Mocking-up Desired UI Behaviors from UI Element-Based Recording

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

I. Introduction

Software developers often ask for support from other developers, but effective communication about programming problems can be challenging. In the context of user interface (UI) development, effective communication about interactive behaviors of a UI is particularly difficult as it often requires a visual demonstration of the UI behaviors as supporting context. My motivational study found that our participants can always correctly understand a request when it includes video demos of the problem UI behavior and desired UI behavior. I summarized that an ideal request regarding a UI interactive behavior problem should include interrelated natural language description, relevant code, and demonstrations of non-desired and desired UI behaviors. I observed that developers often provide only the demonstration of not desired UI behavior and then describing the desired behavior on top of it. Unable to provide desired UI behaviors makes communication about UI behavior ineffective, and I argue this is a limitation of existing techniques. In this work, I would like to propose a solution to address it.

Existing tools, such as online discussion forums, allow developers to take multiple screenshots or insert examples via external links (e.g., jsFiddle.com) to provide visual context. However, capturing these behaviors often requires a sequence of user inputs which can be difficult to express in a static and linear form (e.g., text or screenshots) that is easy to understand. Prior work has argued that online discussion forums should be domain-specifically designed for more effectively communication [1]. Video recording these behaviors could be a solution, but requesters might miss necessary contextual information in their description. For example, our preliminary study found that when describing a desired behavior for one UI element, participants often forgot to provide the constraints for how other UI elements' should behave. Video recordings could support better communication of ideas with appropriate visual context, but the content in videos (e.g., UI elements) are not reusable.

II. Motivational Studies

To find more evidence of this problem, I conducted an in-lab exploratory study with seven undergrads from my university. Each session lasts for an hour. All of them had at least one year of experience developing interactive UIs and had completed a team project in their UI development class. I

978-1-7281-0810-0/19/$31.00 ©2019 IEEE

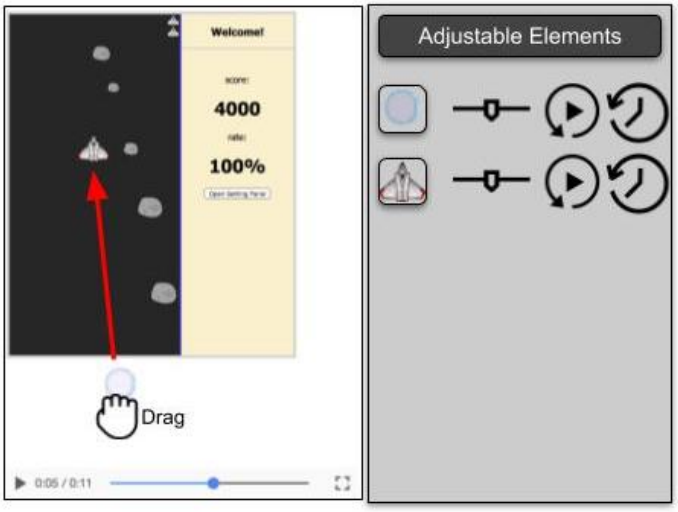

Fig. 1. CoCapture user interface. On the left is the recorded demonstration of non-desired UI behaviors. On the right is a list of DOM elements that are recorded independently. As each element is recorded as its own demonstration, developers can edit their properties, like speed, or location, independently, and remix the elements to reconfigure the desired UI behaviors.

asked three of them to act as a requester and make requests about a set of common UI behavior problems found on a web UI course (UMich EECS493) using a commercial tool, Scrimba [2]. We used Scrimba as it captures users' keystroke level activities and cursor movement on its editor, which provides more information in the recording. Participants were provided code, incorrect output, and the desired output. I asked the remaining four participants to act as helpers, to comprehend these requests and describe what they understood about the request. I then showed them the problems that were presented to the requesters and ask them to evaluate the requests by comparing their understanding to the ground-truth. In summary, I found that

- helpers can correctly understand all the requests when *both* a natural language description and video demos of incorrect and desired UI behaivor outputs were given,
- without being able to record the desired behaviors, requesters would often forget to describe some information in the requests,
- without videos of the desired behaviors, helpers would inaccurately understand the requests.

Based on these findings, I am proposing a new system to make it easier to generate desired UI behaviors. One idea that

I have developed is to enable developers to capture a UI-element-based (UIEB) recording of their UI interactive behaviors. Unlike a video recording, UIEB captures DOM object behaviors at the UI element level, which allows requesters to edit UI elements at any time-frame in the recording. By manipulating and remixing elements, requesters can mock-up their desired behaviors from from the recording.

To illustrate the potential user interactions I envision, consider the case of Kyle, an online game developer working on a blasting game project using CoCapture, the proposed system sketched in Fig. 1. On the left of Fig. 1, Kyle records a recording of non-ideal UI behaviors using CoCapture, where the shield (blue bubble) is supposed to overlay on top of the ship to protect it, but appears in the wrong position. On the right of Fig. 1, CoCapture shows a list of UI elements that the requester wants to manipulate, which they select or upload. After recording, Kyle edits the UI elements's from the recording, such as positions, play speed, or size, to mock-up the desired behaviors. For example, in Fig. 1 Kyle pauses the recording at the point where the incorrect behavior starts, and he can now directly drag the shield object to overlay it on top of the ship object. To propagate this overlay behavior, CoCapture allows the requester to add an object constraint between the shield and the ship and make them share the same trajectory within any range of the video.

III. Communication about code

To make software development more effective, I have looked at the challenges and needs that developers encounter during programming support [3]. Through multiple motivational studies, I found strong evidence for a variety of communication challenges with current technologies, such as lack of code context for mutual understanding. Driven by the design implications I drew from these studies, I created Codeon [4], a collaborative programming support tool that enables more effective programming task hand-off between developers and remote helpers with multimodal input interaction techniques. When designing Codeon, one goal was to make communication between requesters and helpers about code syntax easier. For example, Codeon supports voice requests, and the request audio recording is synchronized with the developer's interactions within the editor (e.g., highlighting, scrolling, file switching) and can be replayed in the helpers interface. The multimedia request jointly embodies the dynamics of voice signals and the visual content references, providing a rich, natural context for communication. While Codeon is effective for communication about code, it does not support communication about program output, like user interface behaviors. The dynamic behavior of modern-day UIs introduces new challenges in communication when coupled with code syntax and natural language, which require further understanding.

IV. Communication about complex tasks

Other than Codeon, much prior work has also studied how to better capture context when communicating about complex tasks. Systems like Apparition [5] and SketchExpress [6] enable UI prototype designers to easily hand-off their UI and animation requirements to crowd workers using natural language, hand-sketches, and demonstration. However, they require users to create prototypes from scratch. CoCapture, instead aims to enable requesters to easily mock-up their desired behaviors by editing elements from the captured recording.

V. Future Work

As I continue to explore the design space of user interactions for manipulating and remixing elements in a given UI-element-based recording, there are several questions I plan to answer: *What would requesters want to capture? What would requesters want to edit? Does it help to capture element transforms as well? Does it help to show the difference before and after editing? What element properties do requesters want to manipulate? How can we support adding new elements to the recording? How could requesters add their behaviors and synchronize with the recording? If inserting UI elements is possible, how would requesters want to specify the behaviors of the new elements in recording?* I will first come up with some variations of the features I described above, and then evaluate their usability.

After analyzing the collected feedback, I plan to develop a working prototype that provides all the critical functionality, including capturing UI behaviors as a recording at the UI-element level and manipulating elements in the recording. After pilot testing, I plan to conduct a system evaluation study in which participants will use either CoCapture or existing tools (e.g., scrimba.com) to make and read requests. I will collect qualitative data such as their opinions regarding the ease of use, and quantitative data such as the time they spend on creating and understanding the requests, or patterns for effectively creating desired behaviors for a given codebase state (feature non-existent vs feature in-progress vs feature broken). I am excited about developing this UI-element-based recording technique to support people in better expressing desired interactive UI behaviors based on existing recording.

References

[1] J. Zhu, J. Warner, M. Gordon, J. White, R. Zanelatto, and P. J. Guo, "Toward a domain-specific visual discussion forum for learning computer programming: An empirical study of a popular mooc forum," in *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2015, pp. 101–109.

[2] Scrimba. https://scrimba.com/, 2019.

[3] Y. Chen, S. Oney, and W. S. Lasecki, "Towards providing on-demand expert support for software developers," in *Proceedings of the 2016 CHI conference on human factors in computing systems*. ACM, 2016, pp. 3192–3203.

[4] Y. Chen, S. W. Lee, Y. Xie, Y. Yang, W. S. Lasecki, and S. Oney, "Codeon: On-demand software development assistance," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2017, pp. 6220–6231.

[5] W. S. Lasecki, J. Kim, N. Rafter, O. Sen, J. P. Bigham, and M. S. Bernstein, "Apparition: Crowdsourced user interfaces that come to life as you sketch them," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 1925–1934.

[6] S. W. Lee, Y. Zhang, I. Wong, Y. Yang, S. D. O'Keefe, and W. S. Lasecki, "Sketchexpress: Remixing animations for more effective crowd-powered prototyping of interactive interfaces," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, 2017, pp. 817–828.