analysis of students' code. We provide a user-friendly interface designed for instructors to create search queries without prior knowledge of regular expressions, facilitating the creation of queries with minimal search syntax understanding. Our user interface presents search results in a way that facilitates the identification of class-wide patterns and trends. We designed our tool for programming instructors, focusing on searching through small, short, and standalone code samples that are typical of introductory courses. The user interface we develop is optimized for searching through many similar variations of a common piece of code, making it particularly suited for educational settings. We also evaluated the efficacy of RunEx through a user study with programmers and programming instructors. Our findings indicate that RunEx can significantly enhance participants' abilities in three key areas when compared to baseline systems. Firstly, it can assist them in iden-

This material is based upon work supported by the National Science Foundation under DUE 1915515.

DiLogics: Creating Web Automation Programs With Diverse Logics

Kevin Pu
jpu@dgp.toronto.edu
University of Toronto

Jim Yang
jima.yang@mail.utoronto.ca
University of Toronto

Angel Yuan
angel.yuan@mail.utoronto.ca
University of Toronto

Minyi Ma
minyi.ma@mail.utoronto.ca
University of Toronto

Rui Dong
ruidong@umich.edu
University of Michigan

Xinyu Wang
xwangsd@umich.edu
University of Michigan

Yan Chen
ych@vt.edu
Virginia Tech

Tovi Grossman
tovi@dgp.toronto.edu
University of Toronto

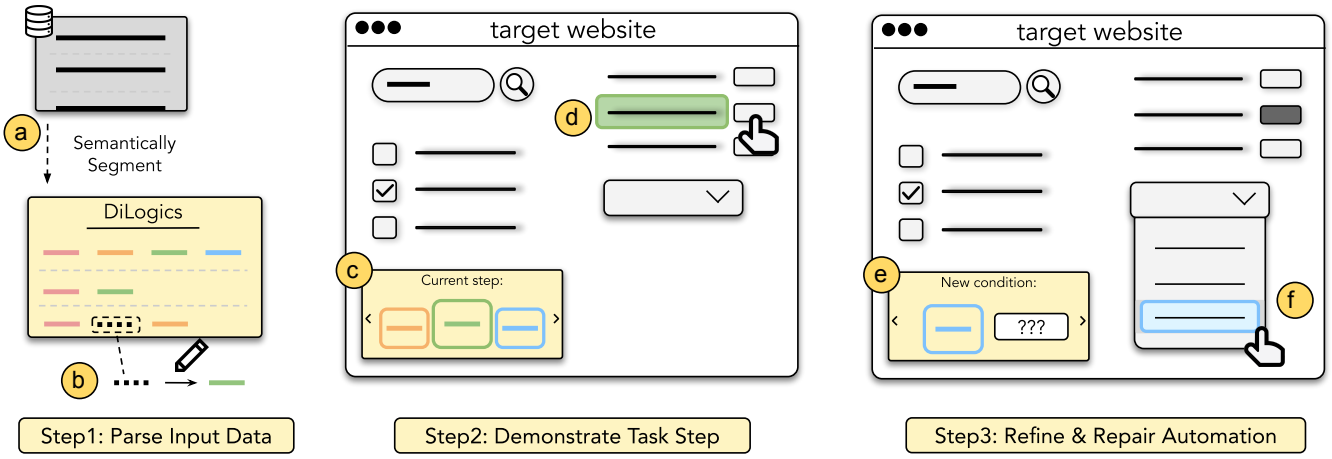


Figure 1: The workflow of DiLogics. (Step 1) To create web automation programs for data entry tasks, users first upload an input file, which is semantically segmented into the steps representing different task specifications ⓐ. Users can edit inaccurately segmented steps ⓑ. (Step 2) Users then follow a carousel of current steps ⓒ and demonstrate the corresponding UI actions to fulfill each specification. DiLogics highlights semantically relevant web page elements, guiding users to perform demonstrations ⓓ. (Step 3) After two iterations of demonstrations, DiLogics learns the mappings between different steps and actions and automates the remaining task steps, generalizing GUI actions based on the specification's semantic meaning. Users can refine the program logics ⓔ at any stage of execution by editing the steps or adding new demonstrations ⓕ.

ABSTRACT

Knowledge workers frequently encounter repetitive web data entry tasks, like updating records or placing orders. Web automation increases productivity, but translating tasks to web actions accurately and extending to new specifications is challenging. Existing tools can automate tasks that perform the same logical trace of UI actions (e.g., input text in each field in order), but do not support tasks requiring different executions based on varied input conditions. We present DiLogics, a programming-by-demonstration system that utilizes NLP to assist users in creating web automation programs that handle diverse specifications. DiLogics first semantically segments input data to structured task steps. By recording user demonstrations for each step, DiLogics generalizes the web macros to novel but semantically similar task requirements. Our evaluation showed that non-experts can effectively use DiLogics to create automation programs that fulfill diverse input instructions. DiLogics provides an efficient, intuitive, and expressive method for developing web automation programs satisfying diverse specifications.

UIST '23, October 29–November 01, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0132-0/23/10…$15.00
https://doi.org/10.1145/3586183.3606822

VizProg: Identifying Misunderstandings By Visualizing Students' Coding Progress

Ashley Zhang
University of Michigan
Ann Arbor, Michigan, USA
gezh@umich.edu

Yan Chen
Virginia Tech
Blacksburg, Virginia, USA
ych@vt.edu

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

ABSTRACT

Programming instructors often conduct in-class exercises to help them identify students that are falling behind and surface students' misconceptions. However, as we found in interviews with programming instructors, monitoring students' progress during exercises is difficult, particularly for large classes. We present VizProg, a system that allows instructors to monitor and inspect students' coding progress in real-time during in-class exercises. VizProg represents students' statuses as a 2D Euclidean spatial map that encodes the students' problem-solving approaches and progress in real-time. VizProg allows instructors to navigate the temporal and structural evolution of students' code, understand relationships between code, and determine when to provide feedback. A comparison experiment showed that VizProg helped to identify more students' problems than a baseline system. VizProg also provides richer and more comprehensive information for identifying important student behavior. By managing students' activities at scale, this work presents a new paradigm for improving the quality of live learning.

KEYWORDS

programming education at scale, code visualization

ACM Reference Format:
Ashley Zhang, Yan Chen, and Steve Oney. 2023. VizProg: Identifying Misunderstandings By Visualizing Students' Coding Progress. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3544548.3581516

1 INTRODUCTION

Programming instructors often conduct in-class coding exercises—short programming activities that students perform independently—to give students hands-on practice, assess students' progress, and identify students that are falling behind. By identifying and working with struggling students, instructors can strengthen students' understanding of the material and give them a better intuition for important concepts. However, if left unaddressed, small misunderstandings can escalate to become long-term learning barriers for students. Therefore, instructors should be able to identify struggling students and their misunderstandings during in-class exercises promptly and reliably. However, identifying problems in real time is difficult for several reasons. First, misunderstandings tend to be implicit, abstract, and not readily apparent without carefully reading students' code. However, it is often not possible to read students' code at scale in large classes or for shorter exercises. Second, there are many aspects of students' code (including aspects that the instructor might not anticipate) that instructors need to consider to gain insight into potential learning barriers. This suggests that there needs to be a better way to monitor students' code at scale.

Past research has explored ways to address these challenges. For example, Codeopticon [16] allows instructors to monitor students' code in real-time. However, Codeopticon requires that instructors read students' code individually, making it difficult to assess students' performance as a whole, particularly when needing to scale to large classes. Overcode [14] addresses the scalability issue by clustering and visualizing student code submissions [14]. However, it was designed for post-hoc analyses rather than providing real-time feedback and does not consider the need to monitor students over time. We also found in our interviews with programming instructors that time sensitivity and large class sizes make it difficult for instructors to identify learning challenges during in-lecture exercises. Ideally, instructors should be able to easily identify problems among many students' coding activities in real-time.

In this paper, we propose new techniques to address these problems and allow instructors to visualize and understand students' status in real-time for in-class programming exercises. Our design takes inspiration from maps of physical spaces. On a map, if we know a person's starting point, destination, and location, we can easily determine how close they are to their destination. With real-time updates, we could also determine if they are progressing to their destination or if they might be lost. What if checking where a student is on a programming exercise could be as easy as seeing *where* they are on a map? Although prior work has represented code in 2-D spaces [14, 19, 36], our approach is the first to do so in a way that explicitly encodes human-understandable meaning to the space (their problem-solving approach and their progress) and that can work in real-time (as students are typing). This work represents an initial step to show the feasibility and benefits of this approach.

We introduce VizProg, a tool that allows instructors to monitor and inspect large numbers of students' coding submissions over time by presenting students on a 2D map. In VizProg, students' status is represented as a position that encodes 1) similarities in students' code (as 2D Euclidean distances); 2) how students *approach* the exercise (using vertical space); 3) students' progress—how close

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04…$15.00
https://doi.org/10.1145/3544548.3581516

SemanticOn: Specifying Content-Based Semantic Conditions for Web Automation Programs

Kevin Pu
jpu@dgp.toronto.edu
University of Toronto

Rainey Fu
rainey.fu@mail.utoronto.ca
University of Toronto

Rui Dong
ruidong@umich.edu
University of Michigan

Xinyu Wang
xwangsd@umich.edu
University of Michigan

Yan Chen
yanchen@dgp.toronto.edu
University of Toronto

Tovi Grossman
tovi@dgp.toronto.edu
University of Toronto

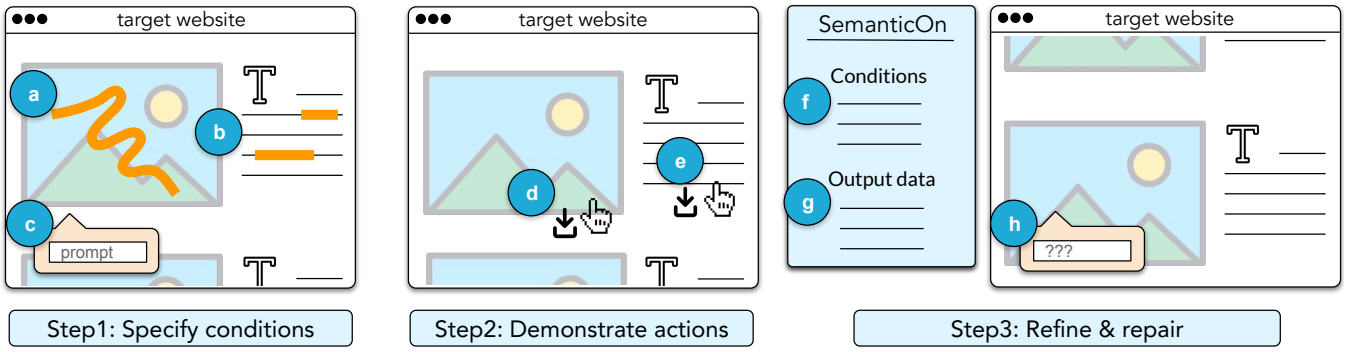


Figure 1: The workflow of SemanticOn. There are three steps to creating a web automation program with semantic conditions using SemanticOn. (Step 1) To specify semantic conditions, users can either describe their intent in text (*User Enters*, ⓒ) or indicate the section of interest by brushing through an image ⓐ or highlighting parts of a text ⓑ (*System Suggests*). SemanticOn then encodes these specifications with computer vision and natural language processing techniques into web program conditions. (Step 2) To create the intended web automation program, users demonstrate the actions on the website using WebRobot, including image downloading ⓓ and text scraping ⓔ. (Step 3) Once the program is executed, users can also easily coordinate with SemanticOn to refine the semantic conditions (ⓕ, ⓗ) or take back control to add or remove data manually ⓖ.

ABSTRACT

Data scientists, researchers, and clerks often create web automation programs to perform repetitive yet essential tasks, such as data scraping and data entry. However, existing web automation systems lack mechanisms for defining conditional behaviors where the system can intelligently filter candidate content based on semantic filters (e.g., extract texts based on key ideas or images based on entity relationships). We introduce SemanticOn, a system that enables users to *specify*, *refine*, and *incorporate* visual and textual semantic conditions in web automation programs via two methods: natural language description via prompts or information highlighting. Users can coordinate with SemanticOn to refine the conditions as the program continuously executes or reclaim manual control to repair errors. In a user study, participants completed a series of conditional web automation tasks. They reported that SemanticOn helped them effectively express and refine their semantic intent by utilizing visual and textual conditions.

KEYWORDS

Web automation, PBD, user intent, semantics

ACM Reference Format:
Kevin Pu, Rainey Fu, Rui Dong, Xinyu Wang, Yan Chen, and Tovi Grossman. 2022. SemanticOn: Specifying Content-Based Semantic Conditions for Web Automation Programs. In *The 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22), October 29-November 2, 2022, Bend, OR, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3526113.3545691

UIST '22, October 29-November 2, 2022, Bend, OR, USA

ACM ISBN 978-1-4503-9320-1/22/10…$15.00
https://doi.org/10.1145/3526113.3545691

1 INTRODUCTION

Enterprises, governments, and schools often use web-based applications to manage their businesses and services. Other than information consumption, users such as clerks, data scientists, and researchers often employ these web platforms to conduct tasks that are repetitive yet essential, such as data scraping and data entry. Performing these tasks manually can often lead to human errors (e.g., data duplicates, missed entries), which can cause inefficiencies.

Mimic: In-Situ Recording and Re-Use of Demonstrations to Support Robot Teleoperation

Karthik Mahadevan
University of Toronto
karthikm@dgp.toronto.edu

Yan Chen
University of Toronto
yanchen@dgp.toronto.edu

Maya Cakmak
University of Washington
mcakmak@cs.washington.edu

Anthony Tang
University of Toronto
tonytang@utoronto.ca

Tovi Grossman
University of Toronto
tovi@dgp.toronto.edu

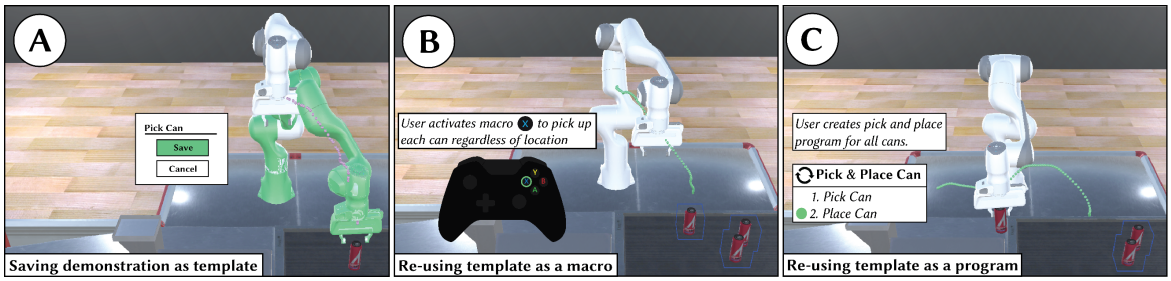


Figure 1: Mimic is a system that allows robot users to record arbitrary robot trajectory demonstrations and (A) save them as templates. Later, they can re-use templates for remotely teleoperating the robot to perform manipulation tasks through: (B) macros, parametrized templates activated by buttons, and (C) programs, sequences of parametrized templates that operate autonomously.

ABSTRACT

Remote teleoperation is an important robot control method when they cannot operate fully autonomously. Yet, teleoperation presents challenges to effective and full robot utilization: controls are cumbersome, inefficient, and the teleoperator needs to actively attend to the robot and its environment. Inspired by end-user programming, we propose a new interaction paradigm to support robot teleoperation for combinations of repetitive and complex movements. We introduce Mimic, a system that allows teleoperators to demonstrate and save robot trajectories as templates, and re-use them to execute the same action in new situations. Templates can be re-used through (1) macros—parametrized templates assigned to and activated by buttons on the controller, and (2) programs—sequences of parametrized templates that operate autonomously. A user study in a simulated environment showed that after initial set up time, participants completed manipulation tasks faster and more easily compared to traditional direct control.

UIST '22, October 29–November 02, 2022, Bend, OR, USA

ACM ISBN 978-1-4503-9320-1/22/10…$15.00
https://doi.org/10.1145/3526113.3545639

CCS CONCEPTS

• **Human-centered computing** → Human computer interaction (HCI); Interactive systems and tools.

KEYWORDS

Robot teleoperation, end-user robot programming

ACM Reference Format:
Karthik Mahadevan, Yan Chen, Maya Cakmak, Anthony Tang, and Tovi Grossman. 2022. Mimic: In-Situ Recording and Re-Use of Demonstrations to Support Robot Teleoperation. In *The 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22), October 29–November 02, 2022, Bend, OR, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3526113.3545639

1 INTRODUCTION

Robot teleoperation is necessary in many cases where robots cannot operate fully autonomously, such as for surgery and manufacturing [12, 36, 58]. In these scenarios, teleoperators need to attend to, and actively make decisions about the robot's movement based on an understanding of its environment gained through sensors (e.g., cameras). Joysticks or controllers [24, 27] are a primary method for commanding robots through combinations of complex and repetitive movements [37, 41]. However, joysticks typically have fewer degrees-of-freedom (DoF) compared to the high degrees-of-freedom that robots use to plan movements [9, 27], among other difficulties [14].

Automating repetitive tasks has the potential to relieve the teleoperator's workload. We are inspired by how collocated end-user

ASTEROIDS: Exploring Swarms of Mini-Telepresence Robots for Physical Skill Demonstration

Jiannan Li
jiannanli@dgp.toronto.edu
Department of Computer Science, University of Toronto
Toronto, Ontario, Canada

Maurício Sousa
mauricio@dgp.toronto.edu
Department of Computer Science, University of Toronto
Toronto, Ontario, Canada

Chu Li
chuchu.li@mail.utoronto.ca
Department of Computer Science, University of Toronto
Toronto, Ontario, Canada

Jessie Liu
heng.liu@mail.utoronto.ca
Department of Computer Science, University of Toronto
Toronto, Ontario, Canada

Yan Chen
yanchen@dgp.toronto.edu
Department of Computer Science, University of Toronto
Toronto, Ontario, Canada

Ravin Balakrishnan
ravin@dgp.toronto.edu
Department of Computer Science, University of Toronto
Toronto, Ontario, Canada

Tovi Grossman
tovi@dgp.toronto.edu
Department of Computer Science, University of Toronto
Toronto, Ontario, Canada

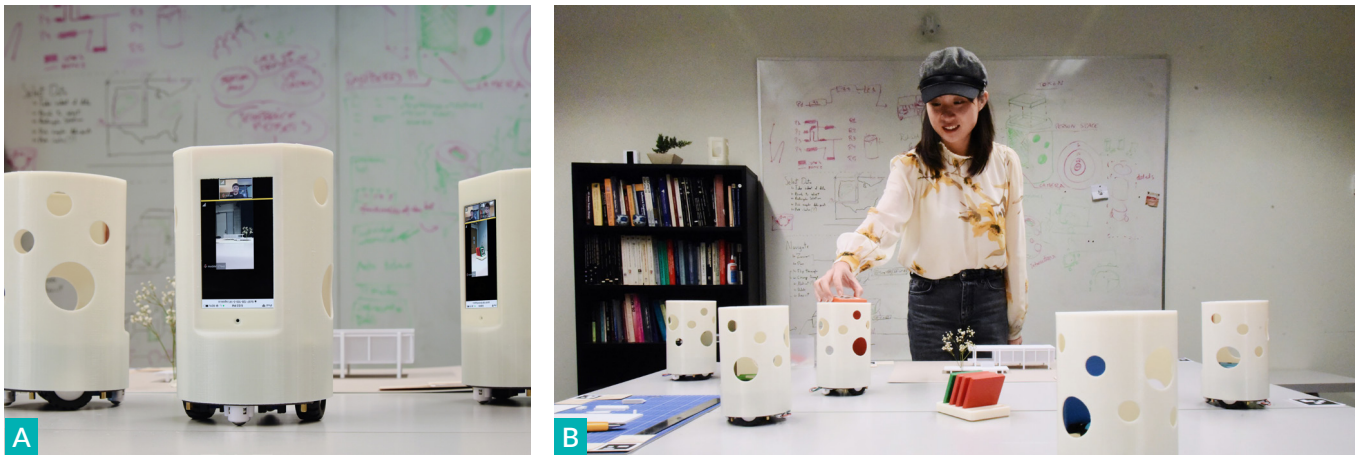


Figure 1: Asteroids is a novel approach for remote physical demonstrations using a swarm of telepresence robots. A) With Asteroids, remote audience members can inhabit and control small robots on a workbench to follow the instructor's guidance or roam around looking at activities at various locations and scales. B) And, a demonstrator can physically interact with the remote audience and use tangible artifacts to control the flow of the demonstration.

ABSTRACT

Online synchronous tutoring allows for immediate engagement between instructors and audiences over distance. However, tutoring physical skills remains challenging because current telepresence approaches may not allow for adequate spatial awareness, viewpoint control of the demonstration activities scattered across an entire work area, and the instructor's sufficient awareness of the audience. We present Asteroids, a novel approach for tangible robotic telepresence, to enable workbench-scale physical embodiments of remote people and tangible interactions by the instructor. With Asteroids, the audience can actively control a swarm of mini-telepresence robots, change camera positions, and switch to other robots' viewpoints. Demonstrators can perceive the audiences' physical presence while using tangible manipulations to control the audience's viewpoints and presentation flow. We conducted an exploratory

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9157-3/22/04…$15.00
https://doi.org/10.1145/3491102.3501927

Umitation: Retargeting UI Behavior Examples for Website Design

Yan Chen
yanchen@dgp.toronto.edu
University of Toronto
Toronto, Ontario, Canada

Tovi Grossman
tovi@dgp.toronto.edu
University of Toronto
Toronto, Ontario, Canada

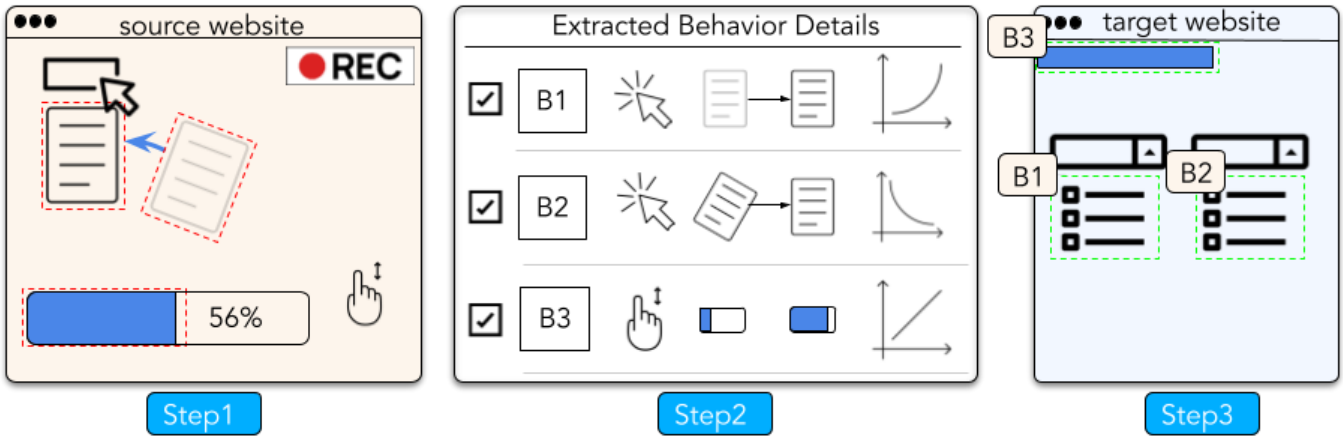


Figure 1: The workflow of Umitation. There are three steps to using Umitation to retarget example UI behaviors from a source website to a target website. (Step 1) Users first specify one or more source elements on the source website and then record their behaviors by interacting with them. Umitation will automatically capture the Document Object Model (DOM) changes. (Step 2) Umitation displays the low-level details of the recorded behaviors on its main panel, and users can directly manipulate the meta data of the behaviors. (Step 3) Umitation guides users to retarget the behaviors to appropriate elements (e.g., structurally similar elements) on the target website.

ABSTRACT

Interface designers often refer to UI behavior examples found in the wild (e.g., commercial websites) for reference or design inspiration. While past research has looked at retargeting interface and webpage design, limited work has explored the challenges in retargeting interactive visual behaviors. We introduce Umitation, a system that helps designers extract, edit, and adapt example front-end UI behaviors to target websites. Umitation can also help designers specify the desired behaviors and reconcile their intended interaction details with their existing UI. In a qualitative evaluation, we found evidence that Umitation helps participants extract and retarget dynamic front-end UI behavior examples quickly and expressively.

UIST '21, October 10–14, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8635-7/21/10…$15.00
https://doi.org/10.1145/3472749.3474796

KEYWORDS

Retarget design, UI behavior examples, user intent and disambiguation

ACM Reference Format:
Yan Chen and Tovi Grossman. 2021. Umitation: Retargeting UI Behavior Examples for Website Design. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21), October 10–14, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3472749.3474796

1 INTRODUCTION

Interactive behavior, defined by Myers et al. [48] as the "feel" (as opposed to the "look") of a user interface (UI), is a key component of the modern website. When designed well, interactive UI behaviors can make a website engaging and user-friendly. UI designers often browse behavior examples and use them on their own artifacts to explore different aspects of the design [22, 50]. Exploring existing solutions that fit the current context can help designers creatively address unfamiliar situations [30, 31]. In-context interactive mockups could also help make team communication more effective [11]. However, current practices for leveraging design examples on a new interface have four main limitations: they are often 1) informal, with designers using professional tools (e.g., Adobe XD, Figma) with a limited set of pre-defined modules; or 2) ad hoc, with designers

CoCapture: Effectively Communicating UI Behaviors on Existing Websites by Demonstrating and Remixing

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

Sang Won Lee
Virginia Tech
Blacksburg, Virginia, USA
sangwonlee@vt.edu

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

Figure 1: The workflow of CoCapture. There are four steps of using CoCapture to communicate new UI behavior mockups on an existing website. (Step 1) Users first capture existing interface behaviors (base scene) by interacting with the website (scrolling), and CoCapture will automatically capture the Document Object Model (DOM) changes. (Step 2) In CoCapture's main panel, users can add new behaviors on top of the base scene by demonstration; that is, by directly manipulating any elements (e.g., drag and drop the red element in the replay and see immediate changes). (Step 3) Users can remix (post-edit, e.g., change duration) added behaviors to finalize the mockup. (Step 4) Users can refer to the DOM elements or added behaviors in the textual description using hypertext.

ABSTRACT

User Interface (UI) mockups are commonly used as shared context during interface development collaboration. In practice, UI designers often use screenshots and sketches to create mockups of desired UI behaviors for communication. However, in the later stages of UI development, interfaces can be arbitrarily complex, making it labor-intensive to sketch, and static screenshots are limited in the types of interactive and dynamic behaviors they can express. We introduce CoCapture, a system that allows designers to easily create UI behavior mockups on existing web interfaces by demonstrating and remixing, and to accurately describe their requests to helpers by referencing the resulting mockups using hypertext. We showed that participants could more accurately describe UI behaviors with CoCapture than with existing sketch and communication tools and that the resulting descriptions were clear and easy to follow. Our approach can help teams develop UIs efficiently by bridging communication gaps with more accurate visual context.

KEYWORDS

Rapid UI prototyping; UI design communication

ACM Reference Format:
Yan Chen, Sang Won Lee, and Steve Oney. 2021. CoCapture: Effectively Communicating UI Behaviors on Existing Websites by Demonstrating and Remixing. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3411764.3445573

1 INTRODUCTION

Mockups are widely recognized in Human-Computer Interaction (HCI) as invaluable tools for communicating and evaluating design ideas. A mockup can help ground descriptions of UI functionality and can serve as a "boundary object" that allows designers to communicate with developers and other stakeholders. Mockups are useful *throughout* the User Interface (UI) development lifecycle, from exploration to refinement. However, most tools for mockup creation are built for the earlier (exploratory) stages of UI development.

There are several challenges when creating mockups as communication tools in the later stages of UI development—for example, to

CHI '21, May 8–13, 2021, Yokohama, Japan

ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445573

Towards Supporting Programming Education at Scale via Live Streaming

YAN CHEN*, University of Michigan, USA
WALTER S. LASECKI, University of Michigan, USA
TAO DONG, Google Inc., USA

Live streaming, which allows streamers to broadcast their work to live viewers, is an emerging practice for teaching and learning computer programming. Participation in live streaming is growing rapidly, despite several apparent challenges, such as a general lack of training in pedagogy among streamers and scarce signals about a stream's characteristics (e.g., difficulty, style, and usefulness) to help viewers decide what to watch. To understand why people choose to participate in live streaming for teaching or learning programming, and how they cope with both apparent and non-obvious challenges, we interviewed 14 streamers and 12 viewers about their experience with live streaming programming. Among other results, we found that the casual and impromptu nature of live streaming makes it easier to prepare than pre-recorded videos, and viewers have the opportunity to shape the content and learning experience via real-time communication with both the streamer and each other. Nonetheless, we identified several challenges that limit the potential of live streaming as a learning medium. For example, streamers voiced privacy and harassment concerns, and existing streaming platforms do not adequately support viewer-streamer interactions, adaptive learning, and discovery and selection of streaming content. Based on these findings, we suggest specialized tools to facilitate knowledge sharing among people teaching and learning computer programming online, and we offer design recommendations that promote a healthy, safe, and engaging learning environment.

CCS Concepts: • **Human-centered computing → Empirical studies in HCI**; • **Applied computing → Collaborative learning**.

Additional Key Words and Phrases: Live streaming; programming education; informal learning; live coding

ACM Reference Format:
Yan Chen, Walter S. Lasecki, and Tao Dong. 2020. Towards Supporting Programming Education at Scale via Live Streaming. In *Proceedings of CSCW '20: ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW '20).* ACM, New York, NY, USA, 22 pages. https://doi.org/TBD

1 INTRODUCTION

In the last few years, video-centric learning materials from online platforms such as YouTube, Coursera, and Khan Academy have enabled people of all ages and countries to share and gain knowledge [1, 2, 4]. These videos are often well-structured, pre-recorded, and carefully edited by experienced instructors or even professional teams, and the content often covers specific topics, concepts, or techniques. However, the effort required to start sharing knowledge using these approaches is often high. Moreover, the asynchronous interaction and non-immediate feedback can impede learning and reduce student engagement [10, 64].

*This work was completed while the author was an intern at Google.

CSCW '20, 2020,

ACM ISBN TBD…$15.00
https://doi.org/TBD

Bashon: A Hybrid Crowd-Machine Workflow for Shell Command Synthesis

Yan Chen, Jaylin Herskovitz, Walter S. Lasecki, Steve Oney
University of Michigan, Ann Arbor, United States
{yanchenm, jayhersk, wlasecki, soney}@umich.edu

***Abstract*—Despite advances in machine learning, there has been little progress towards creating automated systems that can reliably solve general purpose tasks, such as programming or scripting. In this paper, we propose techniques for increasing the reliability of automated systems for program synthesis tasks via a hybrid workflow that augments the system with input from crowds of human workers. Unlike previous hybrid workflow systems, which have been focused on less complex tasks that crowd workers can do in their entirety (e.g., image labeling), our proposed workflow handles tasks that untrained crowd workers cannot do alone (i.e., scripting). We evaluate our approach by creating BashOn, a system that increases the performance of an automated program that generates Bash shell commands from natural language descriptions by ∼30%. Our approach can not only help people make program synthesis tools more robust, reliable, and trustworthy for end-users to use, but also help lower the cost of downstream data collection for program synthesis when a preliminary model exists.**

***Index Terms*—program synthesis; crowdsourcing; crowd workflows**

I. Introduction

Bash is a complex but powerful Unix shell and user interface. Users can parameterize and combine Bash commands to quickly perform complex operations that would take much longer in a Graphical User Interface (GUI). However, correctly writing a Bash command requires complex reasoning skills and years of practical experience. Even experienced Bash users often need to rely on support from various resources like Stack Overflow or man pages [1], [2] for recalling code syntax.

Although researchers have created automated systems to help users write Bash scripts and programs from natural language descriptions [3]–[5], they are limited to accomplishing constrained tasks. In our tests with Tellina [4], a state-of-the-art automated tool that translates natural language queries found in the wild into Bash commands, we found a 10% accuracy rate. Part of this is because currently, most automated systems only work reliably in domains where there is sufficient, well-organized training data [6] and the "output" of the system is simple [7]. The only reliable source for support in complex domains like Bash scripting is from human experts, who can be difficult to find.

In this paper, we propose leveraging non-expert crowd workers to boost the accuracy of an existing program synthesis system (Tellina [4]) for generating Bash commands from natural language descriptions. We show that despite not having expertise with Bash scripting, crowd workers can increase the effectiveness of the fully automated system by nearly 30%. We also introduce Bashon (Fig. 1), a user interface that uses our hybrid workflow to allow users to generate Bash commands from natural language descriptions.

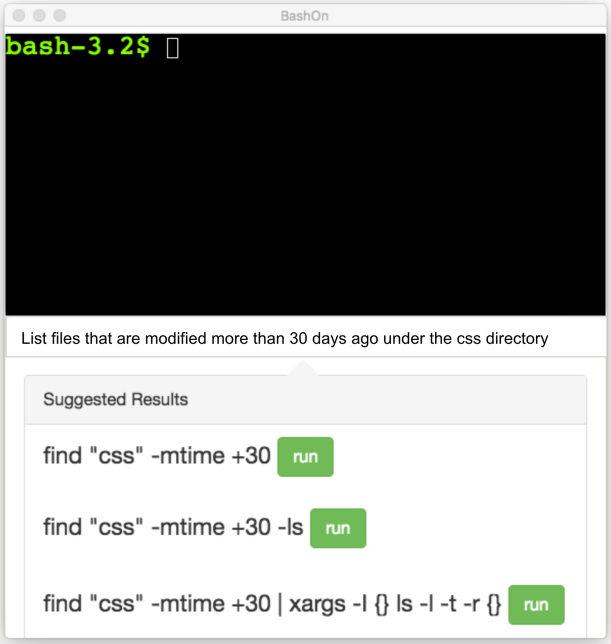


Fig. 1. Bashon allows users to create Bash commands from natural language. It uses a hybrid workflow that integrates crowd workers and an AI for program synthesis. In this example, the user types a command in natural language: "List files that are modified more than 30 days ago under the css directory" and Bashon proposes three Bash commands. The user can press 'Run' to execute any of the proposed commands

An important feature of Bashon is that it relies on crowd workers who do not have experience with Bash commands. This is in contrast with previous hybrid workflow systems, which rely on crowd workers' ability to perform a task in its entirety [8]–[12]. Thus, when using non-expert crowd workers, these systems are limited to domains that have relatively low complexity (i.e., little prior knowledge is required). A key insight when designing Bashon is that although non-expert workers are less effective than an automated system at generating Bash commands, they can be effective in targeted portions of its workflow. For example, although a given crowd

978-1-7281-6901-9/20/31.00 ©2020 IEEE

EdCode: Towards Personalized Support at Scale for Remote Assistance in CS Education

Yan Chen[1], Jaylin Herskovitz[1], Gabriel Matute[1], April Wang[1], Sang Won Lee[2], Walter S. Lasecki[1], Steve Oney[1]

[1]*University of Michigan, Ann Arbor, United States, {yanchenm,jayhersk,gmatute,aprilww,wlasecki,soney}@umich.edu*
[2]*Virginia Polytechnic Institute and State University, Blacksburg, United States, sangwonlee@vt.edu*

***Abstract*—Programming support methods, like discussion forums and office hours, are important in CS education, but difficult to scale. In this paper, we introduce EdCode, a system that allows students to seek remote instructional support within their IDE in a way that resembles in-person support. It also allows instructors to provide contextualized responses by referencing students' code, and curate and publish their answers for an entire class by selecting only the relevant part of the code referenced, thereby helping to avoid plagiarism. We evaluated EdCode with a series of usability studies and identified benefits and challenges for its use in programming courses. Students found that the perceived quality of support from EdCode was comparable to that of support from in-person office hours, and both students and instructors found publishing and viewing other students' answers helpful.**

***Index Terms*—Programming Education, Remote Assistance, Scalable Support**

I. Introduction

The enormous growth of software development jobs has led to a rise in the desire to learn to code [1], [2]. As a result, demand for computer science (CS) courses has swelled and instructional resources have struggled to keep up with this demand. Prior work has shown that instructional resources, such as in-person support, can help students improve their performance in programming courses [3], [4]. In-person support makes it easy for instructors to access students' code, allows instructors to proactively help, and is personalized [5]. However, instructional support can be hard to scale in courses that have high student-teacher ratios, as many CS courses do.

Many courses use Q&A forums (e.g., Piazza) to scale support for students but forum participation is often low due to their public nature and the inability to have a natural conversation [6]–[11]. Several systems, including Codeon [12], have been shown to be effective at scaling remote programming assistance in work settings. However, programming support in educational settings is different than work settings in terms of goals, stakeholders' expertise, and collaboration structure. For instance, students often lack sufficient knowledge and understanding to phrase a question correctly compared to professional programmers. Instructors would prefer to guide students with hints and questions rather than giving away the solutions, whereas professional programmers tend to offer straightforward solutions that can be replicated by others.

978-1-7281-6901-9/20/$31.00 ©2020 IEEE

In this paper, we introduce EdCode, a remote support system that allows instructors to provide personalized assistance to students and publicly share their support with coding questions in programming classes. We conducted two needfinding studies and hypothesized that 1) supporting both asynchronous and synchronous interaction could help instructors better guide students' learning processes, 2) allowing instructors to refer to portions of code in their responses could make their explanations clearer, and 3) allowing instructors to select relevant code in students' questions to share with the entire class would help scale their support. EdCode instantiates these ideas by adding three new features on top of Codeon—**contextualized explanations**, which allows students to make text-based help requests with code context (by highlighting relevant code snippets) from their working context (IDE state) in asynchronous fashion; a **chat tool** that allows instructors to converse with students within each question; and **selective code publicity**, which allows instructors to publish students' questions and answers by selecting code regions that are relevant to the question while concealing the rest of the solution. In this way, other students could review questions and answers curated by the instructors, leading to fewer duplicate questions and saving time for instructors and students alike.

We conducted a virtual office hour study in an introductory programming course. Both student and instructor participants reported that the quality of answers from EdCode was better than their existing Q&A forum (Facebook Group). Students also found the answers were comparable to those obtained from in-person office hours, and they could understand the questions and answers with only the curated version of the original code visible. Our contributions are:

- study-based insights into the needs and challenges of using existing asynchronous remote support tools in programming classes;
- a system (EdCode) that addresses these needs and challenges, used as a 'technology probe'; and
- evidence that EdCode has the potential to be useful in a classroom setting.

II. Related Work

Community question-answering websites (e.g,. Stack Overflow) are common help-seeking systems, but they often lack personalized support for programmers, especially novices [13]–[15]. To scale personalized support, prior work has proposed

Sifter: A Hybrid Workflow for Theme-based Video Curation at Scale

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

Andrés Monroy-Hernández
Snap Inc.
Seattle, WA, USA
amh@snap.com

Ian Wehrman
Snap Inc.
Venice, CA, USA
iwehrman@snap.com

Walter S. Lasecki
University of Michigan
Ann Arbor, MI, USA
wlasecki@umich.edu

Steve Oney
University of Michigan
Ann Arbor, MI, USA
soney@umich.edu

Rajan Vaish
Snap Inc.
Venice, CA, USA
rvaish@snap.com

ABSTRACT

User-generated content platforms curate their vast repositories into thematic compilations that facilitate the discovery of high-quality material. Platforms that seek tight editorial control employ people to do this curation, but this process involves time-consuming routine tasks, such as sifting through thousands of videos. We introduce Sifter, a system that improves the curation process by combining automated techniques with a human-powered two-stage pipeline that browses, selects, and reaches an agreement. We evaluated Sifter by creating 12 compilations from over 34,000 user-generated videos. Sifter was more than three times faster than dedicated curators, and its output was of comparable quality. We reflect on the challenges and opportunities introduced by Sifter to inform the design of content curation systems that need subjective human judgments of videos at scale.

CCS CONCEPTS

• **Human-centered computing** → **Collaborative and social computing systems and tools**; • **Computing methodologies**;

KEYWORDS

Crowdsourcing, video processing, social media, hybrid workflow, video content analysis

ACM Reference Format:
Yan Chen, Andrés Monroy-Hernández, Ian Wehrman, Walter S. Lasecki, Steve Oney, and Rajan Vaish. 2018. Sifter: A Hybrid Workflow for Theme-based Video Curation at Scale. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Every day, millions of people around the world create, share, and consume short videos on platforms like Snapchat, TikTok, and Douyin. These platforms use a variety of curation approaches to help their users discover high-quality and recent ("fresh") content. These approaches leverage artificial intelligence (AI), user-sourcing, or dedicated curators [12]. AI techniques rely on algorithmic aggregation and the ranking of relevant content based on metadata, such as tags [28]. These approaches are scalable but limited in their capacity to identify content attributes that require subjective assessments and nuanced cultural understanding. User-sourcing approaches rely on end-users' votes or "likes" to identify high-quality popular content, such as on Reddit [30]. These approaches are also scalable, but have the potential to silence minority opinions or to be dominated by content manipulation strategies like "brigading" [10]. Lastly, curator-based approaches rely on staff curators to identify and organize compelling content, such as on Snapchat's Discover [35] or Twitter's Moments [38] (Fig. 1). These approaches give platforms editorial control and overcome machines' inability to make subjective assessments and prevent adversarial users from manipulating content selection, but are limited by scale [12]. Specifically, it is difficult to scale curators' ability to find appropriate content from a corpus of videos that is large and rapidly growing—on Youtube, for example, over 500 hours of video content is uploaded every minute.

In this paper, we introduce *Sifter* to scale the third type of curation strategy (dedicated curators). Sifter combines automated video processing techniques and crowdsourced human expertise to provide on-demand assistance to dedicated video curators in the process of selecting and collecting content (i.e., the "select and collect" phase in Fig. 2). In this phase, curators have to rapidly browse through large "fresh-content" corpora to collect just enough raw material that might fit a coherent narrative [2, 39], or theme (e.g., "magic tricks", or the movie Lion King). As the corpora often have more qualified (e.g., interesting, relevant) materials than needed, curators do not have to exhaust all the items.

This setting makes our problem unique, but daunting for three main reasons. First, despite being short, videos often take more time to consume and infer than other media, like images, as they contain multimodal signals (e.g., visual, audio, text caption) and interconnected actions. As a result, curators may have to watch the videos multiple times to grasp the essence, which extends the task time. Second, the corpora often contain many unqualified videos that are distracting, further slowing curators down. Although automated video analysis techniques have become promising, machines

IMX '20, JUNE 17 -19 , 2020, Barcelona, Spain

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

Towards Providing On-Demand Expert Support for Software Developers

Yan Chen[1], **Steve Oney**[1], **Walter S. Lasecki**[2,1]
School of Information[1], Computer Science & Engineering[2]
University of Michigan – Ann Arbor
{yanchenm,soney,wlasecki}@umich.edu

ABSTRACT
Software development is an expert task that requires complex reasoning and the ability to recall language or API-specific details. In practice, developers often seek support from IDE tools, Web resources, or other developers to help fill in gaps in their knowledge on-demand. In this paper, we present two studies that seek to inform the design of future systems that use remote experts to support developers on demand. The first explores what types of questions developers would ask a hypothetical assistant capable of answering any question they pose. The second study explores the interactions between developers and remote "experts" in supporting roles. Our results suggest eight key system features needed for on-demand remote developer assistants to be effective, which has implications for future human-powered development tools.

Author Keywords
Pair programming; intelligent assistants; crowdsourcing

ACM Classification Keywords
H.5.m Information Interfaces and Presentation (e.g. HCI): Miscellaneous; K.6.1 Management of Computing and Information Systems: Software Development

INTRODUCTION
Software development is an expert task that requires complex reasoning skills within the bounds of formal language syntax and Application Programming Interface (API)-specific functionality. Support for recalling keywords, function names, argument types and methodologies, and other details can come from a variety of sources. Current support usually comes either from tools built directly into Integrated Development Environments (IDEs) or from other developers.

IDE tools provide contextualized, easily-accessible, and on-demand support for developers, but are generally limited in the types of feedback they can provide (e.g., syntax error highlighting and function auto-complete) because the system cannot truly understand user queries or the context of the problem. We find that these limitations preclude many questions that developers *would* ask while programming.

CHI'16, May 07-12, 2016, San Jose, CA, USA

http://dx.doi.org/10.1145/2858036.2858512

To overcome the limitations of automatic approaches, support from other human developers is often enlisted. This can take several forms, each with their own trade-offs. In a work environment, an expert colleague can provide useful support in the specific languages and frameworks in use, but is unlikely to be available on demand or able to support frequent questions. Web forums can provide a wealth of information about general questions, but typically do not provide highly personalized support or quick responses to developer queries. Additionally, many of the most successful developer forums restrict the types of questions that can be asked and when they can be asked. For example, Stack Overflow [41] explicitly discourages the types of project-specific code requests that we observed in our motivating studies. It also requires some level of reciprocity from users in the form of useful answers to other developers' questions. Providing sufficient context for others to help solve a problem can also be a time-consuming task that often takes multiple turns of interaction to complete.

Hired freelance developers can provide tailored support for specific questions, immediate answers, and even the ability to hand off sub-tasks for independent completion. However, in addition to the monetary price of hiring a freelancer, the traditional hiring process adds significant time and preparation effort costs (interviews, initiation, etc.). Lastly, none of these expert solutions provide the in-context support that IDE tools do, adding the need for an additional context switch to and from a developer's workflow.

The rise of online expert crowd platforms, such as Upwork [14], allows on-demand, programmatic hiring of developers for pay. However, the complexity of the hiring and onboarding process needed to find reliable workers who have sufficient knowledge of the project remains unchanged. Reducing the hiring overhead is critical to making expert support in development processes feasible. Ideally, asking for help would be as simple as informally saying the question at hand, and providing in-context support within an IDE.

In this paper, we introduce a new space of support systems and present two studies that inform the design of such systems in the future. In the first, we explore the types of questions developers might want to ask an on-demand support agent. In the second, we examine the logistical challenges of providing on-demand remote developer support.

Codeon: On-Demand Software Development Assistance

Yan Chen[1], Sang Won Lee[2],Yin Xie[1], YiWei Yang[2], Walter S. Lasecki[2,1], and Steve Oney[1,2]
School of Information[1], Computer Science & Engineering[2], University of Michigan, Ann Arbor
{yanchenm,snaglee,xieyin,yanyiwei,wlasecki,soney}@umich.edu

ABSTRACT
Software developers rely on support from a variety of resources—including other developers—but the coordination cost of finding another developer with relevant experience, explaining the context of the problem, composing a specific help request, and providing access to relevant code is prohibitively high for all but the largest of tasks. Existing technologies for synchronous communication (e.g. voice chat) have high scheduling costs, and asynchronous communication tools (e.g. forums) require developers to carefully describe their code context to yield useful responses. This paper introduces Codeon, a system that enables more effective task hand-off between end-user developers and remote helpers by allowing *asynchronous* responses to on-demand requests. With Codeon, developers can request help by speaking their requests aloud within the context of their IDE. Codeon automatically captures the relevant code context and allows remote helpers to respond with high-level descriptions, code annotations, code snippets, and natural language explanations. Developers can then immediately view and integrate these responses into their code. In this paper, we describe Codeon, the studies that guided its design, and our evaluation that its effectiveness as a support tool. In our evaluation, developers using Codeon completed nearly twice as many tasks as those who used state-of-the-art synchronous video and code sharing tools, by reducing the coordination costs of seeking assistance from other developers.

Author Keywords
Development support; intelligent assistants; crowdsourcing

ACM Classification Keywords
H.5.m Information Interfaces and Presentation (e.g. HCI): Miscellaneous; K.6.1 Management of Computing and Information Systems: Software Development

INTRODUCTION
Software developers rely heavily on support from external resources while programming. Although search engines and Community Question-Answering (CQA) websites (such as StackOverflow [33]) are the most popular resources for developers, the best support is often provided by other developers [31, 20, 12]. Unlike web-based resources, expert developers can provide personalized help, high-level advice, and project-specific code segments, and can often help identify and overcome bugs that are difficult for a single developer to find on their own [35]. However, it is often prohibitively difficult to find other developers willing to help, particularly for developers working outside of a large organization.

Recently, a small set of paid services began connecting developers with remote expert developers [23, 21], who can provide personalized feedback. These services use a *synchronous*, one-on-one model of communication where developers connect to a remote expert, make a request, and communicate via video chat and a shared editor. However, there are several drawbacks to this synchronous model [12]. There is a coordination cost of finding an expert who is available to help at the right time. If the first expert does not have sufficient expertise (which they cannot know until *after* they connect), there is a further cost—in both time and money—to finding a new expert. One-on-one mentoring also requires that the developer be attentive to the remote helper throughout the session. Although this is suitable for teaching-oriented requests where a back-and-forth conversation between developers and helpers is desirable, it is less helpful for tasks that can be handed off entirely to the helper, such as requests for short code snippets.

On-Demand Programming Assistance with Codeon
In this paper, we propose an *asynchronous* on-demand help seeking model for programmers who need support that can be more efficiently provided by remote expert developers than existing methods. We implement and evaluate this model in Codeon, a system that allows developers to request assistance as easily as they can through in-person one-on-one communication, and tracks helpers' responses, directly in the developer's Integrated Development Environment (IDE). As we will show, Codeon makes remote collaboration more practical by reducing coordination costs while still enabling rich communication between developers and helpers. Unlike previous asynchronous collaboration solutions (such as code repositories), Codeon is request-oriented: it makes it easy for developers to make sufficiently detailed requests and send to other developers, making the process quick and effective. Further, Codeon's asynchronous model is more scalable for multiple helpers than synchronous support tools because it allows multiple helpers to work in parallel with the developer. As our evaluation demonstrates, Codeon supports new forms of par-

CHI 2017, May 06-11, 2017, Denver, CO, USA.

DOI:http://dx.doi.org/10.1145/3025453.3025972

Improving Crowd-Supported GUI Testing with Structural Guidance

Yan Chen, Maulishree Pandey, Jean Y. Song, Walter S. Lasecki, Steve Oney
University of Michigan
Ann Arbor, MI, USA
{yanchenm, maupande, jyskwon, wlasecki, soney}@umich.edu

ABSTRACT

Crowd testing is an emerging practice in Graphical User Interface (GUI) testing, where developers recruit a large number of crowd testers to test GUI features. It is often easier and faster than a dedicated quality assurance team, and its output is more realistic than that of automated testing. However, crowds of testers working in parallel tend to focus on a small set of commonly used User Interface (UI) navigation paths, which can lead to low test coverage and redundant effort. In this paper, we introduce two techniques to increase crowd testers' coverage: *interactive event-flow graphs* and *GUI-level guidance*. The interactive event-flow graphs track and aggregate every tester's interactions into a single directed graph that visualizes the cases that have already been explored. Crowd testers can interact with the graphs to find new navigation paths and increase the coverage of the created tests. We also use the graphs to augment the GUI (GUI-level guidance) to help testers avoid only exploring common paths. Our evaluation with 30 crowd testers on 11 different test pages shows that the techniques can help testers avoid redundant effort while also increasing untrained testers' coverage by 55%. These techniques can help us develop more robust software that works in more mission-critical settings, not only by performing more thorough testing with the same effort that has been put in before, but also by integrating these techniques into different parts of the development pipeline to make more reliable software in the early development stage.

Author Keywords

GUI testing; Software testing; Crowdsourcing

CCS Concepts

•**Human-centered computing** → **Human computer interaction (HCI); Interactive systems and tools;** User studies;

CHI '20, April 25–30, 2020, Honolulu, HI, USA.

ACM 978-1-4503-6708-0/20/04. . . $15.00
DOI: http://dx.doi.org/10.1145/3313831.3376835

INTRODUCTION

Software testing is an important, yet often overlooked, part of the software development lifecycle. In the case of GUI development, testing helps developers find functional and usability defects in a system's front-end. This testing requires test cases that consist of a sequence of input events (e.g., writing in the input field and then clicking a button), which we define as *navigation paths*, and the resulting output (e.g., a modal window pops up) [4, 54], which we define as *GUI state*. Prior work has shown that GUI testing can be effective in finding both front-end and back-end defects because they reflect usage scenarios and often execute back-end code [8, 43]. However, due to the multitude of possible user event sequences, it can be challenging to design a comprehensive set of tests even for simple user scenarios (e.g., purchasing an item on an e-commerce site).

Traditionally, software testing was conducted by dedicated quality assurance (QA) teams with formally trained testers. Although these QA teams are reliable, the high cost and delayed responses made them hard to scale and non-flexible for rapid update needs for the software industry today. Automated testing could be one solution, but the inability to create realistic user behavior test cases makes them hard to rely on given the variations in software products *Crowd testing* is an emerging practice that enables testing with more flexibility and scalability than QA teams [15, 27, 48, 49, 50]. It involves recruiting crowd workers (either untrained or trained) from platforms like Mechanical Turk [2] or uTest [3] to perform GUI tests. However, crowd testing often results in a high degree of test case duplication [49], because crowd workers tend to navigate the same common paths while working in parallel. Prior work focused on analyzing workers' responses to identify and remove duplicates [49], rather than preventing the issue. This duplication of test cases can lead to lower test coverage, making the testing process less effective or more costly.

To address this duplication problem, our insight is to augment GUI testing with visual cues that guide testers' attention to unexplored navigation paths. Specifically, we propose *interactive event-flow graphs* and *GUI-level guidance* (Fig. 1), to make crowd testing more effective. These techniques give testers a human-readable navigation path history graph that is situated on testers' current GUI state. This draws on information foraging theory [29, 40, 52], which suggests that providing

PuzzleMe: Leveraging Peer Assessment for In-Class Programming Exercises

APRIL YI WANG*, University of Michigan, USA
YAN CHEN*, University of Michigan, USA
JOHN JOON YOUNG CHUNG, University of Michigan, USA
CHRISTOPHER BROOKS, University of Michigan, USA
STEVE ONEY, University of Michigan, USA

Peer assessment, as a form of collaborative learning, can engage students in active learning and improve their learning gains. However, current teaching platforms and programming environments provide little support to integrate peer assessment for in-class programming exercises. We identified challenges in conducting such exercises and adopting peer assessment through formative interviews with instructors of introductory programming courses. To address these challenges, we introduce PuzzleMe, a tool to help Computer Science instructors to conduct engaging in-class programming exercises. PuzzleMe leverages peer assessment to support a collaboration model where students provide timely feedback on their peers' work. We propose two assessment techniques tailored to in-class programming exercises: live peer testing and live peer code review. Live peer testing can improve students' code robustness by allowing them to create and share lightweight tests with peers. Live peer code review can improve code understanding by intelligently grouping students to maximize meaningful code reviews. A two-week deployment study revealed that PuzzleMe encourages students to write useful test cases, identify code problems, correct misunderstandings, and learn a diverse set of problem-solving approaches from peers.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

Additional Key Words and Phrases: peer assessment, live programming, synchronous code sharing

ACM Reference Format:
April Yi Wang, Yan Chen, John Joon Young Chung, Christopher Brooks, and Steve Oney. 2021. PuzzleMe: Leveraging Peer Assessment for In-Class Programming Exercises. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW2, Article 415 (October 2021), 24 pages. https://doi.org/10.1145/3479559

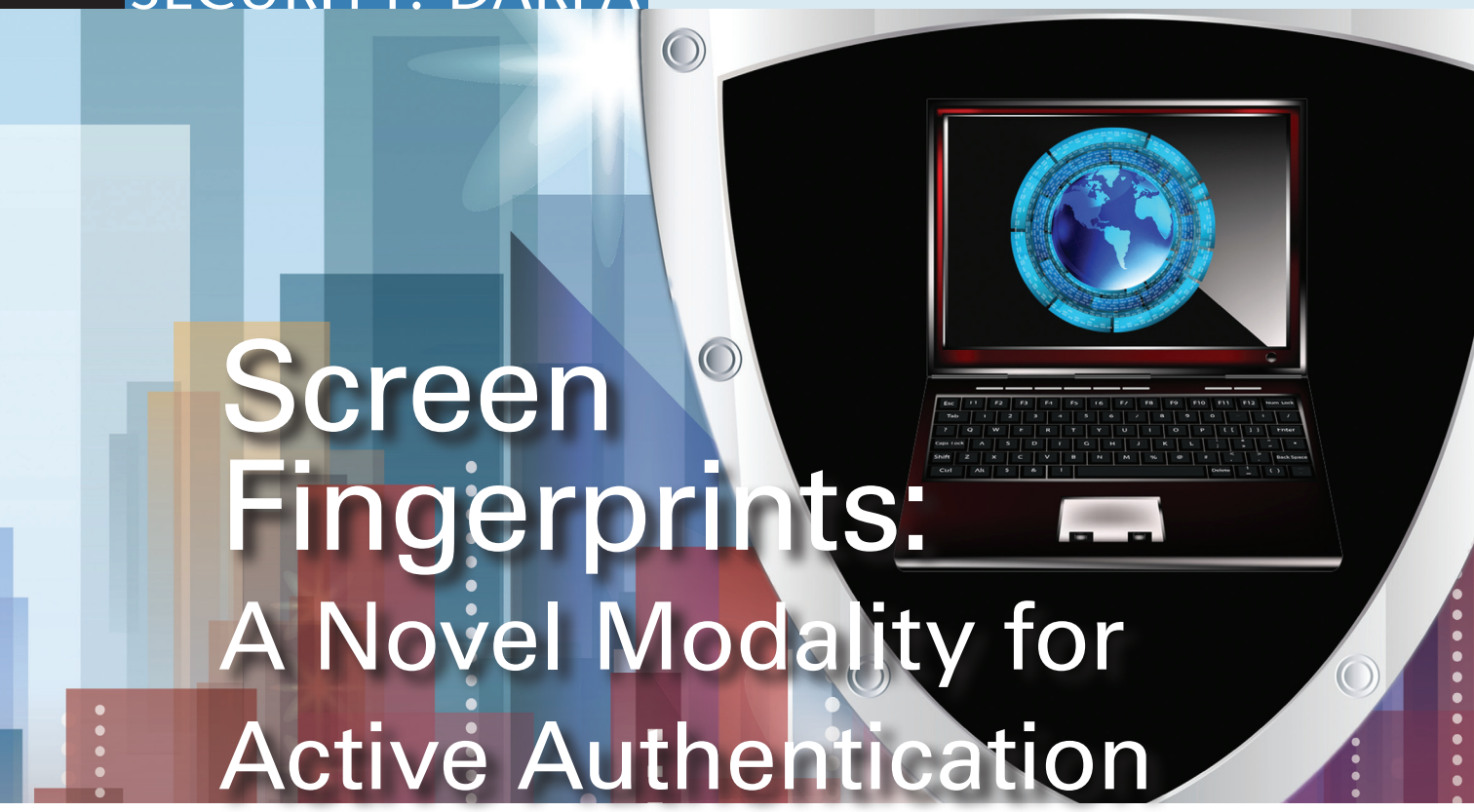
1 INTRODUCTION

Collaborative learning actively engages students to work together to learn new concepts, solve problems, and provide feedback [58]. Programming instructors often use various collaborative

*Both authors contributed equally to this research.

Authors' addresses: April Yi Wang, University of Michigan, Ann Arbor, Michigan, USA, aprilww@umich.edu; Yan Chen, yanchenm@umich.edu, University of Michigan, Ann Arbor, Michigan, USA; John Joon Young Chung, jjyc@umich.edu, University of Michigan, Ann Arbor, Michigan, USA; Christopher Brooks, University of Michigan, 105 S State St., Ann Arbor, MI, 48103, USA, brooksch@umich.edu; Steve Oney, University of Michigan, 105 S State St., Ann Arbor, MI, 48103, USA, soney@umich.edu.

2573-0142/2021/10-ART415 $15.00
https://doi.org/10.1145/3479559



Screen Fingerprints: A Novel Modality for Active Authentication

Vishal M. Patel, *University of Maryland, College Park*
Tom Yeh, *University of Colorado, Boulder*
Mohammed E. Fathy and Yangmuzi Zhang, *University of Maryland, College Park*
Yan Chen, *University of Colorado, Boulder*
Rama Chellappa and Larry Davis, *University of Maryland, College Park*

A screen fingerprint is proposed as a new biometric modality for active authentication. Such a fingerprint is acquired by taking a screen recording of the computer being used and extracting discriminative visual features from the recording.

We propose a novel way of validating the identity of the person at a console by using a *screen fingerprint.* This new cyberbiometric can help measure and analyze active authentication. We acquire a screen fingerprint by taking a screen recording of the computer being used and extracting discriminative visual features from the recording.

The screen fingerprint of an operator captures enough unique human qualities for use as a biometric for authentication. The qualities captured include cognitive abilities, motor limitations, subjective preferences, and work patterns. For example, how well the operator sees is a cognitive ability that can be captured visually by the size of the text shown on the screen. How fast the operator drags a window is a motor limitation that can be captured visually by the amount of motion detected on the screen. How the operator arranges multiple windows is a preference that can be captured visually by the layout of salient edges identified on the screen. What suite of applications the operator uses is a work pattern that can be captured visually by the distribution of application-specific visual features recognized on the screen.

The proposed technology exploits the synergy between recent advances in pixel-level screen analysis[1–3] and vision-based biometrics, such as

 Published by the IEEE Computer Society