# An Intuitive Accessible Web Automation User Interface

Yury Puzis Yevgen Borodin Faisal Ahmed I.V. Ramakrishnan

Stony Brook University, Computer Science Department, Stony Brook, NY 11790-4400
{ypuzis, borodin, faiahmed, ram}@cs.stonybrook.edu

## ABSTRACT
In recent years, the Web has become an ever more sophisticated and irreplaceable tool in our daily lives. While the visual Web has advanced at a rapid pace, assistive technology has not been able to keep up, increasingly putting visually impaired users at a disadvantage. Web automation has the potential to bridge the accessibility divide between the ways blind and sighted people access the Web; specifically, it can enable blind people to accomplish web browsing tasks that were previously slow, hard, or even impossible to achieve. In this paper, we propose and evaluate an intuitive and accessible web automation interface. We validate the design in a Wizard-of-Oz user study with visually-impaired subjects and show that the proposed approach has the potential to significantly increase accessibility and usability of web pages, reduce interaction time, and increase user satisfaction. Our findings demonstrate the feasibility of and emphasize the pressing need for truly accessible web automation technologies.

## Categories and Subject Descriptors
H5.2. Information Interfaces and Presentation: User Interfaces; H5.4. Information Interfaces and Presentation: Hypertext/Hypermedia – navigation.

## General Terms
Human Factors, Experimentation, Design

## Keywords
Web Accessibility, Blind Users, Web Browser, Screen Reader, Macro Recorder, Macro Player, Non-Visual, Audio Interface, Web automation, form filling, Wizard-of-Oz

## 1 INTRODUCTION
Blind people typically access computers with the help of screen readers [1-4], the assistive technology software that narrates screen content and enables navigation over that content with shortcuts or gestures. While screen readers have made text-based web content accessible, they do not adequately support modern web interfaces. Web developers can make their web applications accessible; however, *few* web developers follow accessibility guidelines to the point [5]. But even on the most "accessible" web applications, screen reading is slow and inefficient [6].


W4A2012 - Communication, April 16-17, 2011, Lyon, France. Co-Located with the 21st International World Wide Web Conference.
 ISBN: 978-1-4503-1019-2

The inability of blind users to determine which content can be skipped before listening to it often results in high cognitive load, as a consequence of which, important parts of web page content and actionable objects (such as forms, "add to cart" buttons, etc.) become difficult (and sometimes impossible) to discover and access [7]. This problem is further exacerbated while performing online transactions that span several web pages, such as those for shopping and paying bills [8]. One way to overcome these problems is to employ *web automation* – a technology that reduces the human effort necessary to perform typical web browsing activities such as form filling and clicking links, by performing these actions on behalf of the user.

A number of attempts to popularize web automation have been made throughout the years, both as part of screen-readers and as browser extensions [1, 2, 9]. With a typical web automation tool, the user records a macro (a useful sequence of browsing actions), and later retrieves and replays it to automate the same browsing task. Although in our interviews blind people were quick to come up with scenarios where they could use automation, they admitted to hardly using it at all, because most automation tools have not been designed to be used by blind people and they require explicit creation, management, and reuse of macros. Hence, we see a pressing need for a more intuitive web automation interface that requires no effort on the part of the user.

Creating an intuitively accessible web automation technology calls for a radical redesign, that focuses on the target users and use cases before implementing the underlying algorithms. In this paper, we propose and evaluate a new design of a web automation interface, whose main characteristic is the focus on accessibility and usability. By significantly simplifying the user interface and removing the need to explicitly record macros and then replay them, we are able to address the above-mentioned shortcomings of the existing web automation approaches.

While the proposed web automation interface can benefit both people with and without vision impairments, in this paper, we focus on screen-reader users, who can benefit from accessible web automation the most. The specific contributions of the paper are: (a) reviewing the current state of the art in accessible, as well as regular web automation technology (b) proposing an intuitive web automation interface that eliminates the need for explicitly recording/replaying macros (c) validating the usability and effectiveness of the interface in two realistic use scenarios.

## 2 BACKGROUND
### 2.1 Approaches to Web Automation
#### 2.1.1 *Approaches to Recording a Macro*
Most web automation is done by recording *macros* – sequences of machine-readable commands that automate web browsing actions. The two prevalent approaches to creating a macro include

*handcrafting* a macro script to perform a sequence of actions [11] and recording macros with programming-by-example (*PBE)* [10].

The early (and still used) approaches to automation required handcrafting a script that described the sequences of actions to be performed. This could be accomplished by a software developer who integrated automation into a specific feature (such as JAWS screen reader [1] ResearchIt feature that allows to look up in a dictionary) or by a power-user who wrote out the necessary sequence of steps using some scripting language. Browsers and screen readers still follow this approach, e.g., JAWS [1] and Windows Eyes screen readers [12] enable users to write their own scripts for task automation and share the scripts with other users.

A more advanced PBE approach enables the user to record a sequence of steps, thus, eliminating the need of handcrafting a macro. This is the approach taken by the majority of modern automation tools [11, 13, 14]. To take this further, a macro recorded for one web site can be dynamically adapted to be used for a similar task on a different web site [14].

### 2.1.2 *Approaches to Replaying a Macro*

There are also different ways of replaying a macro [15]: it can be initiated *on-demand*, *e.g*., by using a hotkey or a voice command; it can be initiated by some *system event trigger*, e.g., page-load; it can follow a *user-specified schedule;* an *automated* (i.e. semi-automatic) approach involves using a system event trigger or scheduler to prompt the *user* to initiate a macro; and, finally, a macro can be initiated from another macro as a *subroutine*.

## 2.2 Web Automation Tools

### 2.2.1 *Web Automation Tools for Sighted People*

WebVCR [16], WebMacros [16], iMacros [17], and Robofox [18] systems all use the PBE approach to automation, in which users have to initiate / terminate recording of macros manually.

Robofox [18] makes use of automatically generated assertions to help detect and correct macro failures. Using the Smart Bookmarks [19] tool, the user needs to indicate only the end of the action-sequence recording. Once the recording is terminated, the system determines the starting position retroactively.

CoScripter [11] enables both programming by example and manual collaborative (using web wiki) creation of new and editing existing macros. CoScripter Reusable History (also known as ActionShot) [20] is a tool that records every user action (without the need to initiate/terminate recording) and allows the user to modify the list of actions using the same collaborative interface. ActionShot can convert user actions into pseudo-natural language strings for easier understanding, and relate each action to a screenshot of the webpage. However, ActionShot requires that the user manually indicate the actions to be turned into a macro.

CoCo [21] is a tool that provides an abstraction layer on top of a browser by leveraging CoScripter and ActionShot. CoCo takes a natural-language query, maps the query to a macro, executes the macro (possibly asking the user to evaluate parameters), and extracts a part of a webpage as a response to the user. The macros are retrieved both from CoScripter's collaborative database, and from user's web history logs continuously recorded by ActionShot. The history log is automatically split into sets of consecutive steps (macros) using a few simple rules such as URL change and time elapsed between steps.

Creo [15] is another PBE tool that uses content of a visited webpage to automatically suggest that the user initiate one (or more) of the prerecorded macros. Creo uses knowledge-bases to map webpage content to specific macros by generalizing words used on the webpage and in the macro. For example, a macro may contain a search query for a "cat", which can be generalized to an "animal". Creo will suggest the user to run that same macro when s/he visits a webpage with any other word that can be generalized to an "animal" (such as a "dog").

All of these tools support on-demand execution of macros, while Creo is also automated with event triggers. CoScripter and Robofox support event-based execution. iMacros, CoScripter, and Robofox also support scheduled execution. Unfortunately, none of these automation tools were designed for blind users. And, while most of these tools can theoretically be accessed with the help of a screen reader, they are far from being really usable by screen-reader users.

### 2.2.2 *Web Automation Tools for Blind People*

Most of the popular screen-readers, including JAWS [1], NVDA [3], Supernova [22], and WindowEyes [12], follow the handcrafting approach and enable users to automate tasks by writing scripts; however handcrafting the scripts requires more skill, training, and time than programming-by-example. One notable exception is ResearchIt implemented in the JAWS screen-reader, a tool that automates lookup of information, e.g., dictionary definitions for a particular word. ResearchIt can be accessed via hotkeys, and also presents a scripting API that can be used to add additional information sources.

A Macro feature [13] in the Hearsay non-visual web browser [9], to the best of our knowledge, was the first PBE web automation tool designed specifically for blind screen-reader users. The prototype had a speech-enabled interface for recording and replaying parameterized macros; however, it stayed a scientific prototype.

Another research prototype, Trailblazer [14], is a CoScripter-based tool that interfaces with JAWS screen-reader and provides access to the CoScripter's user-shared macro database. Trailblazer is able to adapt a macro recorded for one website and reuse it to accomplish the same task on a different website. Trailblazer's user interface is integrated into the webpage, allowing the blind user to read the pseudo-natural language description of each action in the macro. While accessible with a screen reader, the Trailblazer interface was not truly usable because it was based on the CoScripter tool, which was designed for sighted people.

# 3 WEB AUTOMATION INTERFACE

Having learned from the shortcomings of the existing automation solutions, we describe the key concepts of an intuitive and accessible interface, following the guidelines proposed in [23].

Our main observation is that *very few actions are applicable in a given browsing state / context, and even fewer are reasonable / useful, i.e. can lead to a meaningful / desirable result.* Armed with the ability to record user states, an intuitive and accessible automation interface can then be based on the following approach:

1. While using a browser and (optionally) a screen-reader, the user can modify the state with user input or utilize predefined keyboard shortcuts or speech commands to review (listen or read through) a pseudo-natural language description of a suggested set of actions relevant in a particular browsing state; such set should be minimal (preferably a single action).

2. The user can then utilize a keyboard or a speech command to select and confirm execution of a *single action* from that set.

The set of relevant actions can be inferred with the help of a predictive model, the development of which is not in the scope of this paper. Such model can be trained on the history of user's actions in the current and preceding states, as well as on the history of user's actions in any state.

In the proposed interface, there is no need to know if a macro is available for a given webpage, how to find it, if it will work in the current system state, or if the macro has finished replaying and, if it did, what actually happened. There is no need for progress feedback, and the possibility that one or more steps will fail without the user realizing it is significantly diminished. The ability of the model to automatically suggest relevant action eliminates the need to know when to start/stop a recording if a macro recorded correctly and if it is sufficiently general. Focusing on one action at a time should help focus the model on generating most relevant suggestions. Overall, using the model would help to minimize the automation interface.

The proposed automation interface would be ready for use at any time (at any browsing state), but it can also be ignored completely or used interchangeably with regular browser and screen-reader interactions. If the user is familiar with the action that can be automated in a given state, action review may be skipped. The user may also choose to skip action confirmation/activation and continue interacting with the browser and the screen-reader. Since this interaction may change the state, the next time the user tries to review or automate an action s/he may be presented with different options. The automation interface does not require introduction of a special mode of interaction and strives to be highly controllable yet minimal.

Execution of an action can optionally trigger voicing of the new state, e.g., voice: "textbox name John Doe", if an action filled the "Name" textbox with value "John Doe". This will help the user understand what just happened and provide a semi-automatic mechanism for verifying that the executed action had the desired effect. The ability to seamlessly switch between the automation interface and the browser will also allow the user to manually review results of recently automated actions and, if necessary, immediately correct the resulting mistakes.

# 4 INTERFACE EVALUATION

## 4.1 Participants and Experimental Setup

Disability Resource Center of the Arizona State University helped us recruit 17 blind subjects for our user study. Gender representation was approximately equal. The ages of participants were evenly distributed and varied from early twenties to late fifties. All subjects were well-versed in the use of screen readers (15 participants used JAWS as their primary screen reader, 1 – System Access, 1 – VoiceOver).

To conduct a Wizard-of-Oz study, we created Automated Assistant (AA): the automation interface and a simulated (hardcoded) model that allows it to propose useful actions on the websites used in the experiment. By hardcoding the model we ensured that the flaws of a specific model would not influence the evaluation of the interface. The AA was integrated with the HearSay Non-visual web browser [9] that provided the commonly used features and shortcuts of the JAWS screen reader and integrated with the Firefox browser. We used the IVONA [24] Text-To-Speech engine with the voice "Eric" and speech rate of 180 words per minute.

Prior to the beginning of the experiment, the subjects were explained how to use the AA. Since all the subjects had experience with using JAWS, our screen-reading interface did not need to be explained and the subjects could use it to browse web pages as efficiently. To measure the effect of learning, the subjects were *not* given the opportunity to practice with the AA.

During the experiment, the subjects were asked to go through two scenarios: compete the checkout process for purchasing an audiobook at http://www.audible.com (scenario 1) and reserve a hotel room at www1.hilton.com (scenario 2). The scenarios were chosen as some of the most representative use cases according to our accessibility consultants. Each scenario was performed 3 times on the same websites (subtasks #1-3), but each time the task was slightly different (buy a different book/reserve a different room). In subtasks 1-2 the subjects were asked to perform the task as they would normally do using a standard screen-reading interface. In subtask 3 the AA was turned on and the subjects were asked to perform the same task once again.

The reason for having two consecutive similar subtasks in each scenario was to measure the learning effect of repeating a similar task for the second time. By the second subtask, the subjects were already familiar with the website and completed the subtask 2 much faster. Comparing subjects' performance on subtask 1 against subtask 3 would have been invalid, skewing the results of the experiment. To demonstrate that the prior experiences of our subjects did not have a statistically significant effect on our measurements, we performed the following tests.

We did not randomize the subtasks with and without the AA, because while using it, the subjects were prompted with every step of the transaction and did not need to navigate on the webpage. So, the knowledge of which steps to be performed would have no effect on the completion time.

## 4.2 Hypotheses and Results

In this section, we present the results of our experiments. Overall, subjects were able to complete the tasks significantly faster with the AA compared to the regular screen-reader interface. During the experiment, we measured the time it took the subjects to complete each task. Following each task, the participants were asked to rate the difficulty of each subtask on a 5-point Likert scale (1=Very Easy to 5=Very Hard).

In Scenario 1, the average interaction difficulty was 3.12, 2.29, 1.65 and the timing was 440 sec., 242 sec., 120 sec. for subtasks #1, #2, and #3 respectively. In scenario 2 the average interaction difficulty was 3.69, 2.75, and 1.56 and average interaction time was 561 sec., 301 sec., 154 sec., for subtasks #1, #2, and #3 respectively. In all cases the improvement between subtasks was statistically significant, with confidence level 0.01.

*Hypothesis 1: Doing a transaction w/AA reduces the completion time compared to not using AA.* In both scenarios the difference between subtasks 2 and 3 was shown to be statistically significant using one tailed t-test (conf. level < 0.0001).

*Hypothesis 2: Doing a transaction w/AA is easier compared to doing it w/o AA.* In both scenarios the difference between the averages in subtasks 2 and 3 was shown to be statistically significant using one tailed t-test (confidence level < 0.005).

In the end of the experiment, the participants were asked two sets of questions. The 1st set was about the experience with the AA (rated on a 5-point Likert scale value with 1=Never 2=Sometimes 3=Regularly 4=Often 5=Always). The 2nd set was about the general experience with online transactions (rated on a 5-point Likert scale value with 1=Strongly Disagree to 5=Strongly Agree). The goal of the questions was to justify the need for web automation and evaluate the effectiveness of the AA in fulfilling that need. The term "online transaction" was defined "as set of browsing steps required to go through a sequence of pages while shopping, banking, etc."

The 1st set of questions (average/std. deviation) is: 1. "I often have to fill out forms that I never filled out before" (3.29/0.92); 2. "I often spend a lot of time figuring out what needs to be done to continue to the next step of an online transaction" (3.29/0.99); 3. "I often spend a lot of time looking for specific links, buttons, or other form elements in a transaction" (3.29/1.05); 4. "I often make mistakes while doing transactions that I have done on the same websites in the past" (1.94/0.75).

The 2nd set of questions (average/std. deviation) is: 1. "I wish I could do online transactions faster than I can with a regular screen reader" (3.88/1.11); 2. "I often experience difficulties while doing online transactions with a screen reader" (2.62/1.02); 3. "Doing the same online transaction for the second time was easier than the first time" (4.59/0.71); 4. "Doing a transaction with the Automatic Assistant was the easiest" (4.12/1.27); 5. "I want to use Automatic Assistant in the future" (4.29/1.10).

## 4.3 Discussion

The confirmation of both hypotheses demonstrated that AA was able to reduce both the perceived difficulty and the actual time taken to complete each task. As can be seen from the post-completion questionnaire results the responses support the hypotheses: the participants agreed that using AA was easier than using the screen-reader interface. Most subjects either agreed or strongly agreed that they wanted to use the AA in the future.

The majority of subjects were comfortable with using AA and demonstrated high confidence when using AA, a very encouraging sign given that they have never had a chance to interact with it before.

Three of the users, after already having started doing a task with the AA, performed one or two steps manually, and then continued doing the task with the help of the AA. The fact that even inexperienced users were able to gracefully recover from confusing situations shows that the AA interface is robust and discoverable.

# 5 FUTURE WORK

In this paper, we showed that our approach to web automation could be very useful for both experienced and novice users, making repetitive web browsing tasks faster and easier to do. Some of the future work opportunities that we see include designing the model that will be capable of storing and predicting steps that will be useful for the user and using crowdsourcing / Social Accessibility [25-26] approach to maintain a database of user actions, and use that database to automate user interaction.

# 6 ACKNOWLEDGEMENTS

The authors would like to thank Terri Hedgpeth at the Disability Resource Center of Arizona State University. This work has been supported by NSF Awards IIS-0808678 and CNS-0751083.

# 7 REFERENCES

1. JAWS, *Screen reader from Freedom Scientific*, 2011.
2. Geoffray, D., *The internet through the eyes of windows-eyes*, in Proceedings of *CSUN* 1999.
3. NVDA, *Non-Visual Desktop Access*, 2011.
4. VoiceOver, *Screen reader from Apple*, 2010.
5. Lopes, R., et al., *Web not for all: a large scale study of web accessibility*, in *W4A* 2010: Raleigh, North Carolina, p. 1-4.
6. Borodin, Y., et al., *More than meets the eye: a survey of screen-reader browsing strategies*, in *Proceedings of W4A* 2010, ACM: Raleigh, North Carolina, p. 1-10.
7. Islam, M.A., et al., *Improving Accessibility of Transaction-Centric Web Objects*, in *SDM 2010*: Columbus, OH.
8. Mahmud, J., et al., *Automated construction of web accessibility models from transaction click-streams*, in *Proceedings of WWW* 2009, ACM: Madrid, Spain.
9. Borodin, Y., et al., *Hearsay: a new generation context-driven multi-modal assistive web browser*, in *Proceedings of WWW* 2010, ACM: Raleigh, North Carolina, USA.
10. Cypher, A., et al., eds. *Watch what I do: programming by demonstration*. 1993, MIT Press. 652.
11. Leshed, G., et al., *CoScripter: automating & sharing how-to knowledge in the enterprise*, in *Proceeding of SIGCHI* 2008, ACM: Florence, Italy.
12. Window-Eyes, *Screen Reader GW Micro*, 2010.
13. Borodin, Y., *Automation of repetitive web browsing tasks with voice-enabled macros*, in *Proceedings of SIGACCESS* 2008, ACM: Halifax, Nova Scotia, Canada.
14. Bigham, J.P., et al., *Trailblazer: enabling blind users to blaze trails through the web*, in *Proceedings of IUI*2009, ACM: Sanibel Island, Florida, USA.
15. Faaborg, A. et al., *A goal-oriented web browser*, in *Proceedings of the SIGCHI*2006, ACM: Montreal, Quebec, Canada. p. 751-760.
16. Anupam, V., et al., *Automating Web navigation with the WebVCR*, in *Proceedings of WWW* 2000, North-Holland Publishing Co.: Amsterdam, The Netherlands.
17. iMacros. *iOpus*. 2009 cited 2009; Available from: http://www.iopus.com/imacros/firefox/.
18. Paolacci, et al., *Running Experiments on Amazon Mechanical Turk.* Judgment and Decision Making, Vol.5, No.5, 411-419, 2010.
19. Salganik, M.J., P.S. Dodds, and D.J. Watts, *Experimental Study of Inequality and Unpredictability in an Artificial Cultural Market.* Science, 2006. **311**(5762): p. 854-856.
20. Li, I., et al., *Here's what i did: sharing and reusing web activity with ActionShot*, in *Proceedings of SIGCHI* 2010, ACM: Atlanta, Georgia, USA. p. 723-732.
21. Lau, T., et al., *A conversational interface to web automation*, in *Proceedings of UIST 2010*, ACM: New York, USA.
22. SuperNova, *Screen reader from Dolphin*, 2010.
23. Puzis, Y., et al., *Guidelines for an accessible web automation interface*, in *Proceedings of SIGACCESS* 2011, ACM: Dundee, Scotland, UK. p. 249-250.
24. IVONA. *IVONA multi-lingual speech synthesis system*. 2005; Available from: http://www.ivona.com/.
25. Kawanaka, S., et al., *Accessibility commons: a metadata infrastructure for web accessibility*, in *Proceedings of ASSETS* 2008, Halifax, Nova Scotia, Canada. p. 153-160.
26. Kawanaka, S., et al., *Accessibility commons: a metadata repository for web accessibility.* SIGWEB Newsl., 2009(Summer): p. 1-7.