Project Proposal CSE 503:

UITalk: Enabling Blind and Visually Impaired Developers to Build Visual Web Layouts

Venkatesh Potluri (vpotluri), Christine Chen (chenc55), Liang He (lianghe)

MOTIVATION

Access to information regarding user interface layouts is essential to accomplish a plethora of tasks, ranging from the ability to better navigate User Interfaces (UIs) to the ability to build one's own representation of ideas (*e.g.,* prototypes), thoughts (*e.g.,* blogs) and online presence (*e.g.,* personal and professional websites). Our observation shows that the UI is the first thing app developers are taught to build [8, 9]. Despite the emphasis on visual interfaces, the necessary information to be able to build a well-designed visual interface is not provided by existing tools for Blind and Visually Impaired (BVI) developers. Though the underlying code (XAML for windows, Android XML for Android, and CSS for web) are inherently accessible, the outputs (or the effects of changing this code) are not accessible to a BVI developer. From one of our team member's own experience as a BVI developer, and from an on-going study with BVI individuals, we identified the following aspects of UI development that are inaccessible.

1. *Template selection.* BVI developers do not have the necessary information to select a template to build upon. In fact, our ongoing study shows that no BVI person selected a template or chose how their website looks without assistance from a sighted person.
2. *Conform to design guidelines.* There are visual design guidelines for developing webpages such as consistent spacing, proper alignment of UI elements, and proper arrangement of layout across screens of different sizes. Existing tools for BVI developers do not give information about adherence to these guidelines while building or modifying visual web UI design. Currently, BVI users' access to the web is limited to the DOM tree and websites where designers have taken care to follow accessibility guidelines
3. *Make changes that do not break the visual design of the website*. For example, a trivial text change on the webpage can result in overlapping text. To validate this further, our ongoing study shows that BVI persons are not comfortable making any changes (*e.g.,* content or stylistic) to their websites and pushing them out without having a visual check by a sighted individual.

For this course project, we will not be addressing 1 (*i.e.,* template selection). Future work will look into utilizing tangibles to convey the various aspects of visual templates. As we address 2 and 3, we foresee the following challenges.

a) Identify design guidelines and provide a real-time mechanism that checks that the webpage UI changes BVI developers make conform to the design guidelines.
b) Create novel interaction techniques that allow BVI developers to change webpage UI design without support from a sighted individual.
c) Direct the BVI developer's focus to the code snippet corresponding to the element they are editing, and convey the visual aspects of the changes made in a non-visual way.

Goals, Contributions, and Limitations

Our primary goal for the scope of the class is to enable BVI developers to edit webpages without breaking visual design guidelines. To achieve this goal, we will (i) add five design guidelines to VizAssert, in addition to its existing usability and accessibility guidelines; (ii) build a tool that leverages these design guidelines to support BVI developers to edit webpages without violating UI design; and (iii) propose interaction techniques that link UI updates on a tablet to code changes in a code editor and facilitate non-visual feedback for UI design and code edits.

For our work, we will restrict our investigation and solutions to the context of web page design. In reality, building a visual interface does not require programming experience; sighted individuals can build their webpages with

WYSIWYG ("What You See Is What You Get") editors[1] (*e.g.,* wix.com). However, this paradigm that abstracts away technical details is not accessible to BVI developers. In this course project, the user of our tool is intended to be a BVI developer and we assume prior web programming knowledge and experience.

TECHNICAL DETAILS

In this section, we describe our proposed system and the design guidelines we plan to incorporate.

System

First, we will manually create a set of webpage wireframes for debugging and testing. We will then build tool(s) to enable editing webpage UIs. In doing this, we will add rules corresponding to our proposed design guidelines to VizAssert [6]. Our tool will leverage VizAssert's capabilities to programmatically analyze and alert the user to changes that break visual web design. To enable BVI developers to modify web page UIs, we plan to use a multimodal and multi-device approach that leverages touchscreen sensing for BVI developers to directly manipulate UI elements and make webpage design changes. We also plan to support direct code editing through a physical keyboard and a desktop or laptop computer, the environment and input device familiar to BVI developers to perform development related tasks. Figure 1 shows our proposed system.

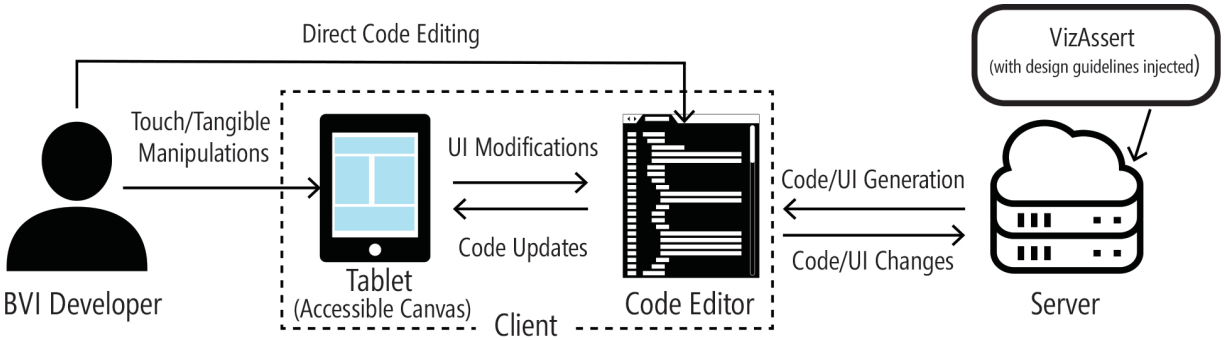

Figure 1. Overview of our proposed system: The BVI developer can change the UI design directly on the tablet or edit code in an editor. The UI modifications and code updates will be synchronized in real time. The generated code and UI modifications are ruled by design, accessibility, and usability guidelines on the server.

BVI developers can make modifications to an existing webpage wireframe on tablet touchscreen (*i.e.*, the accessible canvas). While prior work has investigated gestures [8], capacitive widgets [2], tactile toolkits [9], physical feedback [1] to enable and enhance blind users'interactions with touchscreens, we will propose a set of touch-based interaction techniques powered by the touchscreen and the screen reader that support the UI design for BVI developers. We will detect the user's finger position and movements on the touchscreen and provide continuous audio feedback. For example, a *div* element can be moved and resized by finger gestures (*e.g.*, pinching). The screen reader provides audio updates on these changes. We however provide gestures for illustrative purposes; the final set of gestures may vary.

Besides making modifications to UI elements, the BVI developer can also directly edit the HTML code in a code editor by typing on a keyboard. While the user is editing the code, the updates will immediately be reflected on the corresponding UI element on the tablet. Likewise, the modifications made to the UI elements on the tablet will be directly highlighted as changes in the code snippet. That is, the BVI developer will be informed of these code snippets non-visually. We plan to use the infrastructure (accessible editor and debugger) that Microsoft's Visual Studio Code[2] provides to support editing and reading code corresponding to the UI that a BVI developer chooses to edit. While the IDE choice might not be very important for non-BVI developers, it is particularly salient in our case as VS Code is one of the few IDEs that is accessible to BVI developers. More specifically, alertability [7] (*i.e.,* real-time access to information) is an accessibility challenge without existing solutions. Thus, we propose to use VS Code as it supports ARIA[3] and enables us to give BVI developers real-time access to code related information without changing the screen reader's focus. For example, by using ARIA, we can make the screen reader on the desktop announce "*DIV selected from Line 42 to Line 48*" when the user touches a *div* element on the tablet.

[1] WYSIWYG editor: https://www.mintox.com/support/wysiwyg-editor/introduction-to-the-wysiwyg-editor
[2] Microsoft Visual Studio Code: https://code.visualstudio.com/
[3] WAI-ARIA overview: https://www.w3.org/WAI/standards-guidelines/aria/

As the accessible canvas on the tablet and the code editor on the computer are the clients in our system, we will inject the design guidelines (described in the following section) into the VizAssert framework to provide guidance to the BVI developer towards better visual design of the webpage on the server side. When the BVI user makes UI changes on the client side, the UI changes will be sent to the server and new HTML code will be generated and sent back to the client (updating both the design in the app and the code in the editor).

Design Guidelines

We propose five design guidelines [3, 11] which provide suggestions for a better and more usable web page. We will incorporate them into the VizAssert infrastructure.

- *Responsive UI design*. Since people access web pages on different devices, such as tablet, desktop, and phone, all the elements and content should be organized and displayed correctly on different screen sizes.
- *UI elements should be aligned properly*. The design elements (text, images and content containers) on the web page are organized in a hierarchy. The elements at the same hierarchy should be aligned properly (left-aligned, center-aligned, and right-aligned).
- *Typeface and font size*. Too many variations in font types can be distracting, confusing, and borderline annoying. A common recommendation is to use a maximum of three different typefaces in a maximum of three different sizes.
- *Spacing consistency*. Horizontal and vertical rhythms are equally important in a web page. Varied vertical or horizontal spacing causes visual noise. Consistent spacing between grids (if elements are organized inside grids) create cleaner content that can be easier to consume.
- *Color consistency*. Color is used to group related items and similar colors infer a similarity among objects. UI elements should share the same color scheme, for example, all hyperlinks should have the same color throughout the webpage. Avoid red/green, blue/yellow, green/blue, and red/blue combinations unless a special visual effect is needed. They can create vibrations, illusions of shadows, and afterimages.

EVALUATION CRITERIA

For evaluation, we will conduct our study with two participants: an internal pilot and a BVI developer who is not associated with this project. For the scope of this class, this will help us identify and fix issues with the protocol.

Our current plan includes:

- First, we will ask the pilot participant to design a simple website using the tablet OR provide a simple website for the participant to modify.
- We will assign several predefined tasks/scenarios to the participant based on our knowledge of the design guidelines. Some tasks should result in acceptable modifications that do not trigger warnings and other tasks should trigger warnings due to violations of design guidelines. For example, we might ask the participant to complete a task such as changing the font for five different text boxes in such a way that a design violation occurs (*e.g.*, there are now too many different types of fonts on the webpage). This will allow us to see how the participant responds to the alert (in terms of the level of intrusiveness, usefulness of information in the alert, *etc*.). For the tasks that do not trigger warnings, this will allow us to determine if the participant can understand the webpage UI and improve the design using our tool(s).
- After the test, we will ask the participant to rate the tool(s) according to the system usability scale [10].

REFERENCES

1. Bau, O., Poupyrev, I., Israr, A., & Harrison, C. (2010, October). TeslaTouch: electrovibration for touch surfaces. In Proceedings of the 23nd annual ACM symposium on User interface software and technology (pp. 283-292). ACM.
2. Chan, L., Müller, S., Roudaut, A., & Baudisch, P. (2012, May). CapStones and ZebraWidgets: sensing stacks of building blocks, dials and sliders on capacitive touch screens. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 2189-2192). ACM.

3. Leavitt, M. O., & Shneiderman, B. (2006). based web design & usability guidelines. Background and Methodology.
4. Kane, S. K., Wobbrock, J. O., & Ladner, R. E. (2011, May). Usable gestures for blind people: understanding preference and performance. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 413-422). ACM.
5. Kane, S. K., Morris, M. R., & Wobbrock, J. O. (2013, October). Touchplates: low-cost tactile overlays for visually impaired touch screen users. In Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility (p. 22). ACM.
6. Panchekha, P., Geller, A. T., Ernst, M. D., Tatlock, Z., & Kamil, S. (2018, June). Verifying that web pages have accessible layout. In Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (pp. 1-14). ACM.
7. Potluri, V., Vaithilingam, P., Iyengar, S., Vidya, Y., Swaminathan, M., & Srinivasa, G. (2018, April). CodeTalk: Improving Programming Environment Accessibility for Visually Impaired Developers. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (p. 618). ACM.
8. Build a simple user interface | Android Developers. (n.d.). Retrieved from https://developer.android.com/training/basics/firstapp/building-ui.html
9. R., Wenderlich. (2018, July 24). Your First iOS App · Challenge: Making a Programming To-Do List. Retrieved February 24, 2019, from https://www.raywenderlich.com/5993-your-first-ios-app/lessons/2
10. System usability scale: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html
11. Web design principles: https://theblog.adobe.com/12-dos-donts-web-design-2/