ReferencesPlus: Providing In-Situ UI Discussions with Dynamic Visual Referencing

NAVEEN SIVASANKAR, University of British Columbia, Canada

In pull request (PR) discussions, discussants typically use references as part of communication, and sometimes make references to visual elements. Visual referencing is especially useful when discussing user interfaces, but existing pull request interfaces provide limited support for it in discussions. This paper addresses this deficiency by developing *ReferencesPlus*, a Chrome extension that captures dynamic UI changes and embeds them into pull request discussions to support in-situ UI discussion with dynamic visual referencing. During evaluation, *ReferencesPlus* was compared against the existing GitHub PR interface in a study focusing on the scenarios of discussion comprehension and comment creation. This evaluation showed that *ReferencesPlus* has a lower workload than the existing PR interface and may also enable more precise visual references.

CCS Concepts: • **Human-centered computing** → *Interaction design process and methods*; **Graphical user interfaces**; • **Software and its engineering** → **Collaboration in software development**.

Additional Key Words and Phrases: Pull Request, User Interface, Code Review, Software Development, Reference, Visual Referencing

ACM Reference Format:
Naveen Sivasankar. 2022. ReferencesPlus: Providing In-Situ UI Discussions with Dynamic Visual Referencing. 1, 1 (April 2022), 16 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Code review is an important part of software development, and has been a valuable tool in software projects for many years [1]. Pull requests (PRs) are a more modern approach to code review and this method has gained significant traction since its introduction. This support for pull requests comes due to increased usage of code hosting sites, and because of the capabilities of distributed version control systems on which code hosting websites rely [17]. GitHub is a popular cloud-based code hosting site with great support for open source software development due to the ability to allow developers to clone any public repository, and is the focus of this paper [17, 26]. GitHub relies on Git for version control and developers who work with hosted repositories can take full advantage of Git's features and its integration with the site.

The pull-request lies at its heart of the pull-based software development model. A pull request can be summarised as a proposed change to a project that is submitted by a developer, then reviewed and accepted or rejected by the repository's collaborators [4]. Developers who seek to make changes to an existing repository typically "fork" or locally clone the repository, after which they can make changes without affecting the original project. To contribute to the original project, the developer must make a pull request to the repository's owner(s). This often involves comparing the developer's local changes to the main branch of the original project. GitHub's interface can display these differences

Author's address: Naveen Sivasankar, University of British Columbia, Canada, naveenrs@student.ubc.ca.

and allows other developers to review the changes and suggest improvements via comments. Those with sufficient permissions can then merge the pull request if there are no conflicting changes and if the concerns of reviewers are addressed. Usage of this process facilitates community discussion and allows rapid incorporation of changes into the repository [17].

Discussions are an essential aspect of any pull request [17]. To facilitate communication, PR stakeholders often include references to various elements, such as platform-specific entities, source code, and URLs [7]. GitHub also provides support for referencing visual elements via textual and visual means, which is extremely useful for discussing user-facing changes and developing new interfaces. Prior research has found that UI elements and properties of those elements are often referenced in projects with interface elements, especially in mobile and web development projects. Unfortunately, a lack of native support causes developers to exert more effort when expressing references to relevant elements and makes decoding references more challenging for readers [7].

This paper focuses on GitHub due to its popularity and aforementioned support for open source development. GitHub has a rich pull-request interface with native support for extension using GitHub Apps and OAuth Apps, which leverage internal tools and APIs to improve workflows and add additional flexibility [12]. Discussions are enhanced through a special variation of markdown that allows developers to use custom functionality and GitHub specific features when making comments [13]. In addition, GitHub allows commenters to reference visual media by uploading files and embedding them in comments via auto-generated anonymized URLs [14]. While images and video can convey useful information about user-facing changes, there are significant limitations to this approach.

Theories about grounding in communication discuss the importance of references in establishing a shared understanding between communicators [9, 10]. Visual referencing is a key communication mechanism for establishing grounding in PR discussions, especially when discussing user interfaces. Visual references provide essential context by linking code changes to their user-facing impacts and allowing specific elements to be identified and discussed, which can aid in communication. Unfortunately, support for this mode of referencing in GitHub PR discussions is limited to textual comments, images, and short videos. Although some third-party tools exist to address this shortfall[1], they do not support precise referencing and commenting on specific UI elements and opt for a general purpose solution instead. In summary, a lack of support for visual referencing means that people in PR discussions are constrained by the existing interface and must deal with its disadvantages.

The primary issue with the current pull request interface is a lack of support for visual referencing behaviours, which makes establishing common ground regarding dynamic changes and visual elements more difficult. Chopra et al. identified several challenges with current methods of referencing user interface elements. Using text to communicate visual changes requires verbose descriptions, utilising images and video instead of text still requires additional time and effort, and current methods to include visual references disrupt workflow because of excessive context switching [7]. In addition, file size limits place restrictions on what can be embedded in discussions [14]. These concerns indicate the need for an embedded interface that brings better visual referencing into the existing static context, allowing a more complete picture of the impacts of code changes.

To address these issues, a Chrome extension that captures dynamic changes and visual elements and embeds them directly into a static context is developed, with a specific focus on web development. The tool is designed to capture the dynamic context of web applications and embed it within pull request discussions. This paper proposes that the tool will reduce the workload and confusion experienced by reviewers by making clear links between text and UI elements,

[1]https://www.loom.com/

and by avoiding context switching where possible. The structure of this paper is as follows. First, the design process and implementation details of the tool are discussed. Next, the details an evaluative user-study comparing the tool to an existing interface in the context of pull request discussions are reported. The results gathered from this study are then summarised and used to determine the usability and efficacy of this tool. Finally, potential improvements to the tool and future directions for research are discussed.

2 RELATED WORK

2.1 Code Review and Pull Requests

Much research exists on source code review and specifically on modern code review, a lightweight, informal, and tool-based approach [1]. Some research focuses on the approaches of specific companies and how they inform key ideas, processes, and developer experiences. For example, research investigating Google and Microsoft found that code review is essential for not only finding defects, but also for knowledge sharing, code maintainability, and encouraging transparency among developers [27, 29]. A systematic mapping study by Badampudi et al. [2] discovered a variety of topics in the literature, with improvements to the code review process accounting for approximately 20% of surveyed papers. Particularly relevant are papers focused on using static analysis tools to reduce code review effort and support collaborative modern code review [2]. Papers on these topics indicate that the usage of such tools can provide a time benefit, improve code quality, and support richer, more productive discussions due to their ability to identify common issues and code smells, allowing reviewers to focus on more demanding tasks [3, 18, 20, 25, 28].

In pull-based development projects, code review occurs primarily through pull requests, where code changes are presented to others for review. This process can be time-consuming and may require reviewers to adopt specific processes or verify certain requirements [15]. These two aspects of pull requests mean that concepts such as automatic tools and runtime behaviour are not a priority of this approach. Some limitations have been addressed through the use of bots, which can be integrated into GitHub and provide additional tooling such as automatic static analysis [19]. While there is much research on static analysis tools in code review, little attention is given to other tools and approaches, which may be necessary for understanding behaviours that are not visible from the code alone. This gap is addressed through the tool's ability to capture aspects other than static artifacts, such as the visual and runtime behaviour of the code, and integrate them into the code review context.

2.2 Communication Through Multimodal Referencing

Communication often involves references, which are relations between two objects where one object connects to the other [7]. Effective use of references when communicating requires grounding, which is when knowledge and ideas are shared among discussants [10]. Without achieving common ground, discussants may not be able to understand and identify the focus of a reference due to insufficient shared knowledge and beliefs [9]. Correct identification of referenced objects and de-referencing is essential to successful communication, and usage of references can change the communication behaviour of those involved [24]. This is especially relevant in multimodal interfaces, where information is conveyed through two or more modalities that are cohesively integrated together. Cohen's [11] definition of modality, which concentrates on the "syntactic, semantic and pragmatic properties" of the underlying data, is appropriate in this discussion of multimodal referencing.

Existing research on multimodal referencing has produced tools that link various modalities together in different collaborative platforms [5, 6, 8, 21, 30] Chang et al. [5] developed a system to support remote collaboration called

AlphaRead, which allows physical objects to be referenced via readable annotations in a dynamic video interface. An investigation of video comments revealed referencing patterns involving timed intervals, visual objects, and speech, for which proof-of-concept interfaces supporting those patterns encouraged intuitive ways of video referencing [30]. Chua et al. [8] developed an interface capable of referencing visual materials in online discussions, allowing discussion text to be linked to multiple relevant contexts. A tool with more relation to this paper is *Winder* [21], which simplifies user interface prototyping by linking design objects and particular elements of the mockup to parts of transcripts that represent recorded discussion. Finally, *CoCapture* [6] is a tool that partially addresses challenges with referencing visual elements in web development by capturing existing website interactions and allowing users of the tool to link textual descriptions to user-facing elements. This research acknowledges these tools and takes inspiration from *CoCapture* by developing a tool that similarly captures website behaviour but integrates into pull request discussions to reduce context switching.

3 DESIGN

3.1 Requirement and Constraints

As previously stated, the main contribution of this paper is a tool that embeds the dynamic context of web applications within pull request discussions to allow in-situ UI discussion with visual referencing. Building on Chopra et al.'s [7] discussion of the existing limitations of the interface leads to these requirements for the tool.

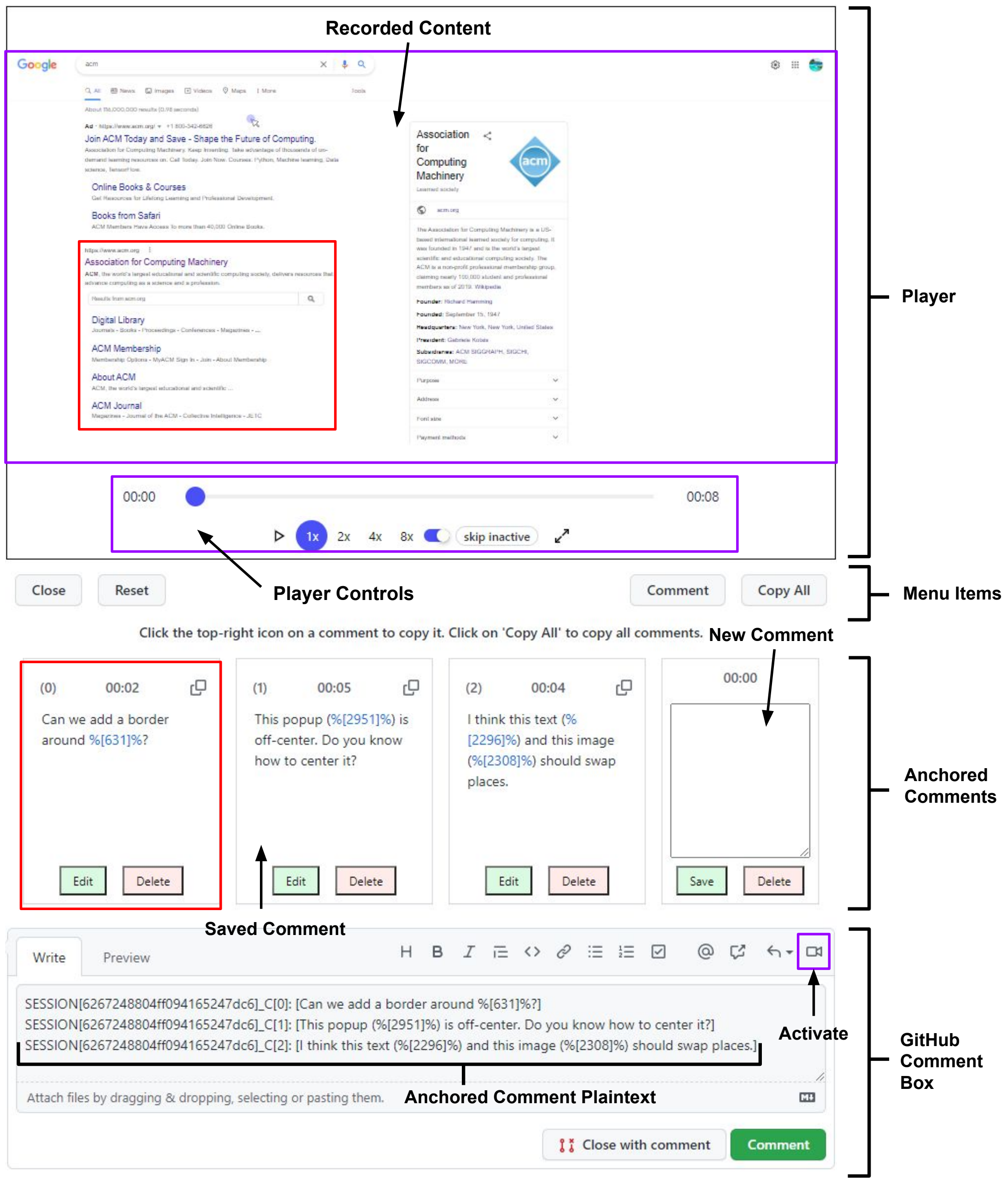
R1: Support visual referencing of both static and dynamic changes.
R2: Capture the visual context around a change or referenced element.
R3: Reduce context switching and keep users in the pull request discussion when possible.

R1 is the most important requirement as it describes the primary purpose of the tool. Both static and dynamic changes are specified given that user interfaces have dynamism and interactivity that cannot be captured via static media. The addition of **R2** ensures that others can understand the spatial and temporal context related to the reference, which clarifies how the referenced change/element relates to the rest of the interface. Finally, **R3** follows directly from the results of prior research and emphasises that the tool must live within the pull request.

The design choices of the tool are somewhat constrained by **R3** as it encourages a specific implementation. The only way to embed the tool into pull request discussion in the GitHub web application is to implement it as a browser extension, since the APIs required to modify existing websites without accessing their code are only accessible to browser extensions. Chromium-based browsers are targeted due to their desktop market share and extension support. Additionally, web applications are the main focus because further tooling is needed to capture and integrate software without a web interface into an existing website, whereas integrating web applications leverages existing browser support and web technologies. With these requirements and constraints in mind, three high-level designs for the tool are explored and considered: annotated video, emulation, and record-replay. These approaches are considered because of their ability to capture the necessary dynamic context of web applications and because they provide options to link text to visual elements.

3.2 Final Interface Design

ReferencesPlus is built using the record-replay approach, and embeds into an existing pull request discussion. Record-Replay involves capturing interactions and events at runtime, then replaying those captured interactions in a different context. This paper focuses on a form of record-replay utilised by *Mugshot* [22], a system that records executions of web



Click the top-right icon on a comment to copy it. Click on 'Copy All' to copy all comments.

(0) 00:02
Can we add a border around %[631]%?

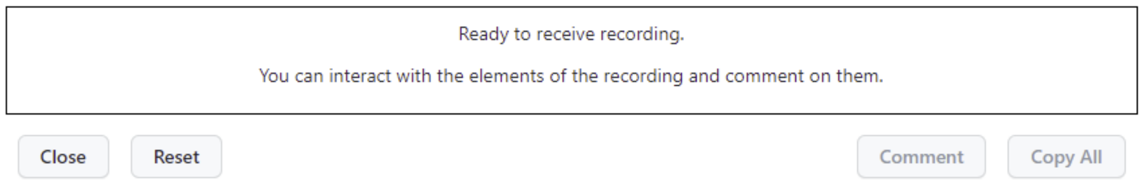
(1) 00:05
This popup (%[2951]%) is off-center. Do you know how to center it?

(2) 00:04
I think this text (%[2296]%) and this image (%[2308]%) should swap places.

SESSION[6267248804ff094165247dc6]_C[0]: [Can we add a border around %[631]%?]
SESSION[6267248804ff094165247dc6]_C[1]: [This popup (%[2951]%) is off-center. Do you know how to center it?]
SESSION[6267248804ff094165247dc6]_C[2]: [I think this text (%[2296]%) and this image (%[2308]%) should swap places.]

Fig. 1. An overview of the *ReferencesPlus* interface. Recorded web interactions appear in the player, and can be controlled via the Player controls. The UI elements in the player can be focused by hovering over them. Comments can be made on the recorded interactions by clicking the "Comment" button, which adds a new anchored comment. Comments on recordings can be shared in GitHub by copying them into the GitHub comment box.

applications and supports deterministic replay of those executions. An annotated screen recording, as well as making references on a live version of the application (emulation), were also considered. Although screen recording should be a familiar experience for many users, this workflow is cumbersome because users have to plan their interactions to make the correct annotations, and the file sizes can grow very large. Emulation requires less effort from users since they can make references as the interact with the application, but is difficult to implement because the emulator must support various tech stacks with robust build tools. Without native support for flexible build configuration, this approach is not feasible. Record-replay has some of the benefits of annotated screen recording and emulation, but doesn't require complex build configuration or excessive planning to capture visual references. It can be made interactive and can be packaged in an interface similar to a video recorder and video player. As a result, it strikes a balance between the two other approaches while providing the required support for referencing, and it is the chosen approach. Now, the ways in which the design of *ReferencesPlus* supports referencing dynamic UI changes in pull request discussions and the rationale behind the various design choices is discussed.


Ready to receive recording.

You can interact with the elements of the recording and comment on them.

Fig. 2. The placeholder and disabled menu items that appear when no recording is present. This placeholder contains text explaining that a recording can be received and what to do with a recording once it is captured. The "Comment" and "Copy All" buttons are disabled until web interactions are captured.

3.2.1 Capturing web interactions. To support capturing web interactions as part of record-replay, *ReferencesPlus* provides a clear entry point for a recording. The interface appears above the comment box at the bottom of a pull request, and is toggled via the "Activate" button (Figure 1). The button utilises GitHub's design language for seamless integration, making it fit in with the rest of the website. Only the "Close" and "Reset" menu items are active while no recording is present, and a placeholder for the player will show up until it is replaced with captured content (Figure 2). Users can then navigate to any website and click *Record Page* in the right-click menu (Figure 3). While recording, users can interact with the page with the caveat that they cannot navigate to a different URL. To stop recording, users must click *"Stop Recording"* in the right-click menu, which stops capturing interactions and sends the captured log to the pull request (Figure 3). These actions are located in the right-click menu because this follows patterns suggested by Google for Chrome extension development [16], and because the right-click menu is accessible in all websites.

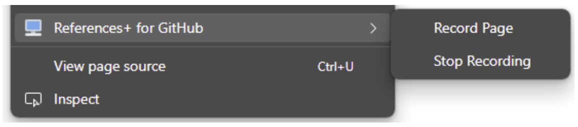
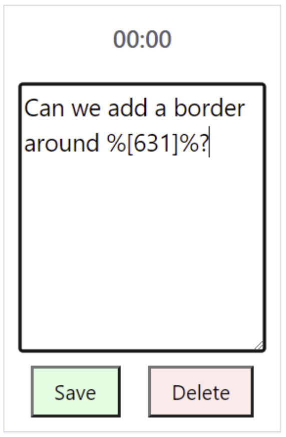
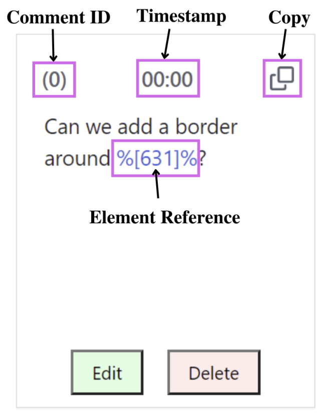


Fig. 3. The right-click context menu is accessible on all websites and allows users to record interactions with the "Record Page" item, and stop recording with the "Stop Recording" item.

3.2.2 Interacting with and referencing visual elements. Recordings appear in the GitHub pull request and replace the placeholder with a video player with various playback controls, which are implemented by an existing library (Figure 1). Once the recording is embedded, no further context switching is required to work with it. Users can play and pause the



(a) Editable comment



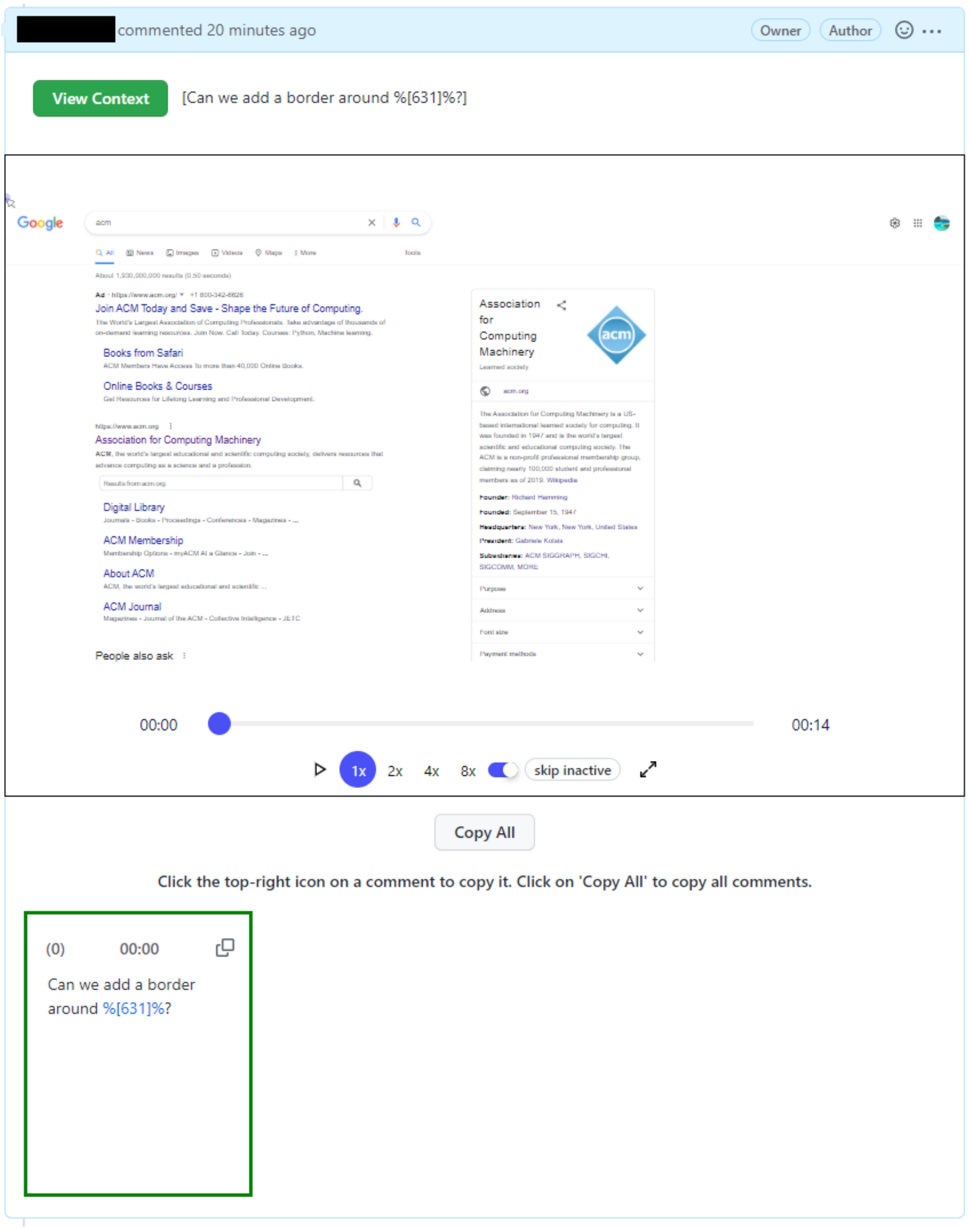
(b) Saved comment

Fig. 4. Editable and saved comments in the *ReferencesPlus* interface. Clicking "Save" changes an editable comment (4a) to a saved comment (4b) while clicking "Edit" converts a saved comment (4b) back to an editable comment (4a).

recording, adjust the replay speed, skip periods with few recorded events, and maximise the content for a better view. *ReferencesPlus* uses this video player interface because record-replay behaviour is camouflaged behind a more familiar view. Unlike a regular video player, this player is augmented to make the recording interactive and hovering over an element will bound it with a red border, indicating that it is focused. To support referencing of visual changes, clicking on a focused element will copy a textual reference to it to the clipboard. This reference takes the form of `%[node_id]%` where `node_id` is a number identifying the referenced element in the recording, and can be pasted into an interface comment.

3.2.3 Using references in communication. Interface comments are the primary way to communicate links between textual content and visual elements. Anchored comments are separated from the GitHub comment box to reduce confusion between content related to the recording and other text. Clicking the "Comment" button underneath the player will generate an editable comment linked to the current time in the recording, and users can add text to the textarea (Figure 4a). Copied references (mentioned in 3.2.2) can be pasted into the textarea as plaintext and will become active when the editable comment is saved and converted into a saved comment (Figure 4b). Saved comments link their textual content to specific UI elements and times in the recording. Clicking a timestamp will move the timeline to that point in time and clicking an element reference will additionally focus the element with a red border. These design choices allow the visual context to be captured in a reference by creating spatial links to a specific element within the recorded web interactions and a temporal link to a specific point in the recording.

3.2.4 Integration with a GitHub Comment. Since the decision was made to separate anchored comments from the regular GitHub comment box, minor effort by the user is required to make those anchored comments visible to others. Users can copy anchored comments as plaintext by clicking the "Copy" button in a comment (Figure 4b) or by clicking "Copy All" button, which simplifies the process of copying many comments (Figure 1). Copied comments can then be pasted into the GitHub comment box as plaintext (Figure 1). When the user makes a comment in the pull request discussion, any text matching an anchored comment is converted into a "View Context" button paired with the anchored


[redacted] commented 20 minutes ago

[Can we add a border around %[631]%?]

Click the top-right icon on a comment to copy it. Click on 'Copy All' to copy all comments.

(0) 00:00

Can we add a border around %[631]%?

Fig. 5. A read-only view of the interface comment embedded in the pull request discussion. The "View Context" button toggles the visibility of the recording and its associated comments, and temporarily highlights the relevant comment with a green border.

comment text. Clicking this button will reveal a read-only view of the interface, complete with the recording linked to that anchored comment and all other anchored comments linked to the recording. The context is hidden behind the button so that users are not overwhelmed by too much information, and can choose when to view it. The read-only view is a simplified version of the editable *ReferencesPlus* interface focused on viewing information, and removes unnecessary controls so the user has fewer distractions.

4 IMPLEMENTATION

ReferencesPlus is a Chrome extension written in TypeScript. TypeScript was chosen instead of JavaScript because it augments JavaScript with static typing and additional features, but compiles down to JavaScript without changing the runtime behaviour [23]. An open source record-replay library called *rrweb*[2] is utilised to enable record-replay functionality. This library captures web interactions, converts them into a serializable data structure with unique identifiers, and reconstructs the data to replay user interactions with the web[3]. The Chrome extension cannot run without compiling the TypeScript files to JavaScript and also needs to access the relevant configuration files to load correctly. To achieve this, the tool uses *webpack*[4], a module bundler with rich plugin support that copies the relevant files to a specific folder and interplays with the TypeScript compiler to produce the necessary JavaScript bundles. The extension uses HTTP requests to connect to a backend server written using Flask, which communicates with a MongoDB database that stores the necessary application state.

Each interface has its own associated state, which is initialized to a particular set of values then updated and saved to the database when certain actions are performed. State consists of the following core attributes: 1) an array of events (a serialized data structure), used to initialize the player, 2) a details object containing the session id and the website linked to the recording, 3) an array of interface comments containing the timestamp, text, and id of the comment, and 4) a number representing the id of the next comment to be created. Message passing APIs are used to share events between the GitHub pull request and the subject of the recording, and also to trigger various actions while the interface is active. To allow recordings to be available to other users besides the creator of the recording, the events are sent using a HTTP request as JSON to the backend server, which then compresses the data with gzip[5] into a binary format and inserts it into the database. With the relevant data now stored in the database, any text patterns in GitHub PR discussions matching interface comments will trigger lookups in the database through the backend server, which will retrieve the relevant state information and provide it to the extension.

5 EVALUATION

To evaluate the *ReferencesPlus* interface, a task-driven user study was conducted. This study compares the tool against the existing GitHub interface for the tasks of comment creation and PR discussion comprehension, which those interacting with PRs are likely to do. To provide more precise definitions about the two interfaces, the existing GitHub interface refers to all functions that GitHub provides to its users in a pull request, along with a screenshot tool which is likely to be available on a user's computer. The *ReferencesPlus* interface is an extension to the existing GitHub interface and users can still make use of the existing features, but users are encouraged to use the tool where possible. This evaluation seeks to answer the following research questions (RQs):

RQ1: Does *ReferencesPlus* reduce the workload experienced by users in comparison to the existing interface?
RQ2: Does *ReferencesPlus* encourage more precise discussions and reduce confusion in communication?
RQ3: How usable is *ReferencesPlus*?

[2]https://www.rrweb.io
[3]https://github.com/rrweb-io/rrweb
[4]https://webpack.js.org/
[5]https://www.gnu.org/software/gzip/

These research questions are concerned with evaluating both the usability and efficacy of the tool in the context of the tasks performed.

A total of eight participants were recruited using convenience sampling. All participants recruited had relevant academic of industry experience in web development, user interface design, and the GitHub pull request interface, although some were more experienced than others. Participants were filtered based on these criteria due to *ReferencesPlus* being designed for web applications and the GitHub interface, meaning that accurate data can only be acquired from participants who meet the tool's intended demographic. The study was conducted online using Zoom[6], with participants installing the extension into Google Chrome on their computer. The operating system of each participant's computer was irrelevant since Chrome extensions are cross-platform and should run on any Chromium browser. Although participants ran different versions of Google Chrome, all participants had a version of Google Chrome that was no older than 3 versions behind the latest version at the time of writing. Once installed, participants were given a short, step-by-step tutorial instructing them on how to use *ReferencesPlus*, followed by an opportunity to ask questions and seek clarification. After completing the tutorial, participants could proceed with the study tasks.

5.1 Methods

To answer the research questions above, two tasks to compare the two interfaces in the context of pull request discussions were designed. A significant concern with the task design is the fact that pull requests are often created by individuals who have some familiarity with the code base associated with pull request, and some individuals may have more experience with the underlying code. To provide fairer comparisons, the difficulty of the tasks was reduced and all participants were forced to be equally familiar with any code being used (not accounting for professional/academic experience). This is achieved by limiting code reading where possible and using a simplified code base where code reading is required. Any code used in the study tasks came from a simple portfolio website provided by the researcher, which is not shared for privacy reasons. This website was chosen due to it being implemented in basic HTML, CSS, and JavaScript, which should be understandable to those with some web development experience.

Each task was divided into two parts, with each part involving a different interface. To limit the effect of factors other than the difference in interface on the task results, the two parts were designed to be similar, with differences other than the choice of interface being avoided if possible. However, making the tasks too similar may cause users who complete the first part of a task to become more familiar with the second part of a task. To mitigate this, some slight differences were introduced and the parts for each task were alternated, meaning that half of the participants started with the existing interface for Task 1 and started with extension for Task 2 while the other half followed the opposite order. On completion of each part of a task, participants were asked to complete a version of NASA's Task Load Index (TLX)[7], which uses various metrics to determine an overall score between 0 and 100. The lower the score, the lower the workload of the task according to the rater. Finally, at the end of each task, participants were asked several open-ended questions to gather more information about their experience with the tool as well as the tool's usability. These questions were:

Q1: What does *ReferencesPlus* enable you to do that the existing interface does not?
Q2: What would you change about the *ReferencesPlus* interface?
Q3: Would you use *ReferencesPlus* in a practical i.e. in the workplace/actual project setting?

[6]https://zoom.us/
[7]https://humansystems.arc.nasa.gov/groups/tlx/

5.1.1 Task 1: Discussion Comprehension. The first study task focuses on understanding the discussion of an existing PR. Two similar PRs making minor code changes to HTML, CSS, and JavaScript were created. Both PRs made changes to the layout, color, and DOM of the website, and also made some animation changes to demonstrate the tool's ability to capture dynamism in user interfaces. Apart from the differences in the code changes, the two PRs differed in how they were created. One PR used only the existing GitHub interface, meaning that only text and some appropriate screenshots were used, while the other PR used the *ReferencesPlus* interface where possible but also included regular text comments. Participants were asked to study the portfolio website and look at its design for 5 minutes, as well as think about how to identify elements of the website and communicate them to other people. After studying the website, they were asked to read through a pull request for the first part of the task with the aim of understanding the changes discussed in the pull request. After that, participants were given several test questions related to the changes and asked to identify where the change would show up in the original portfolio website, which represents the before state of the PR change. Upon completion of questions, participants completed a TLX survey, then repeated the task of studying a pull request and answering test questions with the PR and test questions used for the second part of the task.

5.1.2 Task 2: Comment Creation. The second study task focuses on creating a comment on an existing PR. This task gives participants the chance to explore more features of both interfaces, make references to more complex behaviour, and capture more dynamic aspects of user interfaces. The portfolio website used in Task 1 was not used in this task because it was considered to be too simplistic and therefore did not have the necessary complexity to fully use the features of the two interfaces. As previously mentioned, most people involved with PRs have some familiarity with the codebase associated with the PR. However, there are significant challenges with using a complex codebase, and requiring participants to understand the code can become a significant confounding factor as the task would also test code comprehension, which is not intended. To address this concern, two dummy PRs (one for each part) with no relation to the target website were used even though they do not represent real PRs. This is because the focus of the task is creating a comment and making references to visual elements, rather than reviewing the PR. The websites used for this task were the Microsoft Canada page[8] and Samsung Canada page[9], chosen because they were both websites for large technology companies, had several similarities, but also had clear differences. To mitigate the effect of ordering, half of the participants commented on the Microsoft website with the existing GitHub interface while others commented on the Samsung website, leaving the remaining website to commented on using *ReferencesPlus*. Participants would study one website, looking for things they liked, disliked, or wanted to change about the website's design. After that, participants were asked to create a comment in the dummy PR with 1-2 sentences about the following 5 categories: menu design, font and colour, animation, general layout, and information density. After creating a comment, participants completed a TLX survey, then repeated the steps of studying a website and creating a comment in the PR addressing the 5 categories with the PR and website used for the second category.

5.2 Results

5.2.1 Task 1 Results. The time taken for participants to answer questions was measured in seconds, and their understanding was tested with a series of questions worth 4 points in total. Finally, the TLX surveys that participants completed produced a score between 0 and 100, measuring the workload that participants experienced while performing the task with each interface. These results are summarised in Figure 6, which displays the averages of several measures

[8]https://www.microsoft.com/en-ca/
[9]https://www.samsung.com/ca/

that were captured in the task. It was discovered that participants took approximately 22% longer to answer the questions using *ReferencesPlus* compared to the existing GitHub interface. However, there was a small improvement in the test accuracy as participants had a 2.5% better score when answering questions about the PR that used *ReferencesPlus* as part of the discussion. Finally, the average TLX score for the tool was lower than the TLX score for the existing interface by roughly 16% in the context of an existing discussion.

Task 1: Discussion Comprehension - Results		
Means	**Existing Interface**	***ReferencesPlus***
Time to answer all test questions (s)	300	365
Test score (/4)	3.63	3.72
TLX score	50.75	42.83

Fig. 6. Results from Task 1, which required participants to understand an existing pull request, answer some questions to test their understanding, and fill out a TLX survey. This task was repeated twice: once with the existing interface and once with *ReferencesPlus*.

5.2.2 *Task 2 Results.*

The time taken for participants to create a comment in a pull request was measured in seconds, and the scores produced by the TLX surveys have the same scale as those from Task 1. These results are summarised in Figure 7, which displays the averages of several measures that were captured in the task. The results indicate that participants took about 83% longer, which is nearly twice as long, to create a comment with *ReferencesPlus* than with the existing GitHub interface. It is important to note that for this task, the time taken to capture web interactions with the tool was factored into the time. However, the average TLX score for the tool was once again lower than the TLX score for the existing interface. In this task, the difference was smaller than in Task 1, with a decrease of approximately 10% compared to the existing interface.

Task 2: Comment Creation - Results		
Means	**Existing Interface**	***ReferencesPlus***
Time to create a comment (s)	197	360
TLX score	48.37	43.46

Fig. 7. Results from Task 2, which required participants to study the appearance and design of a website, create a comment on a PR addressing several aspects of the website's design, and fill out a TLX survey. This task was repeated twice: once using existing interface and once using *ReferencesPlus*.

6 DISCUSSION

The results of the study, including the feedback gathered from participants in response to the open-ended questions, are discussed through the lenses of the three proposed research questions (RQs). Additionally, the limitations of the study are addressed.

6.1 RQ1: Reducing User Workload in Pull Request Discussions

Participants in the study indicated that the workload of using *ReferencesPlus* in PR discussions was less than the workload associated with the existing interface. This effect was more pronounced in the discussion comprehension scenario than the comment creation scenario, suggesting that the tool has a greater impact on those trying to understand existing discussions. Although the time taken to complete both study tasks was longer using the tool, it appears as though this did not significantly shift the workload scores in favour of the existing interface.

Feedback from participants was mostly positive, with many reporting that *ReferencesPlus* made it easier for them to refer to specific elements in a website and visualize code changes to UI elements. The existing interface's support for screenshots requires additional effort since they may need to be captured multiple times and manually annotated, whereas *ReferencesPlus* records interactions once and captures those annotations implicitly. Even among participants who viewed the tool favourably, the workflow was a major concern. Most said that there were too many steps in the current workflow, and some steps, such as copying the anchored comments into the GitHub comment box, felt unnecessary. A particularly confusing step was the action of clicking the "Activate" button (Figure 1) before starting the recording, as the instinct of many participants was to directly go to the website and start the recording. While the current design of the tool does reduce user workload, there are several areas of improvement that can potentially produce further reductions.

6.2 RQ2: Precise Communication and Confusion in Discussions

The quantitative data pertinent to **RQ2** does not seem to indicate that *ReferencesPlus* encourages more precise discussion with fewer instances of confusion. The average score of participants was only approximately 2.5% better when using the tool, which is likely within margin of error given the small sample size. A potential explanation for this could be the design of the questions and on a deeper level, the choice of code base. In the interests of reducing complexity, the portfolio website was chosen for Task 1 as it is implemented with more basic web technologies and does not require a lot of background for someone with web development experience to understand. However, it lacks the dynamism and more complex animations required to fully showcase the features of the tool. In order to constrain the duration of a study session, only four test questions for each pull request were asked to each participant, which is fairly small. Asking more questions could reduce the impact of each individual question and potentially lead to a greater difference.

Participant feedback regarding this research question was more favourable, with most reporting that they could make more explicit references to changes and elements. Being more explicit in commenting also meant that participants could directly identify the element(s) mentioned in the captured interactions. On the other hand, there were several suggestions to further improve the tool's ability to support the communication of precise visual references. In the current implementation of the tool, clicking the "View Context" button (Figure 5) reveals all anchored comments associated with the recording and only highlights the selected comment for a short duration. Including all other anchored comments distracts from the relevant comment, so improvements should make the relevant comment more obvious by either permanently highlighting it or removing other anchored comments. Also, highlighting any references to visual elements in an anchored comment when viewing its full context would make those references more apparent. Overall, more work needs to be done before the tool can support very precise references in discussions.

6.3 RQ3: Overall Usability

Quantitative data to measure the usability of the tool was not collected, although the TLX survey scores may indirectly measure this by means of workload. The fact that *ReferencesPlus* had a lower average score for both study tasks may indicate that it was usable, since working with the existing interface was inversely considered more challenging. Most participants said that they would be willing to use the tool in a practical (workplace or project) setting, although some acknowledged that they would only use this tool in particular use cases. The benefit of the tool for frontend development was mentioned several times, as this would allow for precise references and achieve things that are harder to do with existing methods.

Nevertheless, there were a few concerns with the usability of the tool. As previously mentioned, participants raised concerns about the complexity of the workflow, which contained too many steps that were hard to remember. Another deficiency of the tool was the lack of visibility regarding the status of a recording, as participants do not have any way of knowing whether web interactions are being recorded besides an alert when the user first clicks the menu item to record the page. Having a persistent notification or indicator that informs the user of the status of the recording would be very helpful. Users on small monitors often had to scroll up and down to view the entire *ReferencesPlus* interface, suggesting that a more compact interface is preferred. Finally, the integration between the anchored comments and the GitHub comments is not very usable, since the full details of the comment are only available in the hidden context. Participants suggested making anchored comments more interactive within a GitHub comment by allowing element references to directly show the full context of the reference. They also proposed more readable names for references, since the `node_id` values had no meaning to users. Overall, users expressed general satisfaction with *ReferencesPlus* but suggested several clear pathways for improvement. More work needs to be done to refine the tool and improve its usability.

6.4 Limitations

There are several limitations to this evaluation. In the design of the study, the expectation that individuals working on pull requests typically have some familiarity with the code base was identified as a point of concern. Despite addressing this issue by reducing the complexity of the tasks, this results in the study tasks being less representative of typical pull requests in a workplace or open source project. Following on from this point, the complexity of the code was also reduced, which makes the study tasks less representative of typical pull requests since the code base is likely too simple to use the pull-based software development model. Lastly, the evaluation was only performed on 8 participants with web development and user interface design experience, and all participants only had a short time to familiarize themselves with the tool. Due to the small sample size and short learning period, the conclusions drawn from the data cannot provide definitive answers to the research questions posed. A study with more participants and a longer duration may allow the efficacy and feasibility of the tool in the context of the research questions to be more thoroughly evaluated.

There are also some limitations with the implementation of *ReferencesPlus*. Currently, some website assets such as images and fonts may not load correctly. This is likely because of the Content Security Policy of GitHub, which poses a problem for making references to certain elements as they may not be correctly loaded. There are workarounds to resolve this problem, which should be investigated as part of future research or improvements to the tool. Another constraint imposed for security reasons is that JavaScript scripts are not captured by the record-replay library, as they could potentially run malicious code. This may not be resolvable without a deep investigation into the inner workings of the record-replay library or the discovery of some currently unknown solution. Finally, the recordings that users create cannot be manually retrieved so users may have to create a new recording even when a usable one exists in the database. This can be addressed via a session management system and a more robust database schema design that captures the reusability of recordings in its structure.

7 CONCLUSION

This paper investigates an interface designed to capture dynamic user-interface changes and embed them within pull request discussions to facilitate precise visual referencing. Upon evaluating the interface, it was discovered that the interface reduced the workload experienced by participants when compared to the existing pull request interface, and was capable enough to be considered for use in a practical setting. Additionally, it is likely that the interface enables more precise referencing and reduces confusion among discussants, but further investigation is required to properly assess this. The evaluation also suggests several directions for future research. A more robust version of the tool could be developed that incorporates the feedback shared during the evaluation, which could then be evaluated in a more rigorous study to full understand the capabilities of the application as a visual referencing tool. The implementation of this tool can also be used as a base to integrate more forms of referencing, such as code search, into GitHub pull request discussions. There are several suggestions by Chopra et al. [7] not addressed in this paper, which could be implemented as separate applications or incorporated into this tool to address multiple concerns at once. Finally, reframing this tool as something to democratise code reviews and enable those with limited coding experience to provide input may result in exciting new design directions and possibilities.

ACKNOWLEDGMENTS

This work was partially supported by the Huawei-UBC Joint Software Engineering Research Program. I would like to express my sincere gratitude to Dr Dongwook Yoon and Dr Ivan Beschastnikh from the University of British Columbia for their supervision and guidance. Their advice and feedback was crucial to the completion of this thesis and I am grateful to have had this opportunity to work with them. I would also like to thank Dr Cleidson de Souza, Yiyuan Sun, and Suki Cai for their support and feedback. They have shared many insightful ideas that have informed the writing of this paper. Next, I would like to express my gratitude to Zoe Gaouette, my classmate and dear friend, for all the kindness that you have shared with me over the years. Your help as a project partner, a fellow student, and most importantly as a friend has been essential to the completion of my degree and this thesis. Finally, I would like to thank my parents for supporting me throughout my life and in my academic journey, and my younger brother for filling my life with fun and laughter.

REFERENCES

[1] Alberto Bacchelli and Christian Bird. 2013. Expectations, outcomes, and challenges of modern code review. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 712–721. https://doi.org/10.1109/ICSE.2013.6606617

[2] Deepika Badampudi, Ricardo Britto, and Michael Unterkalmsteiner. 2019. Modern code reviews - Preliminary results of a systematic mapping study. In *Proceedings of the Evaluation and Assessment on Software Engineering*. ACM, 340–345. https://doi.org/10.1145/3319008.3319354

[3] Vipin Balachandran. 2013. Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 931–940. https://doi.org/10.1109/ICSE.2013.6606642

[4] Nicky Bleiel. 2016. Collaborating in GitHub. In *2016 IEEE International Professional Communication Conference (IPCC)*. IEEE, 1–3. https://doi.org/10.1109/IPCC.2016.7740497

[5] Yuan-Chia Chang, Hao-Chuan Wang, Hung-kuo Chu, Shung-Ying Lin, and Shuo-Ping Wang. 2017. AlphaRead: Support Unambiguous Referencing in Remote Collaboration with Readable Object Annotation. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. ACM, New York, NY, USA, 2246–2259. https://doi.org/10.1145/2998181.2998258

[6] Yan Chen, Sang Won Lee, and Steve Oney. 2021. CoCapture: Effectively Communicating UI Behaviors on Existing Websites by Demonstrating and Remixing. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, 1–14. https://doi.org/10.1145/3411764.3445573

[7] Ashish Chopra, Morgan Mo, Samuel Dodson, Ivan Beschastnikh, Sidney S Fels, and Dongwook Yoon. 2021. "@alex, this fixes 9": Analysis of Referencing Patterns in Pull Request Discussions. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–25. https://doi.org/10.1145/3479529

[8] Soon Hau Chua, Toni-Jan Keith Palma Monserrat, Dongwook Yoon, Juho Kim, and Shengdong Zhao. 2017. Korero: Facilitating Complex Referencing of Visual Materials in Asynchronous Discussion Interface. *Proceedings of the ACM on Human-Computer Interaction* 1, CSCW (2017), 1–19.

https://doi.org/10.1145/3134669

[9] Herbert Clark. 1996. *Using Language.* Cambridge University Press, Cambridge, United Kingdom. 446 pages.

[10] Herbert H. Clark and Susan E. Brennan. 1991. Grounding in communication. In *Perspectives on socially shared cognition.* American Psychological Association, Washington, 127–149. https://doi.org/10.1037/10096-006

[11] Philip R. Cohen. 1992. The role of natural language in a multimodal interface. In *Proceedings of the 5th annual ACM symposium on User interface software and technology - UIST '92.* ACM Press, New York, New York, USA, 143–149. https://doi.org/10.1145/142621.142641

[12] GitHub. 2022. About apps. https://docs.github.com/en/developers/apps/getting-started-with-apps/about-apps

[13] GitHub. 2022. About writing and formatting on GitHub. https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/about-writing-and-formatting-on-github

[14] GitHub. 2022. Attaching files. https://docs.github.com/en/get-started/writing-on-github/working-with-advanced-formatting/attaching-files

[15] Ivo Gomes, Pedro Morgado, Tiago Gomes, and Rodrigo Moreira. 2009. An overview on the static code analysis approach in software development. *Faculdade de Engenharia da Universidade do Porto, Portugal* (2009).

[16] Google. 2022. chrome.contexMenus. https://developer.chrome.com/docs/extensions/reference/contextMenus/

[17] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An exploratory study of the pull-based software development model. In *Proceedings of the 36th International Conference on Software Engineering.* ACM, 345–355. https://doi.org/10.1145/2568225.2568260

[18] Austin Z. Henley, Kıvanç Muşlu, Maria Christakis, Scott D. Fleming, and Christian Bird. 2018. CFar: A Tool to Increase Communication, Productivity, and Review Quality in Collaborative Code Reviews. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* ACM, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173731

[19] Zhewei Hu and Edward F Gehringer. 2019. Improving Feedback on GitHub Pull Requests: A Bots Approach. In *2019 IEEE Frontiers in Education Conference (FIE).* IEEE, 1–9. https://doi.org/10.1109/FIE43999.2019.9028685

[20] Jing Jiang, Jiangfeng Lv, Jiateng Zheng, and Li Zhang. 2021. How Developers Modify Pull Requests in Code Review. *IEEE Transactions on Reliability* (2021), 1–15. https://doi.org/10.1109/TR.2021.3093159

[21] Tae Soo Kim, Seungsu Kim, Yoonseo Choi, and Juho Kim. 2021. Winder: Linking Speech and Visual Objects to Support Communication in Asynchronous Collaboration. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* ACM, 1–17. https://doi.org/10.1145/3411764.3445686

[22] James W Mickens, Jeremy Elson, and Jon Howell. 2010. Mugshot: Deterministic Capture and Replay for JavaScript Applications.. In *NSDI*, Vol. 10. 159–174.

[23] Microsoft. 2022. TypeScript for the New Programmer. https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html

[24] Martin Mühlpfordt and Martin Wessner. 2005. Explicit referencing in chat supports collaborative learning. (2005).

[25] Sebastian Muller, Michael Wursch, Thomas Fritz, and Harald C. Gall. 2012. An approach for collaborative code reviews using multi-touch technology. In *2012 5th International Workshop on Co-operative and Human Aspects of Software Engineering (CHASE).* IEEE, 93–99. https://doi.org/10.1109/CHASE.2012.6223031

[26] Mohammad Masudur Rahman and Chanchal K Roy. 2014. An insight into the pull requests of GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014.* ACM Press, 364–367. https://doi.org/10.1145/2597073.2597121

[27] Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. 2018. Modern Code Review: A Case Study at Google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice.* ACM, 181–190. https://doi.org/10.1145/3183519.3183525

[28] Moran Shochat, Orna Raz, and Eitan Farchi. 2009. SeeCode – A Code Review Plug-in for Eclipse. In *Haifa Verification Conference.* 205–209. https://doi.org/10.1007/978-3-642-01702-5_21

[29] Devarshi Singh, Varun Ramachandra Sekar, Kathryn T. Stolee, and Brittany Johnson. 2017. Evaluating how static analysis tools can reduce code review effort. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).* IEEE, 101–105. https://doi.org/10.1109/VLHCC.2017.8103456

[30] Matin Yarmand, Dongwook Yoon, Samuel Dodson, Ido Roll, and Sidney S Fels. 2019. "Can you believe [1:21]?!": Content and Time-Based Reference Patterns in Video Comments. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems.* ACM, 1–12. https://doi.org/10.1145/3290605.3300719