Project Proposal CSE 503: 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 (personal and professional websites). Our observation shows that the UI is the first thing app developers are taught to build [1, 2]. 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 to 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 up on. 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*. BVI developers are not aware of design guidelines. Their access modalities to the web are limited to the DOM tree and accessibility specific guidelines met by website designers.
3. *Make changes that do not break the visual layout of the website*. In our ongoing study, we observed that BVI persons are not comfortable making changes to their websites and pushing them out without having a visual check by a sighted individual.

We propose using Viz-Assert [6] as the infrastructure driving the verification for aforementioned 2 and 3. For each of the components mentioned above, we foresee the following challenges.

1. Identify non-visual interactions to convey visual templates. This could be done either through tangibles or by customizing touch interactions on tablets with the multi-touch screen.
2. Identify design guidelines, evaluate feasibility of implementing these in Viz-Assert.
3. Direct 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, Limitations, and Contributions

Our primary goal for the scope of the class is (1) to enable BVI developers to edit layouts without breaking visual changes and (2) to enable BVI users to make these changes conforming to some design guidelines. While the examples above emphasize the importance of visual aspects of interfaces in mobile apps, the challenges are relevant in designing web pages as well. 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 like wix.com. Though this is the ideal we would like to meet, we will scope down to assuming that the user of our tool is a BVI developer.

In this work, we contribute in three aspects: (1) propose a method that allows the BVI developer to directly explore and manipulate UI element on a touchscreen; (2) build a tool that supports UI design for BVI developers; and (3) integrate web design guidelines into the design experience for BVI developers.

TECHNICAL DETAILS

We are not focusing on enabling template selection and interpretation for BVI developers at scale in the scope of this course. We will however manually create a set of web page wireframes for debugging and testing. We will build tool(s) to enable editing spatial layouts. Our tool will rely on Viz-Assert to programmatically analyze and alert the user to changes that break the visual layout. In addition to existing tools for ensuring that webpages are accessible, we will add rules corresponding to design guidelines to Viz-Assert. We plan to use the infrastructure (accessible

editor and debugger) that Microsoft's Visual Studio Code provides to support editing and reading code corresponding to the UI that a BVI developer chooses to edit. To enable BVI developers to modify the web page UIs, we plan to use a multimodal and multi-device approach that leverages a tablet touchscreen for sensing the BVI user's input, and a physical keyboard and a desktop or laptop computer for code editing. Figure 1 shows an overview of our proposed system.

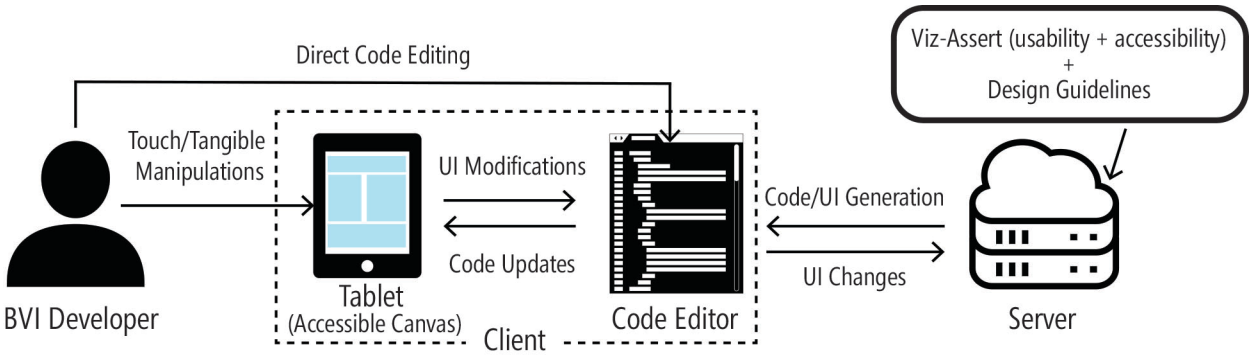

Figure 1. Overview of our proposed system: The BVI developer can change the UI design directly on the tablet or edit code in an editor. The UI modifications and code updates will be synchronized in real time. The generated code and UI are ruled by design, accessibility, and usability guidelines on the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

System Workflow

First, BVI developers can directly manipulate the selected webpage UI template (we manually create for this course) on a tablet. Then, BVI developers can make modifications to the template. We will explore two interaction techniques to enable BVI developers to input UI design modifications: (i) using multi-touch sensing offered by the touchscreen and direct manipulation abilities offered by touch screen based screen readers, and (ii) creating a set of tangibles that provide tactile feedback to BVI developers. For the first approach, we will detect the user finger's position and movements on the touchscreen and provide continuous audio feedback. For example, the app we create (accessible canvas) can detect that the finger is touching a *div* element and the element moves when the finger moves. The screen reader will read and speak the position of the *div* on the webpage. When the user touches the screen with another finger and without releasing the previous finger, the user can change the *div*'s width by horizontally pinching the two fingers. Again, the screen reader reports the updated size of the *div*. For the tangibles based approach, we add one more layer between the touchscreen and the user's finger, therefore providing tactile feedback while the user is manipulating UI elements.

The BVI developer can also directly edit the HTML/CSS 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 (*i.e.*, the accessible canvas). Likewise, the modifications made to the UI elements on the tablet will be directly highlighted as changes in the code snippet. Both the accessible canvas on the tablet and the code editor on the computer are the clients in our system.

On the server side, we reuse Viz-Assert system to provide usability and accessibility guidelines. In addition, we will inject the design guidelines, which will be described in the following section, into Viz-Assert framework to provide guidance to the BVI developer towards better visual design of the webpage. When the BVI user makes UI changes on the client side, the UI changes will be recorded on the server and our system will generate new HTML/CSS code and send it 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, 4] which provide suggestions for a better and more usable web page. We will incorporate them into the Viz-Assert 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 an internal pilot where Venkatesh will test out the functionality of the tool(s) developed. Though not equivalent to running a full-on study with BVI participants, 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*.).
- We also ask the participant to freely explore and redesign a given webpage UI following predefined tasks. In this test, we would like to examine if the participant can understand the webpage UI and improve the design by using our tool(s).
- After the test, we will ask the participant to rate the tool(s) according to the system usability scale [5].

REFERENCES

1. Build a simple user interface | Android Developers. (n.d.). Retrieved from https://developer.android.com/training/basics/firstapp/building-ui.html
2. 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
3. Web design principles: https://theblog.adobe.com/12-dos-donts-web-design-2/
4. Leavitt, M. O., & Shneiderman, B. (2006). based web design & usability guidelines. Background and Methodology.
5. System usability scale: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html
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.