Here's What I Did:
Sharing and Reusing Web Activity with ActionShot

Ian Li

Human Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15217
ianli@cmu.edu

**Jeffrey Nichols, Tessa Lau,
Clemens Drews, Allen Cypher**

IBM Research – Almaden
650 Harry Road
San Jose, CA 95120
{jwnichols,tessalau,cdrews,acypher}@us.ibm.com

ABSTRACT

ActionShot is an integrated web browser tool that creates a fine-grained history of users' browsing activities by continually recording their browsing actions at the level of interactions, such as button clicks and entries into form fields. ActionShot provides interfaces to facilitate browsing and searching through this history, sharing portions of the history through established social networking tools such as Facebook, and creating scripts that can be used to repeat previous interactions at a later time. ActionShot can also create short textual summaries for sequences of interactions. In this paper, we describe the ActionShot and our initial explorations of the tool through field deployments within our organization and a lab study. Overall, we found that ActionShot's history features provide value beyond typical browser history interfaces.

Author Keywords: ActionShot, CoScripter, web browser history, reuse, sharing, social networking

ACM Classification: H.5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Algorithms, Human Factors

INTRODUCTION

People's actions are recorded every time they browse the web, but the page-based history that browsers store typically contains only page titles and URLs. Users rarely find this browser history useful [5], and part of the reason may be that typical browser histories are not sufficient for describing all the actions that a person does while browsing. For example, typical browser histories do not include the values that are entered into forms, nor do they document interactions with complex AJAX-centric web sites.

Users might find browser histories more useful if they could easily reuse portions of their previous interactions or share relevant details about their browsing history with others. Web sites such as del.icio.us and Magnolia allow

users to share bookmarks; Digg and Reddit allow users to share interesting web pages that they found. However, these web sites only allow people to share the URLs of individual pages. If people want to share what they did on a web site, they have to write it down manually, which can be so tedious that they forego sharing the information. Social scripting services such as CoScripter [8] allow users to record and share interactions with websites, but these tools require forethought and planning to enable recording at the right time to capture a reusable script. Moreover, CoScripter's one-to-all sharing model was found to deter many users [8], who asked for finer grained control over with whom they shared their scripts. An enhanced browser history could solve these problems, letting users easily grab sequences and share them with the desired audience.

In order to explore these ideas, we created *ActionShot*, an extension to the Firefox web browser built on top of the CoScripter web recording/playback platform [8]. ActionShot records web browsing history at the level of interactions, such as entering a value into a form field, turning on a checkbox, or clicking a button. This goes beyond typical web history interfaces and gives users a more complete picture of the actions they performed on every web page they visited. ActionShot provides an interface to this history data that allows for easy browsing and searching, where each step is described as a pseudo-natural language string that is easy for users to interpret and accompanied by a screenshot that allows users to see exactly how each step was performed in the context of the page.

While we believe that users will find many uses for their improved history data, our focus has been on two specific uses: reuse and sharing. ActionShot's history data can be reused through the re-execution of recorded steps at a later time, either by creating a script from a set of actions or executing steps individually from the ActionShot interface. Sharing is supported in the following ways:

- Posting an action sequence to Facebook

- Posting a summary of an action sequence to Twitter

- Sending a sequence of actions via email

- Copy-and-pasting a sequence as text

- Converting a sequence of actions into a CoScripter script and sharing that script on the CoScripter wiki
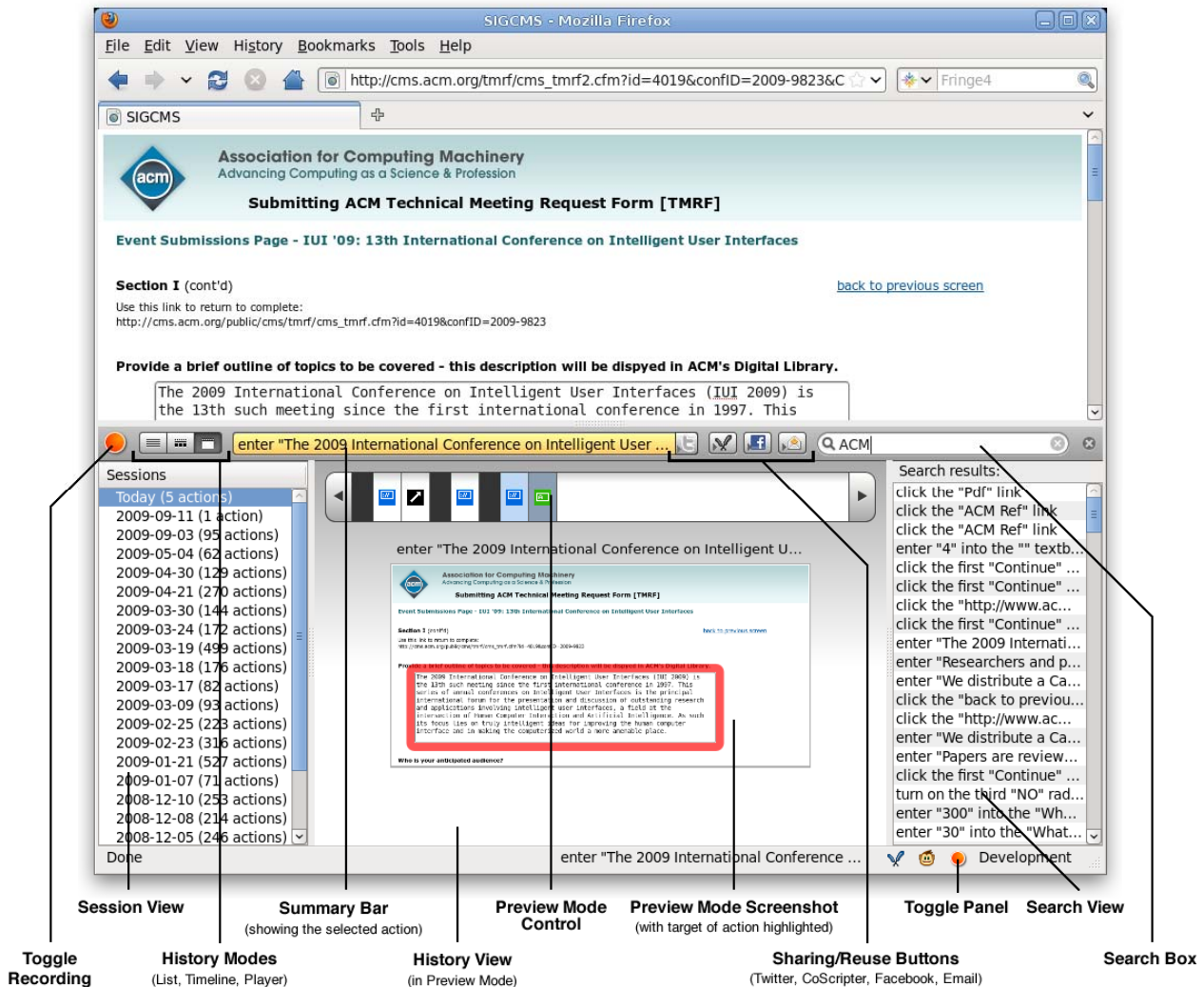
Figure 1. The ActionShot interface embedded in a Firefox browser window.

In a think-aloud lab study and two field deployments, we saw that users understood these sharing features and found them to be useful in certain situations. We also conducted a lab study to examine whether users were effective at extracting reusable sequences of actions using the ActionShot user interface.

We begin by putting ActionShot in context with past research in this area. We then describe a scenario to illustrate how ActionShot is used and the implementation of Action-Shot's recording and sharing features. Next, we discuss the evaluations of ActionShot that we have conducted, including the field deployments inside our organization and the lab study. We conclude with a brief description of some possible directions for future work.

RELATED WORK
We have organized the related work into two categories: searching and exploration of browsing history, and generating scripts for web tasks.

Search and Exploration of Browsing History
Most web browsers keep a record of browsed pages that users can search and explore, including the very first graphical web browser, Mosaic [1]. However, search is limited because the history usually only consists of URLs and document titles. Safari 4 contains a new history feature that records both the HTML content and screenshots of pages, which improves search and graphical exploration of the history. Google Web History offers a similar feature with additional trending information and improved search, but requires the installation of the Google Toolbar. MIT's eyebrowse[1] system also tracks the web pages you visit and shares them as a feed with other users. More recently, a browsing history implementation called the Contextual Web History [5] has improved search of browsing history by using additional metadata, such as time of visit, visual appearance, and page text. ActionShot adds the capability

---

to record what users did within browsed pages, which allows users to search by actions (*e.g.*, find the form where I entered my address, or find what I did *after* I logged in to southwest.com).

Some studies have shown that providing screenshots of web browsing improves recall of visited web sites [6, 11]. ActionShot's visual history goes beyond these systems by also providing details of the actions performed on the page. In this respect, ActionShot is similar to the Tableau system [3] and Nakamura and Igarashi's visualizations of user operation history [9]. These systems do not show visualizations for web browsing activity however, but instead for a visual database and a graphical editor, respectively.

ActionShot is related to the macro-by-example system [7] created for Chimera, a graphical editor for 2D illustrations, UIs, and text. The system allows users to create macros by selecting sequences of actions from the operation history of the editor. Chimera did not include support for sharing macros however, nor did it include the range of visualizations present in ActionShot.

Scripting the Web
The ActionShot system is built on top of the CoScripter platform [8]. CoScripter allows users to generate human-readable commands by demonstrating a sequence of web-browsing actions. Unlike CoScripter, ActionShot does not require the user to explicitly record a script. Instead, users' browsing actions are continually recorded on the user's local disk for later retrieval and exploration.

WebVCR [2] and WebMacros [10] are systems that also record web browser actions and require users to explicitly indicate when a recording should begin. Unlike ActionShot and CoScripter, both systems have internal representations of the recorded actions that are not conducive to manual editing or sharing with others.

Smart Bookmarks [4] gives users the ability to "bookmark" dynamically-generated web pages by searching backwards though recent history and identifying the actions needed to navigate to the current page. While Smart Bookmarks lets users save or share actions from the current browsing session, ActionShot lets users share any actions they have ever performed by providing a visual interface for browsing and searching a user's entire history.

SCENARIO
In this section, we present a scenario to illustrate how ActionShot can be used to facilitate finding and sharing of sequences of actions from web browsing history.

Alice has just finished serving as conference treasurer for an annual ACM conference (PoCS). One of the duties of the conference treasurer is to prepare a conference budget, called a TMRF, and submit it to the ACM for approval. This process involves submitting a web form that requests many pages of information, including details about the conference (number of attendees, location, description) and details about the paper selection process (review criteria,

number of accepted papers, maximum paper length). Preparing this information required Alice to gather data from multiple websites and coordinate email amongst several conference organizers over several days. Once ACM had approved the submitted form, Alice was done with her duties and, as she would not be serving as treasurer again, quickly forgot many of the details of the process.

Soon after the conference was held, however, Alice received an email from the next conference treasurer, who asked for guidance on the ACM budget submission process. Although Alice could have pointed the incoming treasurer to the URL for the budget process, she wanted to be more helpful. Most of the information needed to complete the process did not change significantly from year to year. If she could pass along the information she had used to fill out the form the previous year, she knew it could be a significant time-saver. Unfortunately she had not had the foresight to explicitly record that information because she did not realize it would be useful in the future. Luckily, Alice had been using ActionShot.

In order to recover her actions on the ACM web form, Alice opens the ActionShot panel (Figure 1) by clicking on the orange ActionShot icon in the browser's status bar. The panel appears horizontally at the bottom of the browser. On the left side is the Session View where Alice can see her browsing sessions organized by date. By default, the current session is selected.

Search and Exploration
Alice starts by using ActionShot's search feature to find her previous interactions involving the TMRF form. In the Search Box on the toolbar, she enters "TMRF". The Search View appears and shows multiple results, including:

1. go to "http://acm.org/tmrf/"

2. enter "PoCS 2008" into the "Conference name:" textbox

Alice clicks on the second result and the corresponding action is highlighted in the History View, which is in List Mode by default (Figure 2a). Alice scans through the detailed textual description of the actions (i.e., the description of the action, the name of the page, and the URL). To visually check that the sequence shown is correct, she can switch the History View to the Timeline Mode (Figure 2b), which displays thumbnail screenshots of the ACM web site. Based on these screenshots, Alice identifies these actions as the sequence she was looking for. Alice can also switch to Preview Mode (Figure 1) to see more detailed screenshots of the actions, in order to further verify that this is the correct sequence.

Reuse
Alice concludes that she has found the correct sequence of actions for submitting a conference budget. In the History View, she selects the start and end of the sequence. She then clicks on the CoScripter icon in the toolbar to convert the sequence into an executable script. The ActionShot panel disappears and the CoScripter sidebar appears on the

a) List Mode

	Command	Time	Page Title	Page URL
	go to "iuiconf.org"	03:23 ...		iuiconf.org
	click the "Conference Overview" link	03:24 ...	2009 International Conferenc...	http://iuiconf.org/
	enter "The 2009 International Conferenc...	03:24 ...	SIGCMS	http://campus.acm.org...
	enter "Researchers and practitioners int...	03:24 ...	SIGCMS	http://campus.acm.org...
	click the "Call for Papers" link	03:24 ...	2009 International Confer...	http://iuiconf.org/overv...
	enter "We distribute a Call For Papers " ...	03:24 ...	SIGCMS	http://campus.acm.org...
	click the "back to previous screen" link	03:24 ...	SIGCMS	http://campus.acm.org...
	click the "http://www.acm.org/sigs/volu...	03:25 ...	SIGCMS	http://campus.acm.org...

b) Timeline Mode

Figure 2. The other two modes of the History View.

left side of the browser. Alice can use the "Step" and "Run" buttons in CoScripter to replay the script and verify that the steps she has selected are reusable.

Sharing

After verifying that she found the correct sequence of steps that perform the desired function, Alice goes back to ActionShot and presses the email button, which begins composing a new email message with the text of the steps she has selected as the body of the email. Because these steps are textual and similar to human-written instructions, the new treasurer does not need to have ActionShot or CoScripter installed to make use of the instructions. Alice sends these steps to the new treasurer, who is very appreciative of Alice's help. Her use of ActionShot has saved the new treasurer from hours of tedious labor, and moreover begins to define best practices for future conferences.

IMPLEMENTATION

This section describes the major aspects of the implementation of ActionShot. First, we explain how the system records a detailed history of the user's browsing actions. Then we describe how the user interface facilitates search and exploration of the history. Finally, we describe how reuse and sharing are supported.

Logging web browsing actions

ActionShot records the actions users perform on web pages by using DOM level event handlers such as 'onclick', 'onchange', and other browser-level event handlers. Specifically, ActionShot records high-level user actions such as clicks on links, checkboxes, list boxes and buttons, as well as text entry into form elements. It also listens to navigational events such as entry of a new URL into the location bar, or using the forward and backward buttons in the browser toolbar to navigate through pages. For each action, ActionShot relies on heuristics to extract a unique human readable label for the target of the action. It then combines the action, target type and target label to generate an English description of the action that was just performed. For

Usage	Data stored
a) History View, Search, Reuse, and Sharing	Description of the action
	Document title
	Document URL
b) Visual display in History View	Document: screenshot and HTML
	Target: screenshot and HTML
c) Session management in History View	Action Timestamp
	Browser Timestamp
	Browser ID
	Logger ID

Table 1. The data stored by ActionShot for each action and what they are used for.

example a 'click' event that occurred on an html '<a>' target that contains the text 'home' will be recorded as "Click the 'home' link". For each recorded action, ActionShot stores several types of information (Table 1).

All of the non-image data, such as the strings entered into text fields and XPath references to the targets of any events, are stored in a sqlite database on the user's hard drive. The use of a sqlite database allows quick access to the potentially large amount of data that may be collected by ActionShot and also obfuscates the data on the hard drive, making it harder to find with simple text searches.

Search

Unlike regular web search, which retrieves complete web pages in response to a keyword query, ActionShot search retrieves matching *actions*, and displays them in the context of the action sequences in which they were performed.

Search is performed over a user's entire recorded history. Search terms are matched with the title of the page, the URL, and the description of the action. Since the description of the action references what the user actually did on the page, the search is more focused and precise than searching over page contents. For example, searching a conventional web history for the term "PoCS" in the budget scenario above would not have returned the budget submission form, because "PoCS" did not appear on the web page itself. However, searching over ActionShot's history would return the form, along with the actions taken on the form, because ActionShot's search covers not just page content but also the action descriptions.

Results of a search appear in reverse chronological order in the Search View on the right side of the ActionShot panel (Figure 1). When a user selects a result, the History View shows the action in the context of the sequence. This is important to give the user an idea of how the action relates to the rest of the sequence.

History Views

A critical part of ActionShot is allowing users to select sequences of actions in their detailed history. We created

three different view modes, List, Timeline, and Preview, to support different means of exploration. The user can switch between these modes by using the History Mode buttons on the ActionShot toolbar.

The List Mode of the History View (Figure 2a) allows users to look at the details of their history as textual descriptions. This mode is similar to the Detail View of Windows Explorer in Windows and the List View of Finder in Mac OS X. This mode shows each action as a row with five columns for a small icon, the description of the action, the timestamp of the action, the page title, and the URL.

The Timeline Mode of the History View (Figure 2b) is a visual way for users to quickly glance at the sequence of actions. This mode shows small thumbnails of the pages that users visited. Next to each thumbnail are small colored icons that describe the type of actions that were performed on the corresponding page. These icons are replicated as small bars above the thumbnails. The following are the four types of actions shown (and their colors):

- Form actions (green). These are actions that are related to web forms, such as check boxes, radio buttons, input fields, and text areas. These actions usually occur within pages and do not cause a page change.

- Link actions (black). These are actions that are related to links and submit buttons. While these actions occur within pages, they usually cause page changes.

- Browser actions (blue). These actions are related to the functions of the browser, such as page reloads, using the back and forward buttons, entering a search term in the browser search box, and entering a URL in the address bar. These actions also cause page changes like the link actions, but they are invoked by controls on the browser toolbar.

- Find actions (red). These are actions related to the user performing a text search on a web page (such as that invoked in most PC browsers by pressing control-F).

The small bars allow users to quickly get an idea of what happened on each page. For example, many pages will only have a black bar meaning the user just followed a link on the page. Pages with web forms will usually have several green bars indicating high activity within the page.

The History View's Preview Mode (Figure 1) accomplishes two things: it allows users to look at each action visually in greater detail than the Timeline Mode and it allows users to follow the sequence of actions in exactly the order that they happened. This mode shows each action one at a time, but with a bigger screenshot of the page. The screenshot also highlights the target of the action with a light red marker. For example, if the user clicked on the "I'm Feeling Lucky" button on the Google home page, the screenshot will have the button highlighted. At the top of the Preview Mode is the Preview Control, which shows a row of icons representing each action and two arrow buttons on the left and right side. Users can click on the icons to jump to an action or click on the arrow buttons to scroll through the sequence.

We thought that users would have difficulties determining where sequences started and ended, so we visually separated sequences of actions using a thick black line. For this implementation, we used a simple heuristic that groups actions by the host name of the site. For example, if a user transitions from a page with the host name google.com to another page with the host name mozilla.com, a black line is drawn between the two actions. We also treated changes to the sub-domain as changing the host, thus a page going from twitter.com to search.twitter.com would be separated by a black line.

All modes allow users to select sequences of actions, which they can reuse and share with others. When multiple actions are selected from the History View, a short descriptive summary of those actions is automatically generated and displayed in the yellow Summary Bar at the top of the ActionShot panel. This summary is used to describe the selected action sequence for reuse and sharing, which are described in the next subsection.

Reusing and Sharing Actions

Once the user has selected a sequence of actions, the user can reuse the action by clicking on the CoScripter icon (🔖) on the ActionShot toolbar. Upon clicking on the icon, ActionShot displays the sequence of actions in the CoScripter panel, which appears on the left sidebar of the browser. On the CoScripter panel, the user can run the sequence to automate the actions. The user can also edit the script to remove extraneous steps, add new steps, introduce control statements such as conditionals and pauses, and ask for user input. When the user is satisfied with the script, the script can optionally be shared on the CoScripter wiki.

ActionShot also has three other ways of sharing history, which are available on the ActionShot toolbar. First, the user can email a sequence of actions to another person by

Figure 3. Example of sequences of actions shared by a user on Facebook.

clicking on the Email icon (⊡). Clicking on the icon triggers the user's default mail client to start composing a new message, with the subject containing the action summary and the body containing the list of selected actions. Second, the user can share sequences of actions on Facebook. Clicking on the icon (⊡) will open up a Facebook page with a form prefilled with the suggested summary and the steps in the action sequence. The user can edit these fields and then share them by clicking on the "Post" button, which causes the sequence of actions to be posted to her Facebook wall (Figure 3). The last method of sharing is sending an update of what the user did on Twitter by clicking the Twitter button (⊡). In this method of sharing, the Twitter web page is opened and the action summary is placed in the "What are you doing?" text box.

AUTOMATIC SUMMARIZATION OF WEB ACTIONS

To facilitate sharing of web activity, we implemented a simple summarization algorithm that picks key words from the selected sequence of browsing actions and generates a short descriptive summary.

The summarization algorithm works as follows. A summary is generated by filling out a template of the form "VERB NOUN on SITE". Given as input the actions selected by the user, the algorithm fills in the slots in this template using several heuristics. The SITE portion of the template is constructed by examining all page titles across the selected actions, tokenizing them into individual words, and then selecting the most common word. This heuristic is based on the observation that most sites tend to put their brand name in page titles (e.g., "tomato diseases - Google Search"). Moreover, this brand identifier will tend to stay consistent   across all pages at the same site (e.g., "tomato diseases - Google News"). Therefore if the browsing history includes multiple pages at the same site, the site's name will tend to occur more frequently.

To fill in the VERB and NOUN portions of the template, we make the observation that the textual description for each action contains words that characterize the activity. Specifically, we focus on the label used to identify the target of each action (e.g., the text of a link, the label of a button, or the caption for a text field). This label often conveys content-rich information about the user's task.

We use a part-of-speech tagger to extract only the two most salient words across all target label words in the selected actions. Using the POS tagger, each label word is classified according to its part of speech. The VERB (NOUN) slot in the summary template is chosen as the verb (noun) that occurs most frequently in the set of target-label words. When multiple words all occur with the same frequency, then one of those most frequent words is selected at random. If no verb is found, the system chooses the default verb "Browse". If no noun is found, that slot is omitted from the template.

Web task	Auto-generated summary
Reserve a conference room using a web-based scheduler	Meeting reservations on RoomWizard
Look up subway schedules for the Washington, DC Metro	Browse travel on Metro
Search for flights between San Jose and Washington, DC	Find flights on Orbitz
Download papers to review from IJCAI website	File view on Twenty
Register for the CHI conference	Continue registration on CHI
Accept an invitation to connect on LinkedIn	Accept profile on LinkedIn
Manage your followers on Twitter	Follow user on Twitter
Get a summary of the past year's 401k activity	Retrieve statement on Fidelity
Buy a discount lift ticket to a local ski resort	Lift ticket on SnowBomb
Find a nearby Indian restaurant on Google Maps	Browse search on Google
Assign papers to reviewers on the GHC 2009 submission website	Suggest assignments on Submission

Table 2. Representative sequences of web browsing actions (left) and their automatically-generated summaries (right).

While simple, this summarization algorithm performs surprisingly well. We have tested it informally on our own ActionShot histories; Table 2 shows examples of some of the summaries generated on some of the activities drawn from our histories. For example, an activity performed by one of the authors was to change a meeting room reservation using a web-based conference room scheduling tool:

- go to "b2126.XXX.XXX.com"
- click the "Reservations" link
- click the "John Smith - MyProject" button
- click the "b_lb_open.gif" button
- enter "Mary Jones" into the "Host:" textbox
- enter "MyProject weekly meeting" into the "Purpose:" textbox
- click the "b_lb_save.gif" button
- click the "Mary Jones - MyProject weekly meeting" button
- click the "b_lb_open.gif" button

Given these actions, our summarization algorithm produced the following summary: "Meeting reservations on RoomWizard". The words "meeting" and "reservations" were drawn from the labels of the buttons being clicked, and the name RoomWizard was the most frequently-occurring title word across all pages in this task.

Though we have not formally evaluated the performance of the summarization algorithm, anecdotal evidence indicates that the simple algorithm works reasonably well in practice. The generated templates are often meaningful and convey some sense of the actions being performed. Table 2 shows a representative set of tasks performed on the web and their summaries, taken from one of the authors' personal browsing history. Because summaries are only used as suggestions and the user always has the option of editing the summary before it is posted, summaries do not need to be completely accurate. Our aim is not to perfectly describe user behavior, but to suggest text that inspires the user to write something more descriptive, and reduces the overhead of sharing. Future work will investigate more sophisticated summarization algorithms that more accurately capture users' intents while browsing the web.

EVALUATION

Our primary evaluation of ActionShot has been through two field deployments within our organization and a lab study to examine users' ability to extract a set of reusable actions that might be shared with someone else. We also conducted a preliminary think-aloud study with four subjects from our organization to ensure that the software was usable before deployment. Through this study we found and corrected many usability problems. Despite these problems, overall most users found the tool potentially useful and expressed interest in using the tool once it was available for general use.

Field Deployments

The first deployment of ActionShot was to six participants in our immediate research group, who used ActionShot for 3 months. This first deployment was very informal, but these participants have contributed a large number of bug reports and feature requests from their use of the tool.

The second deployment was more formal, and included 16 participants from a variety of locations throughout our organization. This deployment began with a pre-survey to understand how our participants currently make use of various browser features, such as bookmarks and history. After two weeks, we again asked our participants to complete a post-survey. Participants that responded to both surveys were rewarded with a small gift.

Since the second deployment, we have made ActionShot available throughout our organization. So far ActionShot has been downloaded 669 times and appears to be in continuous use by at least 30 users.

Deployment Results

Users during both the first and second deployments found ActionShot to be useful. Of the 16 users in the second deployment, only 7 responded to the survey. One of the seven encountered a bug in installation that prevented ActionShot from working, and his results have been excluded. Of the remaining participants, 4/6 rated ActionShot as useful. Of the two subjects that did not find ActionShot useful, one reported being more comfortable with current bookmarking and history tools and the other apparently expected the tool to automatically suggest reusable scripts.

We heard several anecdotes from users about how ActionShot enabled them to share their web activity with others. One common use was to create smart bookmarks (reproducing the series of actions required to navigate to a page without a stable URL). ActionShot users reported creating these bookmarks from activities that were not recent however, not just immediately after the task was performed as supported by Hupp's Smart Bookmark [4] system. Our users reported using ActionShot to create and share smart bookmarks for tasks such as retrieving a company's balance sheet and accessing a wedding registry.

Other users reported using ActionShot as a means to help colleagues use complex web-based systems. In one example, two users took charge of an academic journal and had to manage its complex reviewing system. One user used his recorded history to help the other complete processes that the first user had already figured out. The motivating scenario we presented earlier in this paper (of re-finding instructions for submitting a budget to the ACM and sharing it with another user) was an actual scenario experienced by one of this paper's authors. Another user also reported a similar experience with a complex web-based process. This user had submitted a corporate naming request, which is a long and involved process, and promptly banished it from his mind. Sometime later, a colleague asked for help with this process, and the user was able to search his ActionShot history, recover his actions, and provide assistance to the colleague based on his recorded history.

Another use for ActionShot was to replay previous behavior. One user reported using ActionShot to retrace his steps on a website where visited links were not visually differentiated. Another user reported using ActionShot to replay a series of actions that triggered a bug in the software she was developing, in order to reproduce that bug for a colleague.

A third use for ActionShot was to retrieve information from previous browsing sessions. One user reported using an ActionShot screenshot to retrieve the confirmation number for a car rental, which was no longer available on the car rental web site. Another user reported recovering an account number that had been entered into a field. This particular user was away from home and finding the number in ActionShot allowed him to complete a transaction that might otherwise have had to wait a week or more.

Deployment Discussion

Our users made several suggestions to improve the visualizations used in ActionShot, many of which were addressed prior to the subsequent deployments. Our think-aloud users found the small colored bars that represent classes of actions and thick black lines that separate sequences in the History View to be particularly confusing. We improved the colored bars by adding explanatory tooltips, and we

improved the algorithm that determines when to draw black lines to make it more predictable.

We saw that users made use of the various sharing options available in ActionShot, but one user asked for more. Our organization has an internal Twitter-like service used by employees to share confidential status updates behind the firewall. This user asked for ActionShot to integrate with this service so that he could use it to share instructions for building, testing, and deploying software with his colleagues. This also suggests a different usage model than we had anticipated, as this user wanted to tweet literal action descriptions instead of our auto-generated action summary, like the feature we created for use with Twitter.

Although we saw that users were able to reuse actions from their history, in some cases we found that ActionShot could make reuse easier. For example, the user might select a sequence of actions for reuse that was preceded by multiple steps of trial & error navigation through a web site. Such a sequence will not stand on its own, as it is missing the initial steps to get the browser in the correct context to execute the sequence. We implemented a solution that prepends a script operation to go to the URL of the first step in the sequence, which we found is often sufficient.

Other feedback on reusability suggests interesting directions for future work. One user suggested providing an algorithm to automatically recognize the boundaries of a task, given one of the steps contained within that task. Another noted that within the time range that contained the actions he wanted to share, both relevant and irrelevant actions were included; he recommended better algorithms for intelligently filtering out the irrelevant actions. Yet another user desired more precise control over Action-Shot's recording toggle, suggesting that we be able to turn on/off recording on a site-by-site basis using an interface similar to ad blocking software.

Another aspect of reusability is generality. One user found that the history recorded by ActionShot was too specific for his needs. Rather than wanting to play back the exact same actions he had performed previously, he was looking for a tool that could help him create a general script to accomplish a range of similar tasks. An interesting direction for future work could be to use similar ActionShot recordings as input to a machine learning algorithm to produce the generalized script that this user wanted.

Lab Study of Extracting Reusable Actions
In addition to the think-aloud and field deployment studies, we conducted a lab study to isolate and evaluate a user's ability to extract reusable actions with ActionShot. We feel that extraction of reusable actions is particularly important because it is a necessary first step for sharing web activity with others. Our hypothesis is that for tasks the user has already done, extracting reusable actions in ActionShot will take less time than recording those actions from scratch in CoScripter. In this study, we use CoScripter as a baseline for judging the effectiveness of ActionShot.

An important goal for this study was to ensure that our task was not easier than other extraction tasks that an Action-Shot user might perform in the real world. To meet this goal, it was important to consider the steps that an Action-Shot user must complete when extracting reusable actions and then ensure that our task made each sufficiently difficult. The steps necessary for extraction are:

- Identify the relevant steps to extract from the log.

- Fix the extracted steps by removing extraneous actions, which might be due to navigation errors or other mistakes.

- Test the modified steps with the current version of the web site, because the site may have changed since the steps were originally recorded.

Note that it is also common for users to remove sensitive information, such as credit card numbers, and perhaps generalize steps using CoScripter's personal database feature. Note that this step is not unique to ActionShot however, as CoScripter users must also address sensitive information when authoring scripts.

Based on these steps, to create a realistic study we needed to make sure that it would not be trivial to find the relevant steps in the log and to ensure that some extraneous actions appeared within the relevant steps that the user would need to fix. For this study, we have chosen not to increase the difficulty of our extraction task by changing the test web site between recording and testing.

Method
This study used a three phase design. In the first phase subjects performed four web browsing tasks, in the second phase they received training on both CoScripter and ActionShot, and in the third phase they were asked to create scripts to automate tasks from the first phase using both CoScripter and ActionShot. Our goal is to compare the time needed to create a script in each of the tools. This study uses a *between-subjects* design.

The four web browsing tasks in the first phase were designed to familiarize the subjects with these sites and to generate some browsing history within ActionShot for use later in the study. The first 3 tasks involved visiting three different web sites to shop for shirts (threadless.com, theselectseries.com, and typetees.com in that order). The final task required users to create a simple blog entry on tumblr.com.

The design of this phase increases the difficulty for ActionShot users later on in the study. We chose the first three tasks to be very long and very similar. Scrolling was required to see all of the actions from each task in Action-Shot's list view, and the similarity between tasks made it difficult to distinguish which of the tasks were being seen at any one time. We also chose tasks in which we believed users were likely to make mistakes in phase one, which would create extraneous actions to remove later. We felt errors were likely because each of our tasks had several

Table 3. Average completion times, in minutes, for the reusable action lab study.

	CoScripter	ActionShot	Significance
Creation Time	3:41	1:32	$F[1,13]=34.05$ $p < .01$
Test Time	1:32	2:21	$F[1,13]=3.171$ $p = 0.0983$
Total Time	5:13	3:53	$F[1,13]=4.985$ $p < .05$

unique steps that are confusing when using the sites for the first time, especially back-to-back. Many, but not all, of our subjects made errors when using these features of the sites.

In the second phase of the study, subjects were given brief demonstrations of the features of both CoScripter and ActionShot and then asked to create scripts for the tumblr.com task from phase one in both tools. Subjects were allowed to ask questions throughout the demonstration and the training tasks to ensure that they were comfortable with all of the important aspects of both tools.

The third phase required subjects to create scripts for two of the tasks in phase one (theselectseries.com and type-tees.com) using CoScripter for one site and ActionShot for the other. We counter-balanced the order in which the sites were presented and the tool used with each site, giving us four conditions to which subjects were randomly assigned.

During the third phase tasks, we recorded the time needed to "create" the script, to "test" it, and the total time. We chose this breakdown in an attempt to quantify behavioral differences between the tools. We expected users to spend more time *creating* their scripts in CoScripter compared to ActionShot, because CoScripter users must perform every step of the task to record it whereas ActionShot users must only identify the relevant set of actions in their history and then remove any extraneous actions that they can identify. We also expect ActionShot users to take longer when *testing* their script because they have never seen the script run against the real page and they are more likely to find errors that need to be corrected. For both conditions, the beginning of the test phase was defined as navigating to the beginning of the script and pressing either the "run" or "step" buttons in CoScripter that begin script execution. Users were instructed to test their scripts after creation; however, one subject declined to test their script in the CoScripter condition.

Participants
We recruited 14 participants from our research lab. Most had prior programming experience and 11 had used CoScripter in some way previously. Only 8 of the CoScripter users had attempted to create a script using CoScripter, and none of the participants had any experience with ActionShot. Participants were compensated with an $8 lunch coupon for our research lab's cafeteria.

Results and Discussion
The results of the study are shown in Table 3. We found that ActionShot was significantly faster than CoScripter in creation time ($p < 0.01$) and total time ($p < 0.05$). There was no significant difference between testing time when using CoScripter or ActionShot. The large difference in the average testing time for the two tools may be due to two large outliers in the ActionShot condition.

It is important to ask whether the difficulty of our task was realistic. In terms of selecting actions, we believe our design was a partial success. We did see many users browse the sessions unsuccessfully, but most spent very little time browsing and instead searched using the site url as a keyword. This worked well for the subjects that tried it, because there was only one set of actions for each site in the log. We believe that search will often be an effective means of finding relevant actions, however its effectiveness will depend greatly on how often a user visits the site for which they are creating a script.

As we had hoped, users made mistakes in their phase one tasks, which increased the difficulty of their later phase three tasks. Users not only made errors on the unique steps that we had expected to be confusing, but also made other errors, such as by skipping over a required field during data entry. Skipping a field added extra button presses and corrective text entry actions into the recorded stream. This suggests that our design worked in most cases, and that dealing with extraneous actions is reflected in our results.

Overall, we found that users of ActionShot are more effective at extracting a reusable set of interactions compared to the baseline of using CoScripter to create a new script from scratch. This result might change if search cannot be used to rapidly identify the correct actions or if the web site changes. In the future, we plan to improve our search function, which already works in many situations, and we will investigate how we might adapt extracted steps to match a changed web site.

DISCUSSION
Security and privacy are important issues for ActionShot to address, and we have spent time considering how to effectively balance the need for privacy and security with the usability of the ActionShot tool. Our current design favors usability and relies heavily on the user to manage the security of their data. ActionShot does not send any of its recorded data across the network unless the user explicitly shares actions, so its data is secure provided the physical machine is kept secure. This is hard to guarantee however, as machines may be stolen, hacked remotely, or shared temporarily with another user.

One potential solution for preserving security and privacy is to encrypt ActionShot's data on the hard drive and also require a password to access the ActionShot panel. We have not yet implemented this feature, in part because we do not have a solution for encrypting the sqlite database and in part because we are concerned about the usability impact of requiring users to take an extra step to access their history data. We will explore this further in the future.

We are also considering methods for filtering sensitive information from the recorded data, which would obviate the need to password-protect the ActionShot panel. For example, ActionShot already does not record any values entered into HTML password fields. We have considered using data detectors to extend this capability to other fields. For example, we could use a reliable credit card number data detector to automatically prevent credit card numbers from being recorded. The challenge is that any detector is unlikely to be perfect, resulting in some sensitive data being recorded. We could also allow users to specify whether certain form fields should be recorded, which could be reused every time the user revisits that web page. We could also allow users to block recording on entire web sites, for cases where visiting the site at all is sensitive.

The quality of ActionShot's historical information depends greatly on the recorder, which must detect actions and generate high quality labels for the interactive elements. This becomes especially difficult for pages that use complex HTML and JavaScript. ActionShot relies on CoScripter's recording infrastructure, which can detect any action on a standard HTML form element and often generates high quality labels. Some complex web pages use custom widgets however, and these widgets are not always recorded correctly. CoScripter was recently extended to support the popular Dojo Toolkit, allowing it to record sites created using that toolkit. In principle it could be extended to support other toolkits as well, such as the Google Web Toolkit.

We have a number of plans for the future of ActionShot. First, we are publicly releasing ActionShot with a new name (CoScripter Reusable History) through the CoScripter web site:

`http://coscripter.researchlabs.ibm.com/`

We hope that this public release will allow us to further investigate the usefulness of ActionShot's improved web history across a much wider range of users.

We would also like to improve several existing features of ActionShot. For example, the visualization of activity taking place across different tabs. The search feature could also be improved, perhaps by adding page content into the search index. We are also interesting in adding new functionality to ActionShot, such as a feature that suggests scripts based on the user's current and previous activities.

We also plan to investigate other applications for enhanced browsing history that might make ActionShot even more useful. Some ideas include: helping users track how their time is spent; predictive auto-completion of web actions based on historical activity; inferring higher-level descriptions of behavior, to enable users to reflect on their past activity; and automatic identification and suggestion of potential scripts based on repeated user activity.

CONCLUSIONS
We presented ActionShot, a system for continuously recording user actions in the web browser and capturing

them as a fine-grained history of browsing behavior. ActionShot demonstrates three different ways that this detailed history can be useful to users. First, ActionShot provides visualization interfaces to explore and search through the user's detailed history for sequences of actions. Second, ActionShot allows reuse of actions from the history; a lab study showed that subjects were able to locate relevant action sequences and convert them to a reusable CoScripter script faster than using the original CoScripter interface. Finally, ActionShot goes beyond Del.icio.us, Google Web History, and other social bookmarking sites by allowing users to share *what they did* on web pages; anecdotal reports from our field deployments have found that ActionShot helps users' share their web activity with others.

REFERENCES
1. Andreessen, M., *NCSA Mosaic Technical Summary.* National Center for Supercomputing Applications, 1993.

2. Anupam, V., Freire, J., Kumar, B., and Lieuwen, D., Automating Web navigation with the WebVCR. *Computer Networks*, 2000. **33**(1-6): 503-517.

3. Heer, J., Mackinlay, J.D., Stolte, C., and Agrawala, M., Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 2008. **14**(6): 1189-1196.

4. Hupp, D. and Miller, R.C. Smart Bookmarks: Automatic Retroactive Macro Recording on the Web, in *Proceedings of UIST.* 2007: 81-90.

5. Jin, J., Won, S.S., and Hong, J.I. Contextual web history: using visual and contextual cues to improve web browser history, in *Proceedings of CHI.* 2009: 1457-1466.

6. Kaasten, S., Greenberg, S., and Edwards, C. How People Recognize Previously Seen Web Pages from Titles, URLs, and Thumbnails, in *Tech Report 2001-692-15, Department of Computer Science, University of Calgary.* 2001

7. Kurlander, D. and Feiner, S. A history-based macro by example system, in *Proceedings of UIST.* 1992: 99-106.

8. Leshed, G., Haber, E., Matthews, T., and Lau, T. CoScripter: Automating & Sharing How-To Knowledge in the Enterprise, in *Proceedings of CHI.* 2008: 1719-1728.

9. Nakamura, T. and Igarashi, T. An application-independent system for visualizing user operation history, in *Proceedings of UIST.* 2008: 23-32.

10. Safonov, A., Konstan, J.A., and Carlis, J.V. Beyond Hard-to-Reach Pages: Interactive, Parametric Web Macros, in *Human Factors and the Web.* 2001

11. Woodruff, A., Faulring, A., Rosenholtz, R., Morrison, J., and Pirolli, P. Using Thumbnails to Search the Web, in *Proceedings of CHI.* 2001: 198-205.