Poster: Testing Web-based Applications With the <u>V</u>oice <u>C</u>ontrolled <u>A</u>ccessibility and <u>T</u>esting Tool (VCAT)

Nagendra Prasad Kasaghatta Ramachandra, Christoph Csallner
Computer Science and Engineering Department
The University of Texas at Arlington
Arlington, TX, USA
nagendra.ramachandra@mavs.uta.edu,csallner@uta.edu

ABSTRACT

Many web applications and software engineering tools such as test generators are not accessible for users who do not use traditional input devices such as mouse and keyboard. To address this shortcoming of current applications, this work leverages recent speech recognition advances to create a browser plugin that interprets voice inputs as web browser commands and as steps in a corresponding test case. In an initial experiment, the resulting Voice Controlled Accessibility and Testing tool (VCAT) prototype for Chrome and Selenium yielded a lower overall runtime than a traditional test creation approach.

CCS CONCEPTS

• **Human-centered computing → Accessibility systems and tools**; • **Software and its engineering → Software testing and debugging**;

KEYWORDS

Accessible software engineering tool, voice control, Selenium

ACM Reference Format:
Nagendra Prasad Kasaghatta Ramachandra, Christoph Csallner. 2018. Poster: Testing Web-based Applications With the Voice Controlled Accessibility and Testing Tool (VCAT). In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3195099

1 INTRODUCTION

Many applications assume that their users will primarily interact with the application via traditional input devices such as keyboard, mouse, or touch screen. For example, this is a common assumption made by many web sites and by many professional software engineering tools. These applications often neglect accessibility via alternative input devices. This becomes a problem if a user cannot or does not want to use one of the standard input devices.

The current applications' problem of lacking accessibility is significant in practice, as many people (e.g., due to an injury or disability) cannot use or have a hard time using traditional computing input devices. This lack of accessibility can thus exclude many people from widely used applications that are otherwise part of daily life, such as popular web pages or software engineering tools.

As a concrete example, in this paper we focus on writing test cases with the popular Google Chrome browser and the popular Selenium [6] test case generation and management tool. In a typical workflow, a software engineer (1) first uses Chrome to explore the web site application under test. (2) In a second step, the engineer writes a Java test case in the Selenium framework, to codify the actions taken in step (1). In many cases the software engineer uses for input in both steps keyboard and mouse. While the general domain of accessibility has seen a lot of important contributions over the last decades, it is still hard today to use popular software products such as Chrome and Selenium without mouse or keyboard. Specifically, several approaches exist to automatically transcribe screen contents to voice and provide such program output to the user. However going the opposite direction, translating from user commands to program input, seems to have less support. The likely reason for this mismatch is that going from screen to text mainly requires transcribing known program entities such as strings, but going the other direction may require very complex tasks such as natural language processing, to understand user commands.

Machine learning, natural language processing (NLP), and voice recognition are active areas of research. Only recent years have seen the emergence of relatively robust voice recognition systems such as Alexa [1], Cortana [4], and Siri [2]. Since such state-of-the-art systems do not yet support web browsing, now seems like a good time to revisit the software accessibility challenge.

To address this challenge, this paper describes the Voice Controlled Accessibility and Testing tool (VCAT), which accepts user voice commands to navigate complex web sites on a Google Chrome browser. The system works as a plugin, logs all user actions, and transcribes them as Selenium test cases. In an initial evaluation, we found that by generating test cases from logged user voice commands, VCAT can significantly reduce the time a traditional expert user needs to create Selenium test cases. This bodes well for the target scenario in which a user that requires accessible software browses the web with Chrome and issues voice commands, enabling such a user to perform parts of routine software engineering work.

To summarize, this work makes the following contributions.

- The VCAT scheme generates test cases of web-based applications entirely via voice commands.
- In an initial evaluation, VCAT reduced the time required to create Selenium test cases.

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195099

2 BACKGROUND AND RELATED WORK

VoiceXML [5] is a W3C standard for specifying voice-based human computer interaction. It is used for creating voice response applications, such as commercial interactive voice response (IVR) systems. But VoiceXML does not address legacy software.

The current Web Content Accessibility Guidelines (WCAG 2.0) [7] and WAI-ARIA [3] provide guidelines and standards for making web applications more accessible. By adhering to these standards, an application will be more easily interpreted by accessible technologies such as screen readers and voice over tools. But there are currently no standards or tools for navigating web sites via voice input. Users still depend on traditional keyboard, mouse, or touch-screen inputs.

3 OVERVIEW AND DESIGN

Figure 1 gives an overview of our approach. At a high level, VCAT takes user input via a microphone and uses a Chrome browser plugin to convert the voice input to text commands, interpret the commands, perform the intended actions in the browser, and create corresponding Selenium test cases.

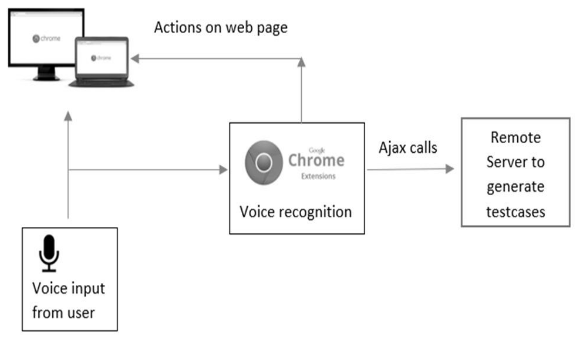

Figure 1: High level system architecture.

To use this scheme, the user must have a stock Chrome browser with the VCAT plugin. (VCAT follows the standard Chrome plugin model, so installing VCAT is straightforward.) By default the VCAT plugin goes into a waiting state, listening for voice commands.

The current VCAT prototype expects that the user gives voice commands in a given predefined grammar. Supporting more general voice recognition schemes is part of future work.

VCAT supports a variety of browser actions and several alternative voice inputs to invoke a given command. Example supported actions include scrolling, minimize, maximize, mouse clicks on links, buttons, images, and navigation menus, selecting options from lists, and textual input into text fields.

4 INITIAL EVALUATION

To evaluate VCAT, we are currently exploring the following two preliminary research questions. (RQ1) How long does it take a traditional expert user to perform web application tasks without VCAT (via keyboard and mouse) and with VCAT (only via voice)? (RQ2) How long does it take a traditional expert user to create a Selenium test case for a web application without VCAT (by coding it via keyboard and mouse) and with VCAT (by VCAT transcribing the voice commands it received in RQ1)? Since a user typically first performs a task in the web application (RQ1) and then codifies it in a test case (RQ2), the overall runtime of performing these two steps in sequence is maybe the most relevant performance metric.

For this initial evaluation, we used basic tasks such as going to the Google web site, inputting the search term "Java", and submitting the search query. Another task was logging into a web based email application. Future work includes both more web application tasks and an evaluation with users from the target audience.

In the experiment we used a Core i7 2.7GHz machine with 12GB RAM running 64bit Windows 10, Chrome v63, and JDK 1.8. The VCAT component that transcribes voice commands to Selenium test cases is a Java component running locally in a Tomcat server. Neither the Chrome plugin nor the setup with Ajax calls to the text-to-test component are optimized for execution performance.

Task	Perf [min]		Create T [min]		Total [min]	
	W/o	VCAT	W/o	VCAT	W/o	VCAT
Email login	1:00	1:45	2:30	0:30	3:30	3:15
Google search	1:00	1:30	2:00	0:30	3:00	2:00

Table 1: Initial VCAT evaluation, where an expert user performed a task with Chrome and then created a corresponding Selenium test case, both with and without VCAT. All times are rounded to the closest quarter minute.

Table 1 summarizes the results. To perform a given application task, a VCAT user took longer than a traditional input user. This is not surprising, as VCAT has to process voice commands, which is more computationally expensive than processing keyboard and mouse events. But the increase in runtime was relatively moderate.

To create a test case, VCAT can significantly reduce the runtime of a traditional approach. This is maybe also not surprising, as VCAT uses the voice recognition results from the earlier application task. VCAT generates the resulting test case, freeing the user from typing. This faster test generation also yields VCAT's shorter total runtime.

5 CONCLUSIONS

To address the accessibility shortcoming of current applications, this work leveraged recent speech recognition advances to create a browser plugin that interprets voice inputs as web browser commands and as steps in a corresponding test case. In an initial experiment, the resulting Voice Controlled Accessibility and Testing tool (VCAT) prototype for Chrome and Selenium yielded a lower overall runtime than a traditional test creation approach.

REFERENCES

[1] Amazon 2018. *Alexa Skills Kit.* Amazon. https://developer.amazon.com/docs/ask-overviews/build-skills-with-the-alexa-skills-kit.html.
[2] Apple 2017. *SiriKit.* Apple. https://developer.apple.com/documentation/sirikit.
[3] Steve Faulkner. 2018. *ARIA in HTML.* W3C Working Draft. W3C. https://www.w3.org/TR/html-aria/.
[4] Microsoft 2017. *Cortana Skills Kit.* Microsoft. https://docs.microsoft.com/en-us/cortana/skills/overview.
[5] Matt Oshry et al. 2007. *Voice Extensible Markup Language (VoiceXML) 2.1.* W3C Recommendation. W3C. http://www.w3.org/TR/2007/REC-voicexml21-20070619/.
[6] Selenium 2012. *Selenium for Java.* Selenium.
[7] Gregg Vanderheiden, Loretta Guarino Reid, Ben Caldwell, and Michael Cooper. 2008. *Web Content Accessibility Guidelines (WCAG) 2.0.* W3C Recommendation. W3C. http://www.w3.org/TR/2008/REC-WCAG20-20081211/.