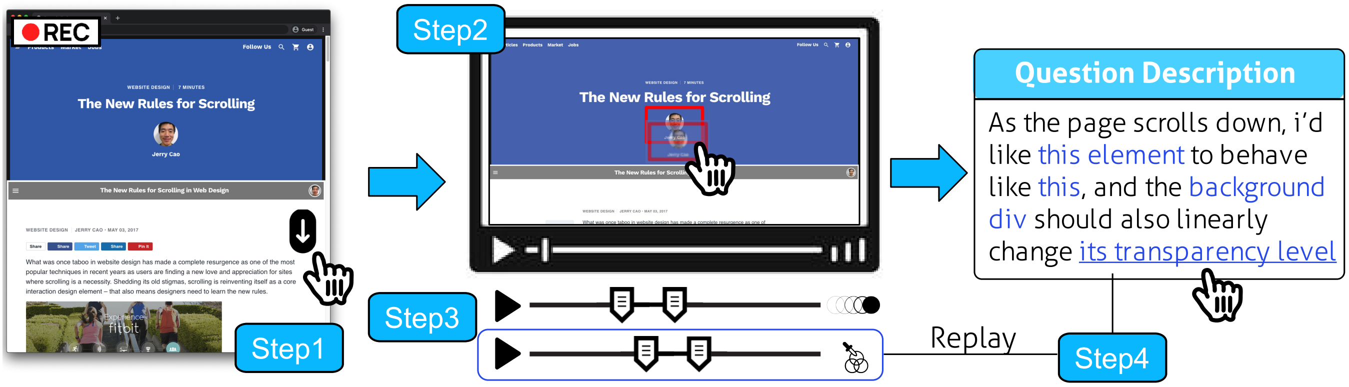
CoCapture: Effectively Communicating UI Behaviors on Existing Websites by Demonstrating and Remixing

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



As the page scrolls down, i'd like this element to behave like this, and the background div should also linearly change its transparency level

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

propose changes to a UI that already works or to describe a desired behavior in a UI that contains an error. First, most tools for creating mockups cannot import assets or behaviors from existing UIs, and it can be tedious to replicate the intricate details of a working UI in a mockup. Second, it can be difficult to communicate how an existing UI should change because it is not easy to point out the difference between the existing behavior and the mockup's behavior—particularly when the change is nuanced or dynamic [43, 48, 51]. Third, mockups that propose changes to existing behaviors often need to be *mixed fidelity* [40]—with high-fidelity representations of existing components and low-fidelity renderings of proposed changes—but few mockup tools support this. These limitations led to our research question: ***How can we make communicating about changes to existing UIs easier and more effective?***

In this paper, we introduce CoCapture, an interactive system that enables users, like UI designers, to **easily create** and then **accurately describe** dynamic UI behavior mockups. These mockups could represent changes the users want to propose or questions they want to ask about an aspect of the existing UI. With CoCapture, users first record the existing UI behavior by *demonstrating* an example interaction on the existing UI (Fig. 1, Step 1). Building on this scene, users can further create dynamic behaviors via *demonstrations* that manipulate DOM elements (Fig. 1, Step 2) and *remix* these demonstrations as a first-class animation object (Fig. 1, Step 3) through direct manipulation and low-fidelity sketching. To help accurately specify the visual changes, users can write Natural Language (NL) descriptions in CoCapture that contain hypertext references to specific aspects of the mockups (e.g., specific DOM elements, new animated effects) (Fig. 1, Step 4).

We conducted two within-subjects studies to evaluate the communication effectiveness of multiple aspects of CoCapture: the effort of creating visual context and the accuracy and clarity of the description. In these studies, we asked "requester" participants to describe a UI behavior and "helper" participants to read the descriptions that requester participants generated. Our results show that compared to traditional sketching and communication tools, the requester participants using CoCapture spent less than a third of the time on text writing, and their descriptions of UI behavior were significantly more accurate. Additionally, the helper participants reported that the descriptions in CoCapture were more accurate, concrete, vivid, and easier to follow.

The **key contribution** of CoCapture is a novel interactive method that combines the DOM element-based recording technique with a demonstrate-remix-replay approach. This makes it easier to prototype on pre-built UIs and to describe user needs regarding dynamic UI behaviors more accurately than is possible with existing approaches. With CoCapture, users can effortlessly explore different possible designs, capture fleeting ideas, and communicate with others about behavior ideas on existing interfaces. Specifically, our contribution includes:

- A set of novel interaction designs and techniques that allow users to capture, demonstrate, remix, and then describe the UI behaviors they want to add or inquire about on an existing UI via direct manipulation.
- CoCapture, a system that integrates all these techniques to make communicating about changes to existing UIs easier and more effective.
- Evidence showing that CoCapture can help designers more easily create UI behavior descriptions that are easier to understand and follow compared to those created by existing approaches.

2 RELATED WORK

As Myers et al. [43] explain, interactive behaviors define the "feel" of a UI (as opposed to its "look"). Tools like VisBug [3] and Poirot [57] help designers quickly change the look of their UIs. We focus on communicating about the feel of the UI. This process often consists of two tasks: making visual references and referring to the visual references. In this section, we review related work and techniques in these fields.

2.1 Creating UI Prototypes and Mockups

Systems like SILK [31] and DENIM [39] lower the overhead cost of prototyping by recognizing designers' sketches as interface elements and implementing the idea of wireframing, respectively. However, they do not support creating prototypes in later stages of UI development. More recently, tools like Rewire [56] and Poirot [57] have made prototyping new designs easier by enabling the users to directly edit elements of existing examples. However, they do not support interactive UI behavior editing, which is more difficult than designing static layouts, as the behaviors are complex to demonstrate and designers have access to limited tools [51]. Commercial tools like Figma [2] and Adobe XD [1] can ease the creation of interactive behaviors but assume their users would reconstruct existing interfaces from scratch, making it hard to scale. Other layout-capturing tools such as WebToLayers[1] and PageLayers[2] automatically convert websites to Photoshop documents. However, they only support static layouts; they do not preserve the DOM structure, element constraints, or dynamic UI behaviors. Crowd-powered systems like Apparition [32] and SketchExpress [34] allow designers or even non-experts to more rapidly create or reconstruct a prototype than they could with existing tools, but these systems also fall short of recreating particularly complex interfaces.

Unlike these systems, CoCapture helps create interactive mockups in the later stages of UI development, proposing changes to a UI that already works or describing desired behaviors in a UI that contains an error. These UIs can be arbitrarily complex, requiring effort to create mockups that replicate existing functionality. Compared to layout-capturing tools, CoCapture also captures the DOM structure, allowing designers to easily add behavior mockups on existing interfaces. This gives them the ability to immediately envision the new behaviors and complete the process of creating a mockup as a reference.

2.2 Visual References as Shared Context in Communication

Many prior studies have reported that people often include screenshots, drawings, or sketches as visual context in communication.

[1] https://neededapps.com/webtolayers/
[2] https://www.pagelayers.com/

These studies have explored questions of how designers communicate desired interactive behaviors to engineers [43, 51], how InfoVis novices describe data visualization [20, 42], how programming novices explain PC game behaviors to the computer [49], and how end-user developers communicate about application extensions with other developers [17]. However, they also consistently found participants' responses to be vague, ambiguous, or imprecise, suggesting future systems should provide a tight feedback loop in which users see immediate results to refine ambiguous descriptions.

Creating visual references can help ground communication when discussing visual design and providing feedback. In a face-to-face or video conference setting, we can use pointing gestures in shared visual spaces to make references to visual information that is difficult to express with words. Much work has studied methods for referring to visual content in communication, including text annotation [44], remote gestures [19, 27, 28], and awareness widgets [13, 18] in different computer-supported cooperative work contexts such as authoring [41, 55, 60–62] and groupware [23, 24]. They have shown that referring methods can facilitate mutual understanding by reducing verbal effort and its associated complexity [22]. The ability to leverage non-verbal communication is an important factor in decreasing the effort of writing clear messages [16].

In programming communication, systems like chat.codes [45] and Callisto [59] use deictic code pointing techniques to facilitate creating code references and connections with text descriptions that help developers discuss code. Codeon [12] is an in-IDE support environment to help requesters and helpers exchange code context easily. MarmalAid [15] allows users to start a real-time conversation on a geometric location in a 3D workspace. Building on these approaches, we aim to address the problem of referring to dynamic and interactive visual references for more effective communication.

2.3 Record, Replay, and Manipulate Existing Interfaces

A core technical part of CoCapture's system is the record and replay (R&R) technique, which is used to record an existing behavior once and then replay it repeatedly and automatically without user interaction. Prior work has used this technique for various purposes. Systems such as Scry [7], Telescope [26], Unravel [25], Doppio [14], FireCrystal [47], and WebCrystal [9] use this approach to help people understand existing UI behaviors. Systems like Chronicle [21], Timelapse [6], and MobiPlay [53] record meta-data (e.g., operations, code editing) and allow users to easily capture the rich data of application behaviors. Our techniques enable designers to not only record arbitrary web interface behaviors, but also to easily add new designs on top.

To ease the creation of new behaviors, FrameWire [38] decreased the effort of communicating new interactive designs by automatically extracting interaction flow from paper prototype video recordings. Many other recording augmentation systems have been developed to help effectively prototype new digital content, such as Montage [35] and Augmented Reality (AR) experiences like Proton [36], or to provide video feedback, like VidCrit [52]. Park et al. developed a technique that enables users to edit text content in a text-based recording while preserving the recording's overall consistency [50]. CoCapture also allows its users to easily add new behaviors over recordings, but instead of simply supporting overlaid visual annotations or user comments anchored to specific parts of the content, it allows users to edit existing elements through direct manipulation.

The field of Programming by Demonstration (PbD) has used similar techniques to augment users' ability to perform tasks with which they often lack expertise. Rousillon [11] and Sugilite [37] allow their users to record and edit the operations via domain-specific languages. CoCapture also supports a set of mockup creation operations that allow designers or even novice users to edit the recording and simulate the desired interactive behaviors.

3 NEEDFINDING STUDIES

We conducted two studies to better understand designers' challenges and needs when communicating about UI behaviors.

3.1 Stack Overflow Analysis

We first conducted an analysis to identify the categories of questions related to UI behaviors that were most frequently asked on Stack Overflow (SO), a well-established Q&A platform in the programming community. We analyzed the 200 most viewed questions that were tagged with JavaScript (JS) and CSS. We used these two tags because they are primarily used for manipulating UI elements (JS) and presenting them (CSS). We read through all the posts, counted other included tags, and documented the use of visual references in the posts. We found that 41 posts (20.5%) included tags that were CSS properties (e.g., height, position) or related to interface element changes (e.g., CSS-transition, sticky menus). We also examined the visual references included in each post and found that 34 posts (17%) included links to a live example of their problem[3], 18 posts (9%) included screenshots, and 5 posts (2.5%) included sketches. This helped us determine which of the most frequently asked question categories also required visual information. We used this information to guide our system and study design.

3.2 Needs and Challenges

Our second study examined how well existing tools can support communication about common UI behavior issues on existing web interfaces. We recruited eight participants with at least one year of UI design and development experience (3 female, 5 male) from the first author's university. Among them, half acted as requesters to ask three questions using Scrimba [5], a state-of-the-art tool that allows its users to simulate in-person communication by recording voice narration and editor activity (e.g., typing, highlighting). The three questions represent the most popular categories we found in our Stack Overflow analysis: a responsive UI task[4], a platform game[5], and an animated effect applied to an object[6]. The remaining participants acted as helpers, reviewing the clarity of the requesters' three questions by comparing their understanding of the requests to the ground-truth desired output (presented as a video demonstration). All helpers were teaching assistants for a UI development course at the first author's university. After the tasks, we held a

[3] https://stackoverflow.com/a/{24414642|5445491|17722497}
[4] https://tinyurl.com/ydev4uwr
[5] https://github.com/starzonmyarmz/js13k-2018
[6] https://semantic-ui.com/modules/transition.html

follow-up interview with all participants in order to help inform the design of CoCapture.

Instead of describing the desired change in the interactive behavior in text, which would result in biases [49], we gave requesters the code (i.e., HTML, JS, CSS), the existing UI, and the desired UI for each task. We also used visual annotation (e.g., cursor pointing) and determiners (e.g., "this part") to point out the differences between the existing and desired UIs without offering specific descriptions. Requesters could access this information throughout the session and were not allowed to use any materials from the desired UI artifact (e.g., no screenshot of the desired UI). This simulates the scenario where a UI designer knows what the desired change is but has no access to the new behavior.

The lead author conducted all of the studies in her research lab. Each session lasted approximately 45 minutes, and sessions were recorded and transcribed. The lead author went through the transcripts and coded them using an open coding approach [10], which included discussions with the research team. We report our findings for requesters [R] and helpers [H] below.

Challenge 1 (C1): [R] Visual context is necessary but difficult to describe on existing interfaces. When asked about their experience describing the UI questions on the given artifacts, all requesters noted the challenge of providing visual information in the request: *"I wish there was an easier way to describe the difference between what's there [existing UI] and what's needed to be there [desired UI]"* (R1). When asked about their experience with UI behavior questions on Stack Overflow, requesters also expressed the necessity of including the contextual information of visual changes as part of the requests. *"I'd prefer to create a video to demonstrate the whole changing process"* (R4). When asked about their needs when creating the requests, R1 suggested having a better way of organizing the information. This indicates the importance of providing visual information changes in a request, as well as a need for easier methods of adding visual information changes to existing interfaces.

Challenge 2 (C2): [H] NL descriptions may lack the necessary details. Four helpers commented that some of the questions were *"too vague"* (H1), *"broad"* (H2, H3), *"unclear"* (H4), or *"hard to understand"* (H2, H3). For example, H2 said, *"I don't understand when they say their HTML pages [are] not responsive, because it is responding to the way they're resizing."* To respond to these "vague" questions, two helpers said they had to provide suggestions based on their best understanding of the questions (H1, H2), while the other two would prefer to follow up with clarifying questions to make the expectations more explicit (H3, H4). This suggests that the current tools might not provide sufficient support for beginners to easily phrase their questions and receive the correct assistance.

Challenge 3 (C3): [R, H] Voice-based video requests can be tedious to make and navigate. Both requesters and helpers reported that they were more used to tools with text (e.g., email and text-heavy images like annotated screenshots) for asynchronous communication, and they found voice-based requests difficult to use. Requesters felt creating voice annotation was *"time-consuming for people who have to think long [about] what they have to say"* (R1), *"hard to synchronize everything together"* (R2), and *"hard to edit later and so that kinda make me more nervous to do"* (R3). The other requester (R4) felt voice recordings were an easy way to annotate visual requests. Helpers commented that the voice requests were *"helpful"* (H2), though they wish the recordings could be *"more slowed down"* (H1) and *"easier to find information like searching through text"* (H3). This feedback suggests that text-based requests can be more convenient for requesters to make and can enable helpers to find information more easily. However, the difficulty in referring to dynamic behaviors and visual elements remains a challenge. We aim to address this in our system.

Need (N): [H] Wish to see the existing and desired behaviors. When asked about the necessary information for understanding a UI behavior question, all helpers chose the existing output and desired output. Half of the helpers felt the related code would not be necessary, as *"there're so many ways of doing something. As long as they're able to understand what the results should be, I can figure out what the code is supposed to do"* (H2). This indicates that both the existing and desired behaviors should be included in a UI behavior question. After being shown the ground truth of the desired behaviors, three helpers found that they had misunderstood at least one question, noting that *"I thought it was going to be this whole block that was moving together"* (H2) or that *"I was actually thinking that this bar here would stay on top above this"* (H3). Helpers suggested creating a video that scrolled through all of the different sketches and screenshots, a concept that aligns with suggestions from prior work [51]. These results indicated the need for a tool that highlights the moving elements that respond to the UI behaviors.

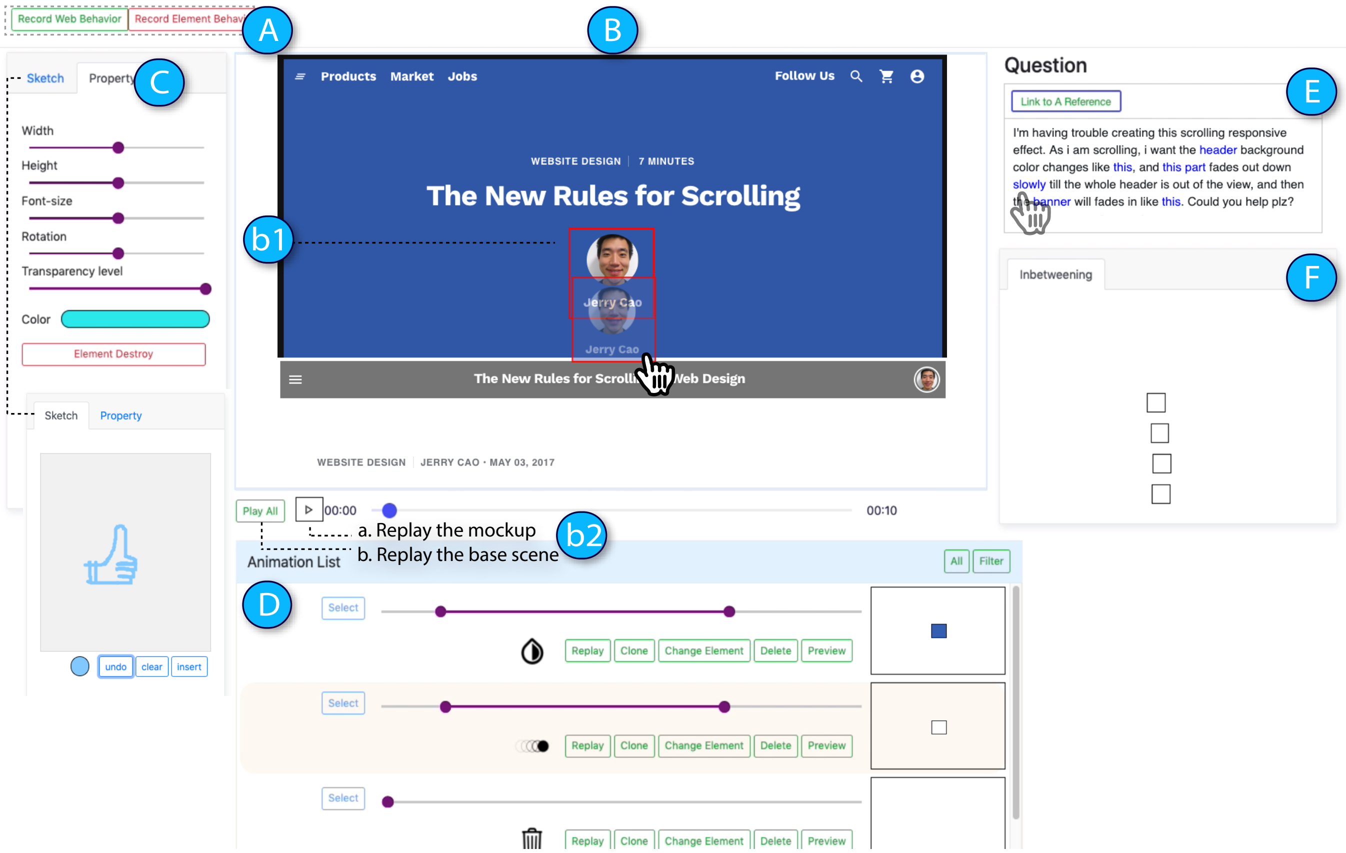
3.3 Design Goals

Driven by the findings (C1–C3, N), we came up with three goals (DG1–DG3) to guide us in designing CoCapture so that requesters can easily prototype behaviors on pre-built interfaces and effectively communicate about the new behaviors with helpers.

- **DG1: Quick to create visual changes (C1, N):** Requesters need to quickly and accurately create their desired UI mockups on existing interfaces and include both the existing and desired behaviors for communication.
- **DG2: Accurate to describe behavioral information (C2, N):** Requesters need to easily and accurately describe the desired behaviors with the created mockups, not just the desired appearance.
- **DG3: Easy and clear to understand the needs (C2, C3):** Helpers need to easily and clearly understand the existing and desired UI behaviors, with each behavior being specified accurately.

4 COCAPTURE DESIGN AND IMPLEMENTATION

Guided by the design goals, we developed CoCapture, a Chrome extension that allows users to easily create mockups on an existing website through direct manipulation and to reference pieces of the resulting mockups in their description using hypertext. In this section, we first illustrate the experience of using CoCapture with a sample usage scenario that embodies many of the use cases identified in our formative studies. We then detail the design of CoCapture.



I'm having trouble creating this scrolling responsive effect. As i am scrolling, i want the header background color changes like this, and this part fades out down slowly till the whole header is out of the view, and then the banner will fades in like this. Could you help plz?

Figure 2: CoCapture's main panel (after Step 1 in Fig. 1). To create an animation, a user first clicks on an existing DOM element (b1) (can select multiple by holding Shift key down) from the base scene (B). Once recording starts (A), the user can demonstrate the desired behaviors by changing the elements' properties (C) or directly dragging them in the base scene. Once the recording is finished, the demonstration will be appended to the Animation List (D) with a set of meta operations, such as adjust start/end time, replay, and preview. The user can write their question (E) and refer to the animations or DOM elements in the scene by selecting portions of text and clicking the 'Link to a Reference' button. Text with references, or hypertext, is in blue with a click affordance displaying the relevant context (i.e., highlight elements (b1), replay animations (D)). The process of the animation (inbetweening) will also be displayed (F). The user can also filter the Animation List (D) to only display the animations related to the selected elements.

4.1 The CoCapture User Experience

Jerry, a junior professional web interface designer, is in the middle of prototyping a website that has most of its content ready. Now he wants to extend the website by adding some interactive behaviors to respond to a user scrolling event (e.g., the user profile picture fades out down[7]). He decides to use CoCapture to create a mockup of his vision and then ask his peers for feedback. Jerry first clicks the Chrome extension on his website to start CoCapture (Fig. 2) and clicks the "Record Web Behavior" button (Fig. 2.A left) to record a demonstration of him scrolling through the original website as if he were the user. Once finished, he clicks the play button (Fig. 2.b2a) to watch the replay of his demonstration (Fig. 2.B), verifying all the relevant context is captured.

To create a mockup where the profile picture fades out down with the appropriate speed and distance as a user scrolls down the whole page (Fig. 2.b1), Jerry first moves his cursor over the image until he sees a dotted border highlight the right element (e.g., not the element that wraps the image). After selecting the element, Jerry presses the "Record Element Behavior" button (Fig. 2.A) and directly demonstrates the desired fading behavior. This behavior requires remixing two independent animations: the element moves down until the bottom half of the image is covered by the gray banner, and the transparency level of the element continuously decreases to half of its original value. Jerry creates these two animations by direct manipulation, seeing the changes immediately: he drags the element to the desired position (Fig. 2.b1) and adjusts the "Transparency level" slider value to half of its original value (Fig. 2.C).

[7] A similar fade-out example: https://html5up.net/massively

The created animations are added to the "Animation List" (Fig. 2.D). Each animation has a set of operators, including an interactive range slider that represents the start/end time, replay, clone, delete, and preview. To set the recorded fading behavior to occur between the time when a user starts scrolling and the header moves out of the view, Jerry scrubs the replay play bar to find these two moments from the base scene recording and then adjusts the animation sliders to align with them.

After Jerry finishes creating the mockup, he adds a text description to his request, which includes references to the relevant DOM elements and animations (Fig. 2.E). Because Jerry knows he can use hypertext to link his description to the relevant artifacts, he writes a very short message and uses pronouns such as "*this* element" or "behave like *this*." Jerry selects part of his description and clicks "Link to a Reference" to select the DOM element or animation that he wants to reference. To ensure the animation matches what he has written in the text, Jerry clicks the hypertext highlighted in blue and reviews all of the animation replays—both in the scene (Fig. 2.B) and in the "Inbetweening" panel (Fig. 2.F).

4.2 Design and Implementation

We describe the technical details of CoCapture in this section.

4.2.1 Step 1: Demonstrating Existing Behaviors. To quickly create visual context on top of an existing interface (DG1), CoCapture allows users to capture a behavior by demonstrating it on an existing website. This saves significant time and effort in creating a shared visual context—which we will call a *base scene*—as users can import arbitrarily complex behaviors from any website rather than needing to create animated visuals from scratch. When users demonstrate existing behaviors on a website, CoCapture captures all DOM changes (e.g., node creation, deletion) and events (e.g., mouse movement, browser window size changes). To capture this data, we rely on an open-source library[8], which in turn uses MutationObservers to track [4] and store the timeline of DOM changes. The serialized DOM change sequence can be replayed on CoCapture's main panel as if it was a screencast (Fig. 2.B). However, unlike a screencast (pixel-based), each frame in the replay still preserves the DOM tree structure from the original interface.

4.2.2 Step 2: Animating the Desired Behaviors. CoCapture includes a prototyping environment that allows users to demonstrate their desired UI behaviors as animations (DG2). Users can modify the replay of the existing behavior by directly manipulating the UI elements in the base scene, recording their changes, and augmenting the base scene with these demonstrations. The reconstructed DOM recording (i.e., the base scene) also preserves the UI states at each time point of the demonstration (e.g., DOM structure). It accomplishes this with the following steps and techniques.
(Manipulating and adding UI element(s) in the base scene). CoCapture transforms all the UI elements from the original website into selectable elements that a user can directly manipulate. Users can select one or more DOM elements in the scene by holding the `Shift` key down. As users hover over each element, CoCapture highlights the element with a red dashed border to ease the selection process (similar to the element selection feature in the Chrome Developer Tool). CoCapture also allows users to create low-fidelity sketches (which it stores as Scalable Vector Graphics (SVG) drawings), import sketches into the scene, and manipulate them like any other DOM element (Fig. 2.C).
(Record a desired behavior as a behavior mockup). CoCapture uses the MutationObserver API to record the attribute changes (e.g., style changes) of the selected elements. Once they have started recording, users can edit selected elements' CSS properties by adjusting the sliders in the relevant side panel tab (Fig. 2.C). They can also perform drag-and-drop operations on any DOM elements to demonstrate their new motion and position. All the manipulations will be represented immediately in the base scene. CoCapture automatically records and stores a continuous series of time-stamped snapshots, which will later be replayed as an animation.

CoCapture supports the modification of 10 commonly used element attribute types, including height, width, font size, rotation, transparency level, color, hide (delete), visible, x, and y. Future work can build on this list by connecting with Chrome Developer Tools or by extracting the existing properties of each element [57].

4.2.3 Step 3: Remixing Added Animations. To help users accurately express and see the desired and existing behavioral information (N, DG2), CoCapture provides a set of features that allow users to remix, review, and fine-tune animations. The design of these features is inspired by the concept of "remixing," an idea widely used in animation creation [34] and music editing [33]. First, each animation is listed below the base scene recording, with the ranges aligning with the exact moments the animation was started or stopped with respect to the base scene recording timeline (Fig. 2.D). Second, the set of operations for each animation (the green buttons in Fig. 3) facilitates creation and adjustment of the animation. Users can replay the base scene (Fig. 2.b2.b), the remixed behaviors (Fig. 2.b2.a), and each animation solo (Fig. 3 "Replay" button) at any time. We describe the details of each feature below (all within the "Animation List" section in Fig. 2.D).
(Time and duration adjustment play bar) The interactive range slider for each animation represents the start and end times of the animation relative to the original recording. A user can move the two handles of the range slider to modify the start time and end time. Because the scene's play bar and the animations' range sliders are visually stacked and follow the same timescale, users can easily remix an animation to align with others in the list. As the range changes, the duration of the animation also changes linearly.
(Replaying and deleting individual animations) CoCapture allows users to replay and delete each animation by clicking the appropriate button below the range slider. When replaying, the elements that were selected when demonstrated will be highlighted with a solid red border (Fig. 2.b1). CoCapture replays an animation in two steps: it first resets the scene to the state at the start time of the animation, and then it replays the demonstrated changes to the elements. Resetting the scene is necessary because certain behaviors are state-dependent (i.e., a banner only appears when the page scrolls to the bottom). Being able to replay individual animations from the correct state helps users precisely envision and reason about further steps. In addition, demonstrated animations can be deleted individually from the scene. This is helpful if the designer

[8] http://rrweb.io

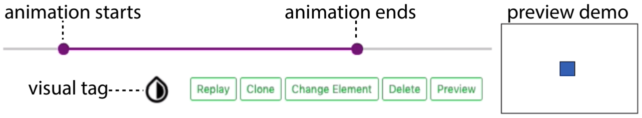
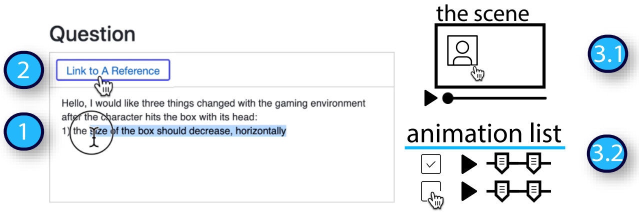


Figure 3: For each created animation, CoCapture provides information that prompts users about the actions they can take, including creating visual tags and replaying or previewing the animation. It also supports a set of operations to make the creation process easier and more accurate, including a range slider for time adjustment, as well as options for cloning, changing, or deleting an element.



Hello, I would like three things changed with the gaming environment after the character hits the box with its head:
1) the size of the box should decrease, horizontally

Figure 4: There are three steps to create hypertext: (1) select portions of the text in the question description, (2) click "Link to a Reference," and either (3.1) select an element in the scene or (3.2) check an animation. The hypertext will be highlighted afterwards.

notices a mistake in the demonstrated behavior and simply wants to redo it.

(Animations at a glance with visual tag and preview) To remind users of the animation type (i.e., edited element properties), CoCapture uses icon-like visual tags and a live preview for each animation. We use six different tags to represent the 10 different attribute changes. From top to bottom, Fig. 2.D illustrates that elements change in color (transparency level, color) ◑ and position (x, y, rotation) ●, and can be removed from the scene 🗑. CoCapture also includes resize (height, width) ⧉, font size AA, and visibility (sketch is added to the scene) 👁 icons. The preview feature (Fig. 3 "Preview" button and preview demo) is also designed to help make animations more glanceable and easier to understand by showing a simplified version of the actual animation (N, DG3). Upon clicking the "Preview" button, the preview demo will play a simplified version of a looping animation where each hollow square represents one relevant DOM element. This simplified animation preview design is inspired by Tufte's minimalism theory for effective information visualization [58]. For example, the first preview demo in Fig. 2.D is currently playing the change in the transparency level of the background element.

(Cloning and changing element) CoCapture stores the animated elements and their mutation arrays independently. This allows it to apply the same set of mutations to different elements. Users can easily create one animation and use the "Clone" or "Change Element" buttons to repurpose it so that the behavior of one element can be adapted to others.

(Filtering the animation) To help explore the mockup details, users can click to select an element (e.g., profile picture) in the base scene and click the "Filter" button (Fig. 2.D) to view only those animations that were created on this element.

4.2.4 Step 4: Communication with Visual References (Hypertext).

To effectively communicate about the interface behaviors (DG2, DG3), CoCapture provides a lightweight text box (Fig. 2.E, suggested in the needfinding results) where users can easily and accurately describe and review UI behaviors with hypertext, a feature that links text description to the visual information. Similar to the hyperlink feature in common text editors, a user may select portions of the text, click the "Link to a Reference" button, and select any DOM elements in the base scene or created animations to link to the text (Fig. 4). Upon clicking part of the text with hypertext, CoCapture highlights and replays the referenced visual context with a solid-colored border (Fig. 2.b1). Meanwhile, to make an animation more glanceable, we designed the "Inbetweening" panel (Fig. 2.F). This was inspired by the technique of *inbetweening* in computer graphics—the process of generating all the frames of a motion sequence given its first and last frames—that helps to accurately communicate about animations [8]. Unlike the preview feature introduced before, the "Inbetweening" panel presents four key frames of the referenced animation using simple linear interpolation in between. More advanced interpolation calculation algorithms could be applied in future work to make it more expressive [54]. Similar to the preview feature design, each key frame represents a simplified state of the relevant elements (e.g., color, proportional size) using a hollow square. For example, in Fig.2.F, the "Inbetweening" panel presents the four key frames of the profile image motion animation when the user clicks the hypertext attached to "slowly."

5 SYSTEM EVALUATION

We conducted two initial user studies to evaluate CoCapture's effectiveness to help users create, describe, and understand UI mockup questions. Primarily, we wanted to answer the following questions:

- Q1: Can designers ask a more accurate UI behavior question using CoCapture than with a text-based communication tool (e.g., email)? (Study 1)
- Q2: Can they also create the questions more quickly? (Study 1)
- Q3: Can helpers easily understand the questions in CoCapture? (Study 2)

6 STUDY 1 - CREATING UI BEHAVIOR QUESTIONS

To evaluate question creation, we designed a two-condition, within-subjects study. We recruited 15 participants (9 male, 6 female, age 25–30) from a local participant pool with an average of 3.5 years of UI prototyping experience. All participants had native or bilingual

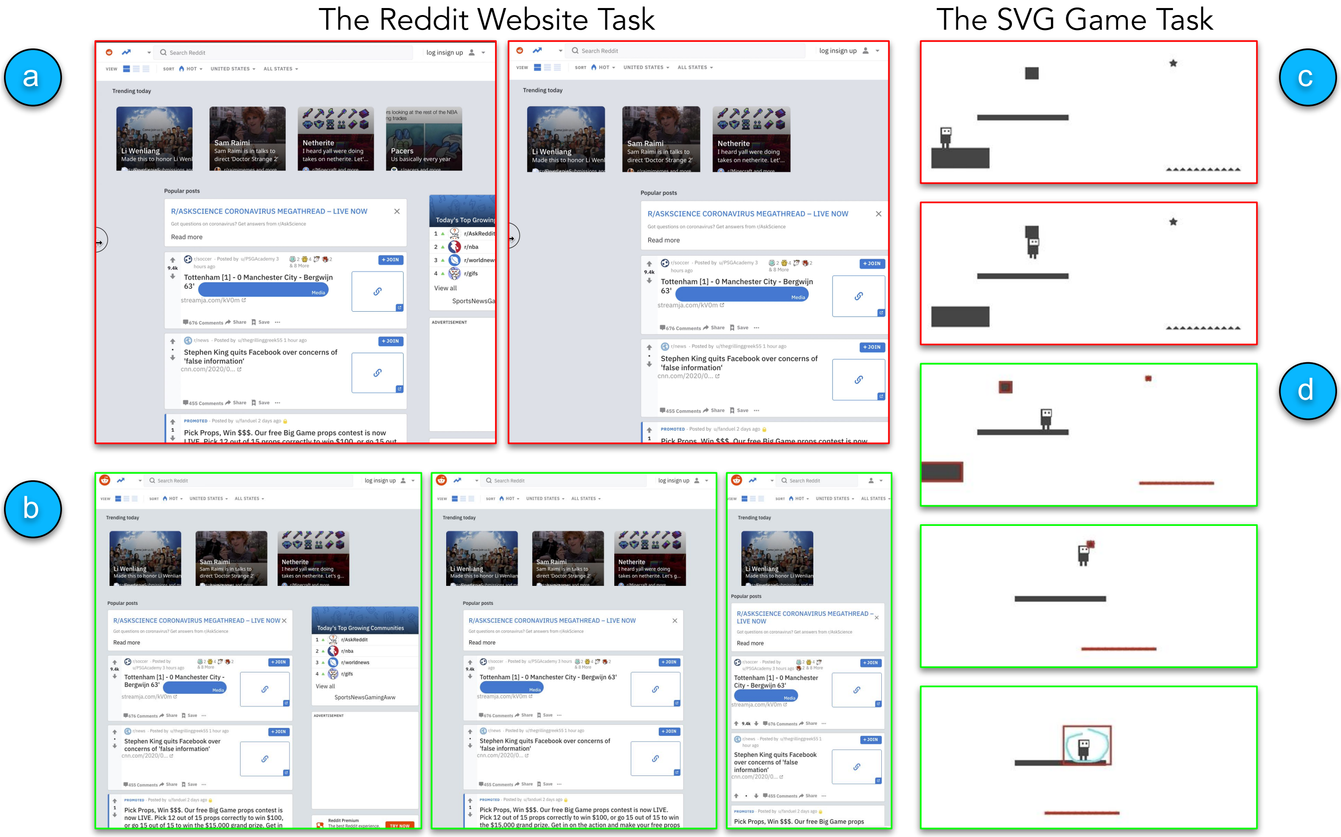


Figure 5: Screenshots of two of the user study tasks: the Reddit website (left column), and an SVG game (right column). The screenshots with a red border (a, c) are the existing UIs, whereas the green ones (b, d) are the desired UIs that the participants were asked to create and describe during the study. The tasks were designed with three goals: realistic, common, and complex. For the Reddit website task, the issue with the existing UI is that some of the DOM elements are not responsive to window resizing user input. For the SVG game task, participants need to add new game behaviors such that when the game character hits the block and the star shape element, the game scene will change dynamically and a new element (the cyan shield shown in the last image) will be added to the scene.

proficiency in English. Instead of using open-ended tasks, each participant was presented with four websites and asked to create one question per website regarding a predefined UI behavior issue. This can help us compare the description accuracy between conditions. They used either CoCapture or the tools in the control condition, in a randomized order. In the control condition, we asked participants to use Gmail[9] and Google Drawings[10] to compose their questions. We chose these tools because of their low learning curve and similar functionality to other prototyping tools. We only recruited people who had prior experience with both these tools. We chose these tools because of their low learning curve and similar functionality to those mentioned in the needfinding study. In the treatment condition, participants first watched a tutorial on CoCapture and replicated the example in the tutorial.

To recreate a situation in which the participants would naturally describe a new UI behavior on top of their websites, similar to a prior study setup [42], we asked participants to imagine themselves as the designer of the task UI, and also told them that a helper with domain knowledge (but no prior information about the task) will review their questions. We conducted follow-up interviews after each session. We compensated each participant with $20 USD for their time.

We asked participants to ask two questions per condition: one for web UI behavior, and one for SVG game UI behavior. We created the tasks with three goals in mind: they need to be commonly seen in practice, the UIs should look realistic, and they should be near the upper limit (in terms of complexity) of what CoCapture can handle. To achieve these goals, we used two common commercial websites (Reddit[11] and Stack Overflow[12]) and an open-source SVG

[9] www.google.com/gmail/
[10] docs.google.com/drawings/
[11] www.reddit.com
[12] www.stackoverflow.com

game[13]. We modified their UIs to replicate the popular UI issues we found in our needfinding study. We designed their complexity to result in highly dynamic transformation upon user input and the interplay between different DOM elements, both of which could help demonstrate the strength of CoCapture. Figure 5 shows two of the tasks: Reddit (left column), and an SVG game (right column). The screenshots with red borders (labeled a, c in Figure 5) are the existing UIs that we gave to the participants at the beginning of the study, and the ones with green borders (labeled b, d) are the desired output that we asked the participants to create and describe. Here we describe how these two tasks can show the ceiling of CoCapture along different dimensions:

- The Reddit task can show two dimensions of CoCapture's strength. First, a large part of the existing UI (e.g., header, trending section, and the content DOM elements) already have pre-built behaviors and constraints. With CoCapture, participants do not have to worry about what they are and how to reconstruct them. Second, one desired behavior is to remove the "Today's top growing" element (right-most one in Fig. 5 b) when the width of the browser is smaller than a certain threshold. CoCapture helps participants to save all the effort of dealing with the correlated behaviors on other non-target elements (e.g., siblings, parents) because the recording automatically preserves the underlying DOM structure within each snapshot, which static layout-capturing tools like WebToLayers cannot.
- The SVG game task shows two other dimensions of CoCapture's strength. First, the desired behavior requires adding a new element to the later part of the demonstration (the cyan shield shown at the bottom in Fig. 5 d). This will require participants to use the "Sketch" feature to add new elements. More generally, this requirement covers all the cases where new elements are required to be added or occur in the later part of a UI behavior, which is infeasible to replicate if using a static capturing tool (single UI state at one timestamp). Second, the desired behavior requires the player to plan out their demonstration, including record imaginary interaction with new elements that they would add later (e.g., colliding with the star as it moves left shown in the 2nd screenshot in Fig. 5 d)). CoCapture makes this process easier by making any visual elements easy to manipulate and modify after the recording is done, which existing screencast recording tools do not offer.

We designed these tasks using a rubric based on the common issues we found from our needfinding study (see supplementary material).

Similar to our needfinding study setup, we want to simulate a scenario where the participants have the desired UI changes in mind but have no direct access to them. To avoid biases, for each task we gave participants both the existing UI and the desired UI (i.e., ground truth), and used only visual annotation (e.g., cursor pointing) and determiners (e.g., "this part") to point out the differences between the existing and desired UIs, without offering specific descriptions. Participants could always access this information throughout the session and were not allowed to use any materials from the desired UI artifact (e.g., no screenshots).

6.1 Results and Analysis

We compared the accuracy of each question—the number of requirements satisfied by the description—and the time spent on asking each question between conditions. To measure the accuracy, we recruited an expert, a teaching staff member from a UI development course taught at the first author's university. We adopted the evaluation method from prior work that measures the correctness of touch behavior implementation [46]. We provided the expert with the task rubric, the existing UI, and the desired UI artifacts (see supplemental material). The rubric included items corresponding to the UI behavior requirements we gave to participants, and all items contributed equally to the accuracy percentages reported below. The expert evaluated the question for each item by examining whether the item description matched. We calculated (the number of matched items / total number of items) to work out the accuracy per question. We used a two-tailed Welch's t-test for our statistical analysis.

6.1.1 *Participants Created More Accurate Questions with CoCapture.*

Participants were able to create more accurate questions in the CoCapture condition ($p < .0001$) than in the control condition, resulting in average recall of 92.08% (σ = 12.17%, CoCapture) and 54.05% (σ = 18.46%, control) of the rubric items (Table 1) (Q1). The questions in the control condition included more words ($\mu = 115.71, \sigma = 68.83, p < .001$) than the CoCapture question did ($\mu = 38.23, \sigma = 18.16$), but we found no significant difference in the number of visual references ($p > .05$). These visual references included images and sketches in the control condition and animations in the treatment condition. This indicates that CoCapture helps participants spend less effort describing the dynamic UI behaviors through writing. We found no sufficient evidence that using CoCapture would force participants to add extra visual information, which can be overwhelming for those who review the questions.

6.1.2 *Participants Who Used CoCapture Spent Less Time Writing.*

One potential downside of using CoCapture is the time that it takes to create desired behaviors by capturing the base scene and adding the behaviors. Overall, we have insufficient evidence that using CoCapture will require more or less time to create a question (Q2) (Control (seconds): $\mu = 645.68, \sigma = 377.39$, Treatment: $\mu = 482.96, \sigma = 190.77, p > .1$). We further observed and annotated the video to break down the overall time, allowing us to understand how participants used CoCapture, especially the time that they spent writing the description and creating visual references. We found that, in terms of time, the trade-off between the two conditions existed in authoring animations and writing textual descriptions. While the participants in the control group spent most of their time writing textual descriptions, the opposite was true in the treatment condition: participants using CoCapture spent most of their time creating visual references via animated behavior (See Fig. 6). In the control condition, the participants' sketch activity included taking screenshots and sketching the desired output, but the limitations on what they could express led them to spend more time describing the behavior via text. These results suggest that

[13] github.com/starzonmyarmz/js13k-2018

	Control	CoCapture
description accuracy (%)***	54.05 (18.46)	92.08 (12.17)
# of words in description**	115.71 (68.83)	38.23 (18.16)
# of visual references (*ns*)	2.29 (1.01)	2.53 (0.52)
# of application switches***	15.10 (6.98)	5.03 (1.46)

Table 1: Measurements from 15 participants using CoCapture and control tools (Email + Google Drawings). Description accuracy (%) is calculated using (the number of satisfied items in a rubric / total number of items). Visual references include images, sketches, and animations. ** indicates $p < .001$, * indicates $p < .0001$. *ns* indicates not statistically significant. Their corresponding standard deviations are in parentheses.**

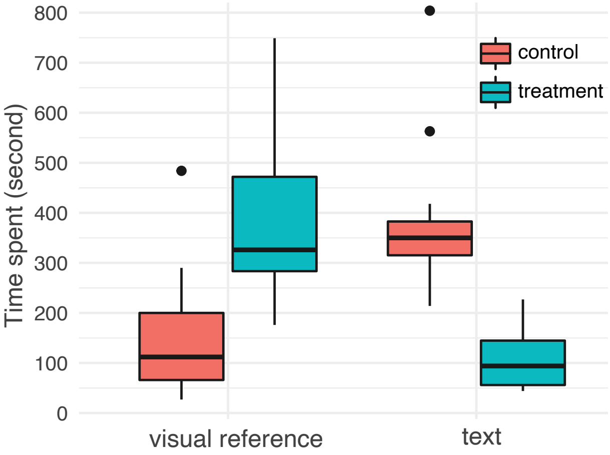


Figure 6: Time spent (s) on visual reference creation and text writing for the two conditions. The participants spent more time on creating animations and less time on writing the textual description in the treatment condition. For both cases, the differences were significant ($p < 0.0001$).

CoCapture encourages designers to actually prototype the behavior visually, which increased the description accuracy, as opposed to describing it in NL.

An additional benefit of using CoCapture was that it required less context switching (e.g., switching between applications), as participants could write down their questions and create visual references in a single application. The participants in the control group switched more frequently between different applications ($p < 0.0001$), as shown in Table 1. This indicates that CoCapture can reduce users' cognitive effort by requiring less context switching across different applications.

6.2 System Usability and Study Insights

To better understand CoCapture's other usability benefits or issues, we ran a thematic analysis on the interview transcripts with our own observations of participants' behavior patterns from the video.

6.2.1 CoCapture Is Needed, Useful, and Easy to Use. All the participants (15/15) recognized part or all of the scope, purpose, and value of CoCapture. For example, P15 summarized that:

"To redo this [study task] in like [Adobe] After Effects, which is probably what you would have to use, you basically have to rebuild every single part of this. At that point, you might as well just wing it and see if you can pull it off. Because it's gonna take such a long time to develop the animation, which will be ridiculous. So yes I think CoCapture totally makes sense."

Additionally, all the participants (15/15) gave positive feedback on CoCapture's UI, feeling it saved them effort on *"thinking about the description [or] writing the code"* (P10) and was easy to use, as *"it's very similar to video or audio editing"* (P1). Furthermore, all participants used the hypertext feature. They found it *"super important"* (P3, P9), thought it helped *"save effort/time"* (P2, P5–P10, P12–P15) in describing behaviors that were difficult to explain via text, and said that it helped them to be *"more concise"* (P4, P11) when creating their questions.

6.2.2 Describing Behavior Without Knowing the Exact Terms Was a Complex Task. In the control condition, we found that some participants were not able to clearly describe (underspecified) certain objects in their email. Five participants struggled to find suitable terms for different interface elements (e.g., the game character). As a result, they either took screenshots of the elements and annotated them or described the object using its properties (e.g., black box). Four participants in the control condition said that they spent a great deal of time writing the question as clearly as possible to ensure that the helper would understand their requests. One participant even searched for the name of a type of game action to make the question *"clear enough for others"* (P4). This suggests that communicating visual information using text can be both time-consuming and difficult. In the treatment condition, we instead observed that the participants were able to simply use demonstrative pronouns (e.g., this) with hypertext.

6.2.3 CoCapture Helped Decrease Users' Mental Processing Costs. Participants in the control condition had to frequently switch between creating visual references and writing text (Table 1). In contrast, 11 participants in the treatment condition created the entire animation first and then wrote the question in CoCapture. CoCapture helped participants *"expend less effort thinking about phrasing and terminology"* (P7). One participant noted that *"once I had the mockup made, I can kind of just refer back to that. So it was easier"* (P5). This suggests CoCapture shaped participants' workflow when creating questions, helping them organize the information and potentially better match the mental model through visual communication.

6.2.4 Capturing Dynamic Behaviors with Static Information Was Difficult. In the control condition, two-thirds of the participants (10/15) used Google Drawings to sketch on top of screenshots or create mockups of the desired interface from scratch. Three participants failed to adequately capture the scene of the game character staying in the air, as it required designers to operate multiple user inputs (i.e., capturing, jumping) at the same time. In contrast, CoCapture enabled participants to pause existing behaviors in the recording, making it easier to add consecutive or overlapping behaviors. The contrast between both groups' feedback paints a clear picture of the benefits brought by CoCapture's ability to manipulate elements of the existing behaviors.

6.2.5 Requests in the Control Condition Overspecified or Underspecified Question Details. When describing visual behaviors, it can be challenging to find the right specificity level. With the questions in the control condition, we observed participants' tendency to either overspecify or underspecify their questions. The larger standard deviation of the number of words (68.83) may be attributed to these varying levels of specification (see Table 1). Additionally, we observed that overly precise or unnecessary details made the resulting description tedious and potentially more difficult for others to read and comprehend, as shown in the study below. *"My question [in my email] was longer because I kind of describe all the behavior. With CoCapture I kind of just say, hey, I want this [referring to an object] to resize like this [referring to an animation]. Instead of saying like, something much more specific"* (P5). In turn, underspecifying made participants' requests unclear, which was confirmed from the follow-up study (Study 2). This reflects the inherent challenges that exist in current methods of visual communication, as they only support text and static visual information creation, making dynamic visual changes difficult to specify. Our results suggest that CoCapture can help specify dynamic information by enabling animation creation and remixing.

7 STUDY 2 - UNDERSTANDING UI BEHAVIOR QUESTIONS

To evaluate CoCapture's effectiveness in helping people understand UI behavior questions, we recruited six participants from two pools—the same participant pool as in Study 1, and Upwork[14]—to review the questions created in Study 1. There was no overlap in participants between the two studies, and all participants had at least two years of UI development experience. Each participant acted as a helper and was assigned to evaluate four questions created by one participant (only one to reduce learning effect) in Study 1. Participants used the rubric (same as in Study 1) to rate each question from 1 (not clear at all) to 5 (completely clear). Before reviewing the two CoCapture questions, participants watched a 5-minute tutorial on the use of CoCapture, and they were also asked to try it to review an example question. After the evaluations, we interviewed each participant. We conducted Study 2 remotely using TeamViewer[15]. Each session lasted 30 minutes, and we paid each participant $15 USD.

7.1 CoCapture Showed the Potential of Facilitating Effective Comprehension of UI Questions

While we did not conduct any statistical analysis due to the small number of participants, the average time spent and clarity assessment showed promise that CoCapture can be effective in helping users comprehend questions. For the CoCapture questions, participants spent 247.25s ($\sigma = 152.03$) evaluating them and gave an average rating of 4.21 out of 5 ($\sigma = 1.32$) for clarity. In contrast, participants rating questions from the control condition spent 311.83s ($\sigma = 176.12$) and gave an average rating of 1.40 out of 5 ($\sigma = 1.54$).

7.2 CoCapture Questions Were Clearly Presented

In the post-hoc survey and interview, all six participants (S1–S6) found that the CoCapture questions were more clearly presented (Q3). One participant noted that *"it [CoCapture] is like combining what the email does the best with what video does best"* (S1), while another said *"it's just like watching a video, and I don't have to guess what the text is asking"* (S5). In contrast, participants thought other people might fail to accurately interpret some UI behaviors written in the control condition due to their lack of expressiveness or implicit assumptions. *"I know it's common sense that the object will disappear when the character hits it, but not everyone knows that, so I can't understand it when reading the question"* (S5).

We also observed that all participants constantly scrolled up and down the questions in the control condition (Gmail). S2 talked about the reason: *"I was scrolling up and down because I was like, where did I see this information? It's not said very clear. I had to read the email a bunch of times to start building up my understanding of what it was asking for."* For the CoCapture interfaces, by contrast, participants found that the hypertext feature helped them navigate the visual references more easily, lowering their cognitive effort. The preview and motion trail feature for each mockup highlighted the associated changes, guiding participants' attention to the relevant information. *"I can actually see the visual like trail of the box moving associated exactly with the NL that it's written in"* (S6).

8 DISCUSSION

In summary, we found that CoCapture helped participants in Study 1 ("requester" from now on) effectively communicate about UI behavior issues by easing the creation of desired behavior mockups on existing websites and by clarifying NL descriptions with hypertext. This in turn made those behavior issues clear and explicit, and it made the descriptions easier to navigate for participants in Study 2 ("helper" from now on). Consistent with prior theory [29], this finding indicates that clear visual context helps people ground communication. However, we observed that the dynamic nature of UI behaviors makes it more challenging for requesters to create descriptions, even with basic visual information (e.g., screenshots, sketches) as seen in [43].

[14] www.upwork.com.

[15] https://www.teamviewer.com/.

8.1 Element-Based Animations as First-Class Objects

As we have shown, requesters spent much more time writing text in the control condition than in the treatment condition, and the length of their written content was much greater. This is because common editor tools, such as Gmail in this study, only support linear sets of static content (e.g., text and images), which is limiting when trying to convey dynamic and interactive behaviors. One requester shared some of his working experience: *"Half the time I work with people where English is their second language. Being able to do it like this [CoCapture] would save a lot of time because the amount of time spent in trying to explain something to make sure that everybody's on the same page is very time-consuming"* (P15). Communicating dynamic visual information via text creates a burden for requesters, who must describe their needs by dismantling the question into text and static images, which can result in an underspecified description; it also burdens helpers, who must connect this disparate information in an attempt to imagine what visual behaviors the requester wants to create.

CoCapture removes these burdens by enabling requesters to communicate their visual context as a whole without breaking it down further. Most helpers found that the UI behavior descriptions in CoCapture were more explicit, contextualized, and clearly presented. This is because CoCapture helped designers form a mockup of their requested visual behaviors to be delivered alongside their textual description. It maintained the dynamic information as a continuous and holistic view. Similar to the suggestions of prior work that code should be treated as a first-class object rather than a block of plain text [63], we argue that for UI behavior mockups, element-based animations should be treated as first-class objects rather than a combination of different pieces of information such as screenshots or plain text descriptions. Additionally, these animations can guide designers to write more concise and understandable questions than they could using the tools in the control condition.

8.2 Hypertext Helps Organize and Ground Communication

Visual context and NL descriptions should co-exist during communication. With the desired UI behaviors created in CoCapture remixed holistically with the pre-built UI behaviors, the hypertext feature made it easier for requesters to explain their needs compared to the linear presentation in the control condition. Similar to prior work [59], the hypertext also served as visual cues to guide helpers in the second study to more easily parse and navigate each question. In the first study, we noticed that this feature freed requesters from following the linear order of a question, as they must with text-only descriptions. For questions that included multiple animations, the hypertext feature enabled helpers to prioritize those that were more important or complex, and it also reduced the burden of effort by helping them plan their response. To understand the questions, helpers constantly interacted with the remixed recording, the referenced artifacts, and the hypertext feature in CoCapture; when reading the emails, the interaction was limited to scrolling up and down. Together, our findings show that CoCapture shifted both requesters' and helpers' main attention from textual to visual information—that is, the dynamic UI behaviors. It also changed their main workflow from linear to cross-referenced between text and UI behaviors, which we have shown to be appropriate (effective) for communicating UI changes that involve the dynamic transformation of multiple interface elements on existing websites.

8.3 Other Use Cases

Our evaluation study focused on the common scenario where a designer wants to modify an animation as it is currently implemented in a web UI. More broadly, CoCapture can be used in other cases or implemented in other prototyping and development tools. For example, CoCapture allows users to copy existing behaviors from the recording and apply them to other elements (existing ones or sketches). This is done by using the "Clone" and "Change element" features (Fig. 3). Also, it allows designers to explore the relations between new designs and existing constraints, helping them to iterate on their own ideas before asking for feedback or development support. In the user study, we also observed that participants use CoCapture to iterate on the designs and figure out the optimal properties (e.g., location, responsive speed) when creating the desired behaviors (there are different ways to create).

8.4 CoCapture's Role in the Design Lifecycle

While we focused on demonstrating CoCapture's strength in supporting re-designing implemented websites, it can also be used at the initial stage of the design because it is common practice to use and refer to existing behaviors from existing websites when designers explore different ideas. If a designer wanted to demonstrate the desired behaviors by borrowing and revising the behaviors from an existing website, CoCapture can still be useful at different stages in the design cycle. This can support designer-developer communication, using revised or existing behaviors from an existing website as a reference. Alternatively, CoCapture can be used at the ideation stage, which can be useful for exploring different ideas as a group.

8.5 System Scope and Limitations

CoCapture is most helpful for communicating UI behaviors that involve the dynamic transformation of multiple interface elements on existing websites. It is less necessary for simple element property changes (e.g., change the background color to red when clicking a button), or UIs that contain few elements or behaviors. In such cases, it is easy to reconstruct the artifacts from scratch using tools like Figma or Adobe XD. Additionally, CoCapture currently only supports web UI behaviors, but its interactive design and the study findings could be applied to or used for reference in other UI environments (e.g., mobile apps). Finally, while we chose the baseline tools (i.e., Google Drawings and Gmail) for their simplicity and low learning curve, they might not be the most commonly used design tools.

8.6 Future Work

Participants in both studies provided suggestions on improving CoCapture's efficacy and usability. Two requesters felt that CoCapture was similar to a video or audio editing tool, though it lacked some common features of those tools, like the ability to automatically save all changes or undo an action with a hotkey. Three requesters wished they could link the same text to multiple mockups or elements. This makes sense because designers often apply the same effect to different elements (e.g., an HTML class), or multiple behaviors to the same element (e.g., the box is moving while shrinking). Additionally, multiple requesters wished CoCapture indicated user inputs (e.g., click) during the demonstration on the base scene replay. Along with this, they suggested adding an alignment feature, like snapping to a certain timestamp when dragging the slider, to help them synchronize animations and the base scene by time, attributes (e.g., x position), or ratios (i.e., change/time). One way to realize this is to modularize the transition behavior like Expresso [30] does. Three helpers suggested that CoCapture highlight each hypertext as its corresponding element or animation is replayed. We will accomplish this by enlarging the hypertext when the recording plays at the start time of the associated mockup. CoCapture also preserves the original web DOM tree structure, and the style (e.g., layout), for each snapshot. Similar to Telescope [26], future work can use this benefit to further link UI elements and behaviors to related code snippets so that helpers can understand the code context cohesively and provide assistance directly.

9 CONCLUSION

In this paper, we proposed an effective approach for UI designers to communicate about changes or issues of interactive UI behaviors on existing UIs. We designed and implemented CoCapture, which instantiates this approach to enable users to 1) easily (through direct manipulation) create animated mockups on top of arbitrarily complex web interfaces by demonstrating and remixing, and 2) effectively (via hypertext) communicate about these mockups with easy referencing techniques. Compared to existing approaches, we showed that CoCapture can help designers to create more accurate and more organized questions with rich context. In sum, CoCapture opens up opportunities for more effective UI iteration by providing a natural and easy way of communicating interface behaviors.

10 ACKNOWLEDGEMENTS

We thank Walter S. Lasecki for his feedback on this work at the early stage, our anonymous reviewers for their helpful suggestions on this work, and our study participants for their time. This work is supported by NSF Award 1915515.

REFERENCES

[1] 2020. Adobe XD. https://www.adobe.com/products/xd.html Accessed: April, 2020.

[2] 2020. Figma. https://www.figma.com/ Accessed: March, 2020.

[3] 2020. Google Inc. https://github.com/GoogleChromeLabs/ProjectVisBug Accessed: March, 2020.

[4] 2020. MutationObserver. https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver Accessed: March, 2020.

[5] 2020. Scrimba. https://www.scrimba.com/ Accessed: March, 2020.

[6] Brian Burg, Richard Bailey, Amy J Ko, and Michael D Ernst. 2013. Interactive record/replay for web application debugging. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. 473–484.

[7] Brian Burg, Amy J Ko, and Michael D Ernst. 2015. Explaining visual changes in web interfaces. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 259–268.

[8] N. Burtnyk and M. Wein. 1971. Computer-Generated Key-Frame Animation. *Journal of the SMPTE* 80 (1971), 149–153.

[9] Kerry Shih-Ping Chang and Brad A Myers. 2012. WebCrystal: understanding and reusing examples in web authoring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3205–3214.

[10] Kathy Charmaz. 2006. *Constructing grounded theory: A practical guide through qualitative analysis*. sage.

[11] Sarah E Chasins, Maria Mueller, and Rastislav Bodik. 2018. Rousillon: Scraping Distributed Hierarchical Web Data. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 963–975.

[12] Yan Chen, Sang Won Lee, Yin Xie, YiWei Yang, Walter S Lasecki, and Steve Oney. 2017. Codeon: On-demand software development assistance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 6220–6231.

[13] Yan Chen, Maulishree Pandey, Jean Y Song, Walter S Lasecki, and Steve Oney. 2020. Improving Crowd-Supported GUI Testing with Structural Guidance. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[14] Pei-Yu Chi, Sen-Po Hu, and Yang Li. 2018. Doppio: Tracking ui flows and code changes for app development. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[15] Parmit K Chilana, Nathaniel Hudson, Srinjita Bhaduri, Prashant Shashikumar, and Shaun Kane. 2018. Supporting Remote Real-Time Expert Help: Opportunities and Challenges for Novice 3D Modelers. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 157–166.

[16] Herbert H Clark and Susan E Brennan. 1991. Grounding in communication. (1991).

[17] Cecília Kremer Vieira da Cunha and Clarisse Sieckenius de Souza. 2003. Toward a culture of end-user programming understanding communication about extending applications.

[18] Paul Dourish and Victoria Bellotti. 1992. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*. 107–114.

[19] Susan R Fussell, Leslie D Setlock, Jie Yang, Jiazhi Ou, Elizabeth Mauer, and Adam DI Kramer. 2004. Gestures over video streams to support remote collaboration on physical tasks. *Human-Computer Interaction* 19, 3 (2004), 273–309.

[20] Lars Grammel, Melanie Tory, and Margaret-Anne Storey. 2010. How information visualization novices construct visualizations. *IEEE transactions on visualization and computer graphics* 16, 6 (2010), 943–952.

[21] Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2010. Chronicle: capture, exploration, and playback of document workflow histories. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. 143–152.

[22] Carl Gutwin and Saul Greenberg. 2002. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work (CSCW)* 11, 3-4 (2002), 411–446.

[23] Carl Gutwin, Mark Roseman, and Saul Greenberg. 1996. A usability study of awareness widgets in a shared workspace groupware system. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*. 258–267.

[24] Carl Gutwin, Gwen Stark, and Saul Greenberg. 1995. Support for workspace awareness in educational groupware.. In *CSCL*, Vol. 95. 147–156.

[25] Joshua Hibschman and Haoqi Zhang. 2015. Unravel: Rapid web application reverse engineering via interaction recording, source tracing, and library detection. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 270–279.

[26] Joshua Hibschman and Haoqi Zhang. 2016. Telescope: Fine-tuned discovery of interactive web UI feature implementation. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 233–245.

[27] David Kirk and Danae Stanton Fraser. 2006. Comparing remote gesture technologies for supporting collaborative physical tasks. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 1191–1200.

[28] David S Kirk and Danaë Stanton Fraser. 2005. The effects of remote gesturing on distance instruction. (2005).

[29] Robert E Kraut, Susan R Fussell, and Jane Siegel. 2003. Visual information as a conversational resource in collaborative physical tasks. *Human–computer interaction* 18, 1-2 (2003), 13–49.

[30] Rebecca Krosnick, Sang Won Lee, Walter S Lasecki, and Steve Onev. 2018. Expresso: Building responsive interfaces with keyframes. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 39–47.

[31] James A Landay and Brad A Myers. 1995. Interactive sketching for the early stages of user interface design. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 43–50.

[32] Walter S Lasecki, Juho Kim, Nick Rafter, Onkur Sen, Jeffrey P Bigham, and Michael S Bernstein. 2015. 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, 1925–1934.

[33] Sang Won Lee and Jason Freeman. 2013. Real-time music notation in mixed laptop–acoustic ensembles. *Computer Music Journal* 37, 4 (2013), 24–36.

[34] Sang Won Lee, Yujin Zhang, Isabelle Wong, Yiwei Yang, Stephanie D. O'Keefe, and Walter S. Lasecki. 2017. 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* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 817–828. https://doi.org/10.1145/3126594.3126595

[35] Germán Leiva and Michel Beaudouin-Lafon. 2018. Montage: A Video Prototyping System to Reduce Re-Shooting and Increase Re-Usability. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 675–682.

[36] Germán Leiva, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2020. Pronto: Rapid Augmented Reality Video Prototyping Using Sketches and Enaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

[37] Toby Jia-Jun Li, Amos Azaria, and Brad A Myers. 2017. SUGILITE: creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI conference on human factors in computing systems*. 6038–6049.

[38] Yang Li, Xiang Cao, Katherine Everitt, Morgan Dixon, and James A Landay. 2010. FrameWire: a tool for automatically extracting interaction logic from paper prototyping tests. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 503–512.

[39] James Lin, Mark W Newman, Jason I Hong, and James A Landay. 2000. DENIM: finding a tighter fit between tools and practice for Web site design. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 510–517.

[40] Michael McCurdy, Christopher Connors, Guy Pyrzak, Bob Kanefsky, and Alonso Vera. 2006. Breaking the fidelity barrier: an examination of our current characterization of prototypes and an example of a mixed-fidelity success. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 1233–1242.

[41] David W McDonald, Chunhua Weng, and John H Gennari. 2004. The multiple views of inter-organizational authoring. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*. 564–573.

[42] Ronald Metoyer, Bongshin Lee, Nathalie Henry Riche, and Mary Czerwinski. 2012. Understanding the verbal language and structure of end-user descriptions of data visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1659–1662.

[43] Brad Myers, Sun Young Park, Yoko Nakano, Greg Mueller, and Amy Ko. 2008. How designers design and program interactive behaviors. In *2008 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, 177–184.

[44] Jasper O'Leary, Holger Winnemöller, Wilmot Li, Mira Dontcheva, and Morgan Dixon. 2018. Charrette: Supporting In-Person Discussions around Iterations in User Interface Design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–11.

[45] Steve Oney, Christopher Brooks, and Paul Resnick. 2018. Creating guided code explanations with chat. codes. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–20.

[46] Steve Oney, Rebecca Krosnick, Joel Brandt, and Brad Myers. 2019. Implementing Multi-Touch Gestures with Touch Groups and Cross Events. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[47] Stephen Oney and Brad Myers. 2009. FireCrystal: Understanding interactive behaviors in dynamic web pages. In *2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 105–108.

[48] Stephen Oney, Brad Myers, and John Zimmerman. 2009. Visions for Euclase: Ideas for Supporting Creativity through Better Prototyping of Behaviors. In *ACM CHI 2009 Workshop on Computational Creativity Support*.

[49] John F Pane, Brad A Myers, et al. 2001. Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies* 54, 2 (2001), 237–264.

[50] Jungkook Park, Yeong Hoon Park, and Alice Oh. 2018. Non-Linear Editing of Text-Based Screencasts. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 403–410.

[51] Sun Young Park, Brad Myers, and Amy J Ko. 2008. Designers' natural descriptions of interactive behaviors. In *2008 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, 185–188.

[52] Amy Pavel, Dan B Goldman, Björn Hartmann, and Maneesh Agrawala. 2016. VidCrit: video-based asynchronous video review. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 517–528.

[53] Zhengrui Qin, Yutao Tang, Ed Novak, and Qun Li. 2016. Mobiplay: A remote execution based record-and-replay tool for mobile applications. In *Proceedings of the 38th International Conference on Software Engineering*. 571–582.

[54] William T. Reeves. 1981. Inbetweening for Computer Animation Utilizing Moving Point Constraints. In *Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques* (Dallas, Texas, USA) *(SIGGRAPH '81)*. Association for Computing Machinery, New York, NY, USA, 263–269.

[55] Hugo Romat, Emmanuel Pietriga, Nathalie Henry-Riche, Ken Hinckley, and Caroline Appert. 2019. SpaceInk: Making Space for In-Context Annotations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 871–882.

[56] Amanda Swearngin, Mira Dontcheva, Wilmot Li, Joel Brandt, Morgan Dixon, and Amy J Ko. 2018. Rewire: Interface design assistance from examples. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[57] Kesler Tanner, Naomi Johnson, and James A Landay. 2019. Poirot: A Web Inspector for Designers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[58] Edward R Tufte. 1983. *The visual display of quantitative information*. Vol. 2.

[59] April Yi Wang, Zihan Wu, Christopher Brooks, and Steve Oney. 2020. Callisto: Capturing the" Why" by Connecting Conversations with Computational Narratives. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[60] Patricia G Wojahn, Christine M Neuwirth, and Barbara Bullock. 1998. Effects of interfaces for annotation on communication in a collaborative task. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 456–463.

[61] Dongwook Yoon, Nicholas Chen, François Guimbretière, and Abigail Sellen. 2014. RichReview: blending ink, speech, and gesture to support collaborative document review. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 481–490.

[62] Dongwook Yoon, Nicholas Chen, Bernie Randles, Amy Cheatle, Corinna E Löckenhoff, Steven J Jackson, Abigail Sellen, and François Guimbretière. 2016. RichReview++ Deployment of a Collaborative Multi-modal Annotation System for Instructor Feedback and Peer Discussion. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. 195–205.

[63] Joyce Zhu, Jeremy Warner, Mitchell Gordon, Jeffery White, Renan Zanelatto, and Philip J Guo. 2015. 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, 101–109.