InterTwine: Creating Interapplication Information Scent to Support Coordinated Use of Software

Adam Fourney[1] **Ben Lafreniere**[1] **Parmit Chilana**[2] **Michael Terry**[1]
[1]Cheriton School of Computer Science, [2]Management Sciences
Univeristy of Waterloo, Waterloo, ON, Canada
{ afourney, bjlafren, pchilana, mterry }@uwaterloo.ca

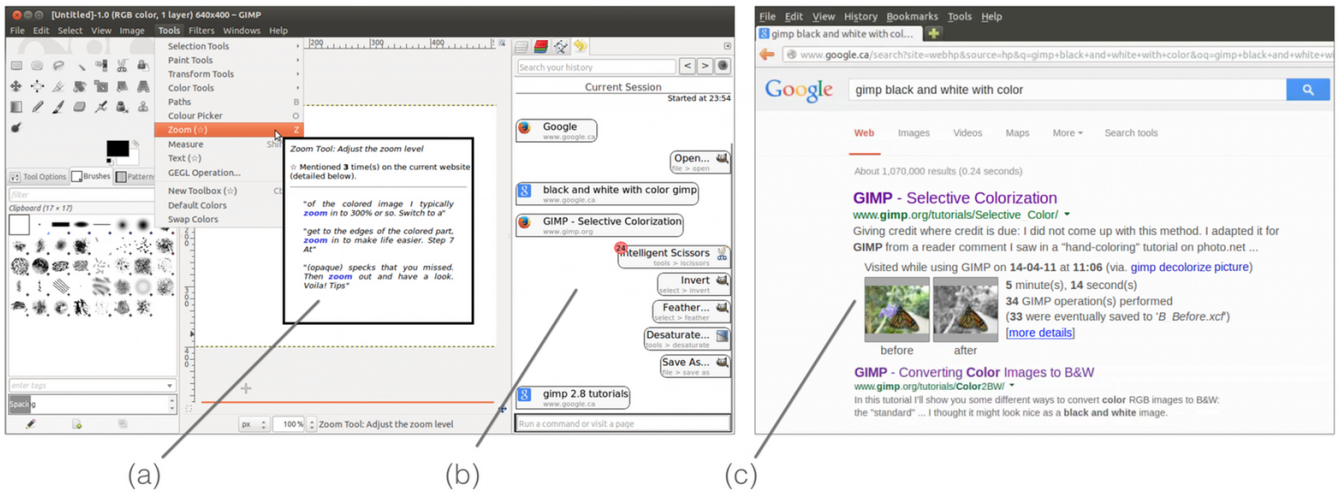

Figure 1. InterTwine links web browsers with feature-rich software applications to create *interapplication information scent*. Specifically, InterTwine modifies application tooltips to describe how they are mentioned on the web page currently open in the browser (a). Likewise, InterTwine extends web search snippets with details of how work documents evolved when that web page was last accessed (c). Finally, InterTwine maintains an interactive interapplication event history to facilitate re-finding past actions and relevant web pages (b).

ABSTRACT

Users often make continued and sustained use of online resources to complement use of a desktop application. For example, users may reference online tutorials to recall how to perform a particular task. While often used in a coordinated fashion, the browser and desktop application provide separate, independent mechanisms for helping users find and re-find task-relevant information. In this paper, we describe *InterTwine*, a system that links information in the web browser with relevant elements in the desktop application to create *interapplication information scent*. This explicit link produces a shared *interapplication history* to assist in re-finding information in both applications. As an example, InterTwine marks all menu items in the desktop application that are currently mentioned in the front-most web page. This paper introduces the notion of interapplication information scent, demonstrates the concept in InterTwine, and describes results from a formative study suggesting the utility of the concept.

UIST 2014, October 5–8, 2014, Honolulu, HI, USA.

http://dx.doi.org/10.1145/2642918.2647420

Author Keywords

interapplication information scent; finding & re-finding

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION

When using desktop software, users frequently rely on the web to access tutorials, troubleshoot problems, or learn how to use specific software features [5, 6]. As users gain experience with the application, they often continue to rely on these external resources, learning *where* to retrieve relevant task-specific information, rather than committing specific operational details to memory [3, 14, 25]. In all cases, online materials provide information that is useful for the successful completion of a given task. This tightly coupled use of web-based information with desktop software suggests that it is worthwhile to consider these separate systems—the web browser and desktop application—as parts of a *single system* for performing work [2].

In this paper, we are concerned with the processes of finding and re-finding task-specific information when using desktop software. In this context, the theory of information foraging provides a useful framework for considering these practices [22]. Information foraging posits that people make use of

information scent to guide their selection and use of information resources within "patches," or collections, of information [4]. When using the web, information scent is provided by elements such as a search engine's autocomplete service, the short page snippets shown in search results, or previously visited links rendered in a different color. In desktop software, menu hierarchies, command names, tool icons, and tooltips all provide affordances that can be considered forms of information scent that assist users in finding relevant functionality.

While these separate systems each provide useful forms of information scent to guide the pursuit of desired information, they largely function independently of one another: the activities in one application have no effect on the presentation of information in the other, forcing the user to manually link information patches between the two applications. For example, desktop software generally has no awareness of when the user turns to the web to learn how to complete a task with the software. Thus, when the user finds relevant information on the web, they must manually connect that information with the affordances and cues provided by the desktop software. In this research, we are motivated to develop mechanisms that more effectively link these distinct systems, to ease the processes of finding and re-finding information within and across applications.

This paper describes *InterTwine*, a system that introduces the concept of *interapplication information scent*. Interapplication information scent links the separate information patches of the web browser and desktop software, injecting novel forms of information scent into both applications to facilitate the finding and re-finding of information. For example, InterTwine embellishes a desktop application's menus with markers (what we call "beacons") for menu items described in the currently open, front-most browser window or tab (Figure 1a). Our current implementation explores these concepts in the context of the Firefox web browser and the GNU Image Manipulation Program (GIMP).

InterTwine is composed of three conceptual entities: 1) a shared, interapplication history produced by recording actions in the web browser and desktop application, 2) mechanisms to identify information likely to be of relevance from this shared history, and 3) novel interface mechanisms that introduce context-aware, interapplication information scent synthesized from this shared history.

Within this system, we demonstrate three different classes of interapplication information scent: application bridges, history snippets, and history digests.

Application bridges are information scent cues that link information in the currently open web page with the relevant features of the desktop application. The previously mentioned menu beacons are an example of application bridges. InterTwine also bridges applications by injecting relevant snippets from the open web document into the tooltips of the desktop application. Together, these bridges help users establish and maintain a shared information context *across* application spaces, easing the process of locating information referenced in one application in the other application.

History snippets are a form of information scent that communicate the context surrounding the past use of a command or web page. InterTwine provides history snippets by embellishing tooltips with a paged display that lists the commands and web pages leading up to, and following, previous invocations of the command. This context helps people re-trace past steps at a glance, and re-access relevant web content.

History digests provide context-dependent summaries of how the desktop application was used with respect to a given web page. InterTwine presents these history digests by augmenting Google search results with summaries that include before and after screenshots of the related application document, and the commands invoked in the desktop application when that web page was open. Warnings are presented to the user for pages in which operations were performed but later abandoned (suggesting the page may not have been that useful for the task). This shared information scent helps users more effectively locate web pages previously found to be useful.

Collectively, this paper makes the following contributions:

1. We present results from a formative study that informed the need for, and design of, mechanisms that provide interapplication information scent
2. We introduce the concepts of a shared, interapplication history and interapplication information scent, and demonstrate three types of interapplication information scent: application bridges, history snippets, and history digests
3. We demonstrate these concepts within InterTwine, a system that ties together the separate information spaces of Firefox and GIMP
4. We validate these concepts via the results obtained from our formative study

In the remainder of this paper we review related work, then present the details of our formative study. We describe the InterTwine prototype, and the types and instances of interapplication information scents it provides, then describe our strategy for constructing interapplication history. Finally, we report how participants of the formative study and iterative design process responded to the final version of the software, and discuss future directions for research.

BACKGROUND

The central premise of this work is that web browsers, online materials, and feature rich applications are often used together as a single productivity system. This perspective is strongly supported by recent research that examines how web search and access to online resources affects work practices [3, 8, 14]. In particular, past work has observed that people are more likely to be able to recall how to re-access helpful material than the actual steps necessary to complete a task. This effect is especially prevalent in the field of software development, where programmers make extensive use of online resources when writing code [3], or when answering technical questions in online forums [8]. More generally, Sparrow et al., describe this phenomena as "*an adaptive use of memory – to include the computer and online search engines as an external memory system that can be accessed at will*" [25].

When users rely on online materials as an external memory store, they are counting on their ability to re-access those materials as needed at a later date. However, people rarely take proactive measures to ensure their ability to return to important information (e.g., bookmarks are underutilized) [12]. Instead, people trust their ability to re-find information using strategies such as web search. Unfortunately, this re-finding strategy is imperfect: users often express frustration in recreating their search sessions [21], and fail to accurately remember up to 30% of previously issued searches after only a few hours [26]. These breakdowns motivated the development of systems such as SearchBar [20] and ActionShot [17], which are both designed to help users more effectively retrace their browsing sessions using data automatically collected in the browser. InterTwine functions similarly to these systems, but is differentiated by two important details: (1) InterTwine focuses on *interapplication* history, and (2) InterTwine actively modifies existing interfaces to enhance and create information scent. We discuss these two details next.

Shared, Interapplication Context

InterTwine maintains a shared interapplication history between the application and the browser. Other projects arrange for applications to share their immediate state or *context*, without maintaining historical details. For example, Pongnumkul et al.'s *Pause and Play* system [23] demonstrates how a 3D modeling application can work together with a video player to more effectively manage video playback when users are following video demonstrations. Likewise, Ekstrand et al. explored how search engines might leverage contextual cues provided by a vector graphics drawing application in order to return more relevant search results for a given task [5]. InterTwine is differentiated from both projects by its focus on injecting novel forms of information scent, and by its use of historical data to help users re-find pages, or to retrace their previously performed steps.

Hartmann's HyperSource [10], and Goldman's Codetrail [9] are systems that also share context between a browser and a given application. More specifically, HyperSource helps developers document and track the origin of code copied and pasted from websites [10]. Codetrail is similar to HyperSource, but forms these associations automatically by comparing recently written code to the text of pages in the web browsing history [9]. InterTwine builds on these concepts, extending them to end-user applications which require different forms of shared context and information.

Scented Widgets and Snippets

InterTwine creates interapplication information scent through embellishments to application widgets and search snippets. In the literature, there are a number of projects that also embellish widgets to enhance their information scent. Notably, Willett et al. present guidelines and a Java framework for embellishing standard interface widgets with small visualizations to enhance navigation of information spaces [27]. Likewise, Matejka et al. demonstrate how interfaces can be overlaid with heat maps describing past use of the software by the user, and by his or her community [18]. InterTwine draws upon these projects for motivation, but differs from these projects by its focus on building and maintaining contextually aware *interapplication* scent trails. For example, InterTwine embellishes widgets only if they are mentioned in the website that is open in the user's browser.

Finally, there are a number of projects that modify or extend search result snippets to enhance their information scent. Ekstrand's system [5], discussed previously, embellishes Google search result snippets for Inkscape tutorials by listing the Inkscape commands listed therein. Likewise, Schwarz and Morris extend search result pages with badges, bar charts and other visualizations, designed to help users judge the credibility of a particular search result [24]. InterTwine's embellishments differ in that they are personalized using interapplication history.

In summary, users of complex software leverage online resources as a form of external memory, but rarely take proactive measures to ensure their ability to re-access this information in the future. Instead, users attempt to re-find materials using search—a strategy that often fails. InterTwine addresses this issue by maintaining a shared interapplication history. It then uses this history to create interapplication information scent, embellishing search results and interface widgets to support finding and re-finding information. While past work has separately explored many of these issues, InterTwine unites these ideas into a single system.

FORMATIVE STUDY

To guide development of InterTwine, we conducted a formative study to: 1) understand the breakdowns that occur when online resources are used to support work in feature-rich applications, and 2) collect feedback on early designs of InterTwine. Eleven individuals (six male, five female, mean age of 26), with varying levels of experience with image editing software, participated in this study. Participants received a $10 Amazon gift card as remuneration for participating in the formative study.

Each session was divided into two parts. In the first part of the session, we asked participants to perform a pair of tasks (described below) using unmodified versions of GIMP 2.8 and Firefox 26. This afforded an opportunity to observe, firsthand, how people find, follow, and re-find online materials when performing tasks in a feature-rich application.

In the second half of each session, participants performed the exact same tasks again, but using our experimental interface designs. These designs ranged from sketches to fully implemented prototypes. For non-functional prototypes, we explained how they functioned and asked users for feedback on their utility to complete the tasks they just performed. For functional prototypes, participants were asked to think aloud as they used the prototypes to complete the same tasks again. Whenever possible, our prototypes were populated with data generated in the first half of each session, thus giving participants a chance to evaluate designs with "live" data. Designs were iterated after each interview, to continually evolve the prototype.

We chose two tasks whose solution is non-trivial in GIMP. One task was to place a thick black border around a sample

of large text (i.e., outlining the text). The other task was to modify a color photo so that the background was black and white (i.e., selective desaturation). In all cases, tasks were presented as pairs of before and after pictures without any descriptive text. The order of presentation of tasks was counterbalanced across participants. Prior to performing the first task, participants were strongly advised to seek online materials for assistance.

Participants' actions were captured by event logging software, screen capture software, and an audio recording device.

Summary of Study Results and Implications for Design

In our study, we found that participants initially experienced trouble locating commands mentioned in the web-based tutorials when switching back to GIMP. Participants also had difficulty recalling their previous actions, both online (e.g., search queries, as in [26]), as well as in the application (e.g., commands and procedures).

While we expected participants would encounter some of these issues, they occurred more frequently and with greater severity than expected. For example, we observed participants identify promising commands mentioned in tutorials, and then almost immediately forget the identity, location, and details of those commands when switching into the desktop application. This difficulty was observed even when tutorials explicitly and unambiguously mentioned the locations of commands in the interface. In all of these situations, participants typically adopted a strategy of systematically exploring an application's menus and tooltips in the hope of recognizing their target.

These results suggest there is value in linking the separate information spaces of the web browser and desktop application, to make finding information presented in one application easier to locate in the other application. The difficulty in recalling past actions also suggests the value of mechanisms that assist in re-finding task-specific information at a later time. These results directly led to the development of application bridges and history snippets in InterTwine.

Our iterative design process also revealed that participants were enthusiastic about seeing previously edited GIMP work documents associated with relevant web pages in web search result snippets. However, they expressed disinterest when presented with fine-grained details of errors, dead-ends, or unsuccessful sequences. Accordingly, we developed history digests that summarize past activities, while indicating which web pages appear to have had no effect on advancing the solution.

We also asked participants how they would feel if non-relevant web pages were wrongly associated with GIMP work documents (e.g., linking a work document to a news article that was coincidentally open in the web browser). Participants commented that such errors were easy to spot from the snippets provided, and could be safely ignored, but posited that a high frequency of errors would quickly eliminate any such system's advantages over the status quo. As such, InterTwine takes several measures to ensure the relevance of any associations made between web pages and GIMP documents.

Finally, when presented with a shared history depicting the use of both applications, our participants expressed a desire to see the full history, without omissions. Early designs revealed only the most important commands or web pages, and were generally disliked, as were designs that presented browsing and procedural details in separate locations. Accordingly, InterTwine includes a shared history that shows *all* activity in both applications.

We now describe InterTwine's design in detail.

INTERTWINE

InterTwine is composed of three conceptual parts: a shared interapplication history, mechanisms to identify potentially relevant information from that history, and interapplication information scent that helps users link the separate information spaces of the web browser and desktop application.

We implemented these individual parts via a plugin-in for the Firefox web browser, a modified version of the GIMP image manipulation system, and a shared datastore and associated shared history service that mediates communication between Firefox and GIMP.

While the shared history service serves a key role in the system, it operates automatically in the background, and is not directly accessed by the user; users access its capabilities through the various interface components described below.

InterTwine modifies GIMP by adding an *interapplication history transcript*, by embellishing its menu items with *beacons*, and by adding *history snippets* to tooltips. In Firefox, InterTwine augments Google search result pages with *history digests*. We describe each of these components in turn.

Interapplication History Transcript

The foundation of the InterTwine system is an interapplication interaction history. While these data are used to derive many of InterTwine's other features, users can also directly view and interact with this history.

Our current implementation depicts this shared history in a pane that adopts the metaphor of a chat program. When an action is performed in an application, a "speech bubble" is produced representing that action (Figure 2). Entries contributed by GIMP appear on the right. Entries contributed by Firefox appear on the left. We chose this metaphor to make it easier to visually parse the histories, and to establish a common visual design that can be used in other parts of the interface.

All items displayed in the transcript are interactive. Clicking on a GIMP command bubble causes the command to be invoked, and clicking on a web page bubble causes the web page to open in Firefox and the web browser to come to the foreground.

The transcript can also be searched. InterTwine's transcripts are indexed by command names, command tooltips, file names, web page titles, internet search queries, web page body text, dates, and times. This extensive coverage is designed to allow users to index into the transcript using almost any detail recalled about a previous session, and addresses the goal of helping users re-find information.

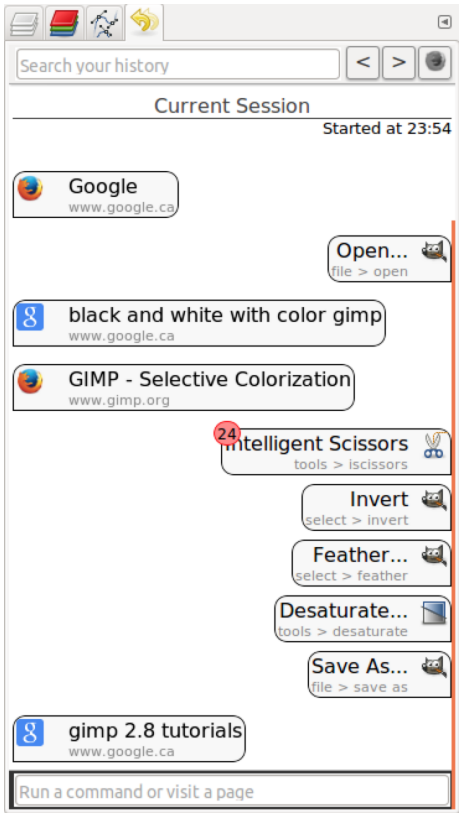

Figure 2. InterTwine adopts the metaphor of a chat program to communicate interapplication event history. When users issue commands in GIMP, the commands appear as speech bubbles on the right-hand side. When users visit pages in Firefox, the pages appear as speech bubbles on the left-hand size.

Finally, embracing the notion of a chat system, users can directly add to this transcript by typing at the chat prompt (Figure 3). When users begin typing at the prompt, their input is automatically completed with the names of GIMP commands, as well as the titles of web pages in the browsing history. Selecting an item from the list of suggestions causes the command to be performed, or the website to be opened and brought to the foreground. This capability draws some inspiration from Hendy et al.'s graphical enhanced keyboard accelerators [11].

As with all InterTwine features, autocompletion makes extensive use of the interapplication history and the current state of both applications when determining what suggestions to provide, as well as their order. As an example, if a web page is open in the user's browser, then the suggested commands are embellished with one style of beacon (i.e., marker) if they are mentioned in that web page, and another style of beacon if they were previously used the last time that web page was open (as seen in Figure 3).

Interapplication Information Scent in Menus and Tooltips

InterTwine modifies the presentation of GIMP menus and tooltips to provide interapplication information scent. This information scent is informed by the shared history and the currently visible web page.

Figure 3. A "chat prompt" is located directly below the transcript. From this prompt, users can execute GIMP operations or visit web pages by typing commands. The prompt autocompletes the input, allowing users to issue commands with only a few key presses.

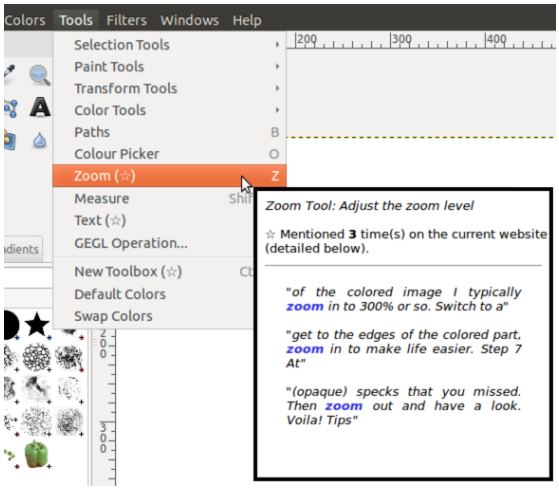

Figure 4. InterTwine embellishes menu items with beacons (star icons) when those items are mentioned by name in the currently visible web page. Likewise, menu tooltips gain snippets describing the context in which each menu item is mentioned in the web page.

InterTwine embellishes menu items with a hollow star icon (a beacon), to communicate that a given menu item is mentioned on the web page the user currently has open in the browser (Figure 4). These menu items' tooltips are also augmented to present web page excerpts that mention the given menu item. These beacons and excerpts are designed to increase the information scent of relevant menu items, and help bridge the separate information spaces of the web page and the desktop software.

InterTwine's menu items are further enhanced based on their history of use. Specifically, if users have previously visited a web page and issued a command with that page open, then InterTwine displays a filled star icon next to the menu item to indicate past relevance. Furthermore, the item's tooltip contains excerpts of the interapplication history transcript detailing its context of use (Figure 5). In cases where the menu item has been used in multiple contexts, users can page through

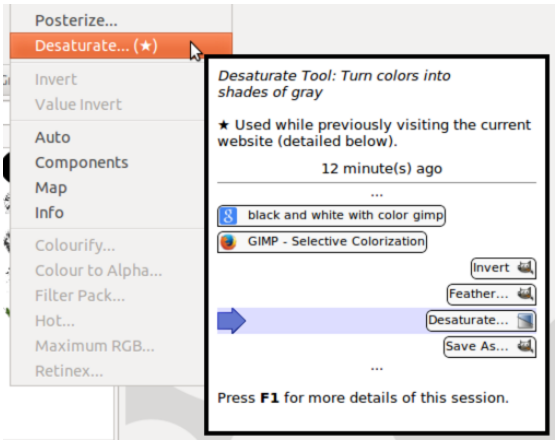

Figure 5. When revisiting a web page, InterTwine places additional beacons (filled-stars) next to commands that were used before in the context of the web page. In these cases, the tooltips gain excerpts from the interapplication history, describing their earlier context of use.

a slideshow of these excerpts using the left and right arrow keys. Finally, at any time, users can press the *F1* key to scroll the full interapplication history (described above) to the corresponding time when the command was used.

Interapplication Information Scent in Web Search Results

In Firefox, InterTwine modifies the Google search results page by enriching the standard search result snippets with details extracted from the interapplication history.

In situations where a search result item has been previously visited, InterTwine retrieves details of the past visit, and generates a summary (or digest) of this information for review (Figure 6). These digests include two screenshots of GIMP's canvas: one taken when the user first accessed the search result, the other taken when leaving the target web page. These images serve as a visual summary of the work that was done when previously visiting the page.

The snippet also details how long the page was previously open, as well as the number of GIMP operations performed while the page was focused in the browser. Two different counts are presented: the total number of operations performed, and the number of operations ultimately saved to the work document. These counts differ when commands are undone, or when the user closes a document without saving or exporting a result. In extreme cases where all commands are abandoned, the snippets instead present a warning (Figure 7).

Finally, users can optionally click a hyperlink in the snippet to reveal the context in which the page was previously accessed. As with tooltips, these details are presented as excerpts from the interapplication history.

IMPLEMENTATION

InterTwine's implementation consists of three components that interoperate on a user's local machine: a modified version of the GIMP image manipulation system, a plugin-in for the Firefox web browser, and a local coordination service that mediates communication between Firefox and GIMP. We describe each of these in turn.

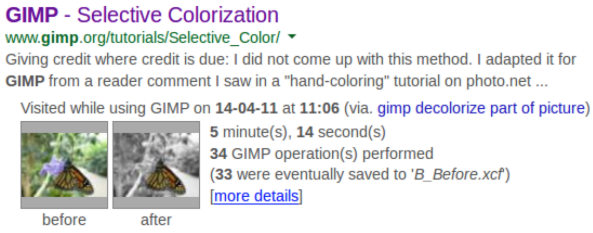

Figure 6. InterTwine modifies Google search result snippets when web pages have been previously visited. Here, InterTwine presents history summaries, which include screenshots of how the user's document evolved when previously reading the page. Additionally, the summaries present statistics describing the time spent, and the number of commands issued.

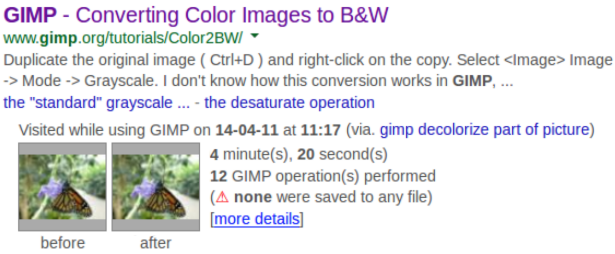

Figure 7. InterTwine's history summaries present warnings to the user in cases where earlier page visits resulted only in commands that were later abandoned (e.g., commands that were undone or unsaved).

InterTwine requires two distinct sets of modifications to GIMP. First, GIMP must be instrumented to record a user's low-level interactions with the software, as well as to record screenshots of the user's work document as it involves over time. GIMP must also be modified to display the interapplication history transcript, as well as custom menu items and tooltips. InterTwine implements the transcript and the custom tooltips by embedding a Webkit browser directly into the GIMP application. As such, InterTwine's tooltips and transcripts are themselves implemented in HTML and JavaScript. This architecture was especially useful during the formative study, allowing us to quickly iterate InterTwine's designs.

As with the modifications to GIMP, InterTwine's Firefox plug-in serves two roles. First, it instruments the browser to track a user's actions online. Second, it modifies the presentation of the interface (i.e., Google search results). In both cases, these actions are achieved by injecting custom JavaScript code into the pages a user visits online. Instrumentation is achieved by coercing visited web pages to signal interface events (e.g., page loads, scrolling, etc.) by making an asynchronous request to InterTwine's local service.

Finally, InterTwine's local coordination service consists of a lightweight web server running in the background of the user's machine. This service both collects instrumentation data from GIMP and Firefox, and generates content for GIMP transcript and tooltips, as well as the Google search result snippets. Additionally, this service processes the instrumentation data to generate the interapplication history. Next, we describe our method of generating and refining the interapplication history.

Creating and Refining Shared Histories

In our formative study, users made extensive use of tabbed browsing, and followed hyperlinks by opening each in a new tab. Once opened, tabs were rarely closed before the end of each session. Instead, participants returned to earlier pages by switching tabs rather than by relying on the browser's backward and forward buttons.

We also observed two users adopting a strategy of "prefetching" search results by opening promising links in new tabs prior to visiting any individual result. In these cases, it was common to retrieve pages that were never actually consulted.

Finally, when participants leveraged online resources to perform unfamiliar tasks, they often continued to explore and experiment with the application's interface, sampling the application's capabilities and undoing many commands.

All of these practices complicate the process of producing a meaningful, shared interapplication history: By the time users complete a task, the histories and interactions logs are extremely noisy, with relevant commands and web pages hidden in a sea of dead-ends and failed experiments. Accordingly, we found it necessary to process and filter the shared interapplication history.

In an effort to improve upon the naïve approach to creating a shared history, InterTwine gathers additional interaction details not present in standard browsing histories. In particular, InterTwine records which browser tabs are visible at any given moment, providing an indication of relevance for each tab. Likewise, InterTwine records interaction events that occur on web pages (e.g., page scroll events), and uses these events to estimate the degree to which a page is being utilized. Specifically, application commands are attributed to web pages only if the web pages mention the name of the application ("GIMP") in their body text, and if the application commands occur within 5 minutes of a web page interaction event.

Additionally, InterTwine tracks the outcome of each command, noting if the command is ever undone, and if not, whether the application document is ever saved (indicating that the work in that document was deemed useful). These data are then reflected in the history digests shown with Google search results: digests display whether a visited web page previously contributed any commands that were still intact when the work document was saved.

FEEDBACK ON FINAL DESIGN

To evaluate the final design of InterTwine, we invited participants from our initial formative study to return 15 days after their initial session. In addition to collecting feedback on how the design had improved (or regressed), this final session afforded an opportunity to observe breakdowns that occur when people repeat tasks several weeks apart, providing a way to validate the system's core purpose (i.e., to assist in re-finding information). Five participants responded to our invitation, and returning participants received an additional $10 Amazon gift card as remuneration.

In this return session, participants were asked to perform one of the original tasks. Since each participant experienced a different prototype in the formative study, and because early prototypes were not always functional, we could not carry forward a participant's earlier interapplication history. Instead, participants were greeted with a fresh installation of GIMP, Firefox, and InterTwine. As with the formative study, minimal instructions were provided. Tasks were presented as pairs of before and after pictures without descriptive text, and participants were advised to seek online materials. InterTwine's features were discussed as they were discovered, and participants were asked for their interpretations before correct use was demonstrated.

Results

At the onset of this final session, all five participants felt they remembered enough about the task to complete it without consulting the web. However, when the time came to actually perform the task, none were able to complete the task without consulting web resources. When asked about this discrepancy, one participant explained:

> *"I think my dependence on the Internet is pretty high. I'm sure I could have [completed the task], but sometimes I doubt myself and think I'll do this faster with Google"* **P2**

After participants visited a few web pages, we called attention to InterTwine's interapplication history, and provided a brief overview of InterTwine's other features. Participants were then asked to continue the task, and to provide feedback as they worked.

Participants were generally positive about InterTwine's individual features. For example, P1, P3 and P5 were very enthusiastic about InterTwine's menu and tooltip enhancements. P3 had the following to say about these features:

> *"You don't have to guess anymore. When you read something and you think yeah, OK, and [then]... I mean, I was powering through this pretty quick, because I thought I knew how to do it. But if you are on a web page and you go up there and... [in] the two seconds you actually read something versus going to do it, you forget what's going on, and then boom, it's right there [referring to InterTwine's beacons] "* **P3**

Likewise, participants P2, P3 and P4 explicitly mentioned finding the shared history to be a useful tool. P2 reported this feature as the single most significant improvement from the prototypes discussed in the formative design process. P3 was especially enthusiastic about the shared history's command prompt, stating:

> *"I'm just thinking like for an advanced user, someone who has been using it for a while, just having the quick keys [command prompt] down here rather than going through all that, that's kind of nice... Yeah, so once you get handy with that I can see it being really, really powerful."* **P3**

All five participants felt that the historical digests, added to Google search results, were especially useful. This was not unexpected, as 10 of the 11 original formative design participants responded enthusiastically to prototypes with this feature. On this topic, P5 noted:

> *"I like where it says how long you've been on the web page and what [you] did. It will tell me I saw this web page quickly and I didn't like it so I X'ed out of there."* **P5**

Finally, P2 noted that InterTwine's features were compelling enough to switch image editors:

> *"My image editor of choice is Paint because it's really simple... I do have Paint.NET on my computer, but I don't use it. But with this sort of input, I would be more likely to use GIMP, because it would help me do some more advanced things that would be difficult to figure out otherwise."* **P2**

Areas for Improvement

While participant feedback was very positive, participants offered suggestions for improvements. For example, despite being very enthusiastic about the menu beacons, P3 initially failed to notice them at all. When the beacons were pointed out by the researcher, the participant explained that the beacons looked like menu text, and he was not sufficiently familiar with GIMP to know if they were something new. This suggests the need to refine the presentation of these markers, so that their designs better communicate their purpose.

Likewise, P5 noted that she found the features of InterTwine's shared history (e.g., search and auto-complete) to be complex, and worried that she would not "use it to its full potential." However, P5 felt that the excerpts of the history, presented in the tooltips and search results page, were acceptable.

Finally, both P1 and P2 requested a mechanism for manually managing the history. In particular, they expressed interest in the ability to rate pages and command sequences, as well as the ability to hide or delete items with poor ratings. We feel that these are excellent feature requests, and that manual management of interapplication history is a compelling design dimension to explore in the future.

A final encouraging sign that InterTwine is offering features that are generalizable, and of value, is that participants offered numerous suggestions for other applications that could benefit from the same interactions. For instance, P5 suggested InterTwine could be applied to SPSS, Minitab, and other statistics packages to help users recall how to perform various statistical tests. Likewise, P1 commented that an InterTwine-like system would be useful when using geographic information systems. Finally, P2 stated:

> *"I'm just thinking like, even writing an essay, like if you had your documents up like in Mendeley or whatever, you can know you wrote this with this document open, and you know what to cite. I'm just thinking of other applications. It seems... I'm just astounded, this is really cool."* **P2**

DISCUSSION & FUTURE WORK

InterTwine represents one targeted exploration of the notion of interapplication information scent. The types of interaction mechanisms chosen in this research were largely determined by following the most salient leads uncovered in the formative study. In this section, we discuss other promising research trajectories, and other possible application domains.

Community Aggregation

InterTwine operates entirely on one's own personal computer, leveraging only personal browsing history and interaction logs. One noteworthy finding from our formative study was that participants were almost entirely unconcerned with sharing their interaction details with a central service like Google (10 of 11 participants had few or no concerns). Given this, one could imagine aggregating a community's interapplication histories, enabling a number of additional applications.

One compelling use of this aggregated data would be to improve indexing of web-based tutorials. For example, a search engine could use this information to influence the ranking of search results, steering users away from tutorials that frequently result in abandoned commands. Likewise, since interapplication history does not depend on page text, these data could be used to index non-textual tutorials such as screencasts and video demonstrations. For example, a search engine could index tutorial videos using the names of the application commands and tools that users typically invoke while visiting each video. Users could then search for video demonstrations by naming commands of interest. With additional instrumentation of the web browser, video timestamps could be extracted and synchronized with application command invocations, allowing search engines to index *into* videos, and enabling capabilities similar to those described in [13, 15, 16]. Beyond indexing web documents and videos, aggregate data could also be applied to command recommendation systems (e.g., [19]), and related projects (e.g.,[7, 18]).

Interapplication Information Scent in Other Domains

InterTwine creates interapplication information scent between a web browser and a feature-rich raster graphics application. Other application pairings are possible—including ones that do not involve a web browser. As an example, a video editing application might notice coordinated use of audio editing software, and could adapt by highlighting the menu commands necessary to insert a new audio track into the video. Likewise, an operating system's file browser might visually modify folder icons in cases where the user has a terminal open in those directories, presenting a persistent visual trail as the user navigates the command shell through the file system. Finally, after using graphical user interface design software, such as QT Creator, programming environments could highlight lines of code related to objects recently worked with in the interface designer.

Fading Information Scent

In our current implementation, interapplication information scent is generated from the past history and current context of both applications. These information scents persist as long as the relevant context is present. However, it is possible that

this context (especially from the shared history) could result in too much information scent accumulating, reducing the effectiveness of the concept.

To deal with this problem, interapplication information scent could fade out over time, similar in spirit to Baudisch et. al's [1] phosphor effects in the interface. For example, menu beacons could begin to fade after a web page has been open for 20 minutes. In general, we have not considered the possibility of incorporating hysteresis into our relevance models, but this capability may be especially useful for some of the mechanisms proposed above (such as the trails left through the file system as the user browses directories).

CONCLUSION

Online resources play an integral role in people's strategies for dealing with the complexities of feature-rich applications. However, applications and web browsers currently function as separate isolated entities, unaware of how activities in one application may relate to activities in the other.

In this paper, we introduced InterTwine, a system that bridges these two application domains through the constructs of *interapplication history* and *interapplication information scent*. We demonstrate three classes of tools to provide interapplication information scent: application bridges, history snippets, and history digests. Together, these mechanisms help users find and re-find task-relevant commands and resources.

A formative study spanning two sessions reveals that InterTwine's features resonate with users and suggest their overall utility. Though InterTwine's current implementation is tied to GIMP and the Firefox web browser, we believe the ideas presented in this paper can be generalized to other feature-rich applications. This sentiment is shared by many of those who took part in the participatory design process, as evidenced by their many enthusiastic suggestions for which applications to work on next.

REFERENCES

1. Baudisch, P., Tan, D., Collomb, M., Robbins, D., Hinckley, K., Agrawala, M., Zhao, S., and Ramos, G. Phosphor: Explaining transitions in the user interface using afterglow effects. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, UIST '06, ACM (New York, NY, USA, 2006), 169–178.
2. Brandt, J., Dontcheva, M., Weskamp, M., and Klemmer, S. R. Example-centric programming: integrating web search into the development environment. In *Proc. CHI '10*, ACM (New York, NY, USA, 2010), 513–522.
3. Brandt, J., Guo, P. J., Lewenstein, J., Dontcheva, M., and Klemmer, S. R. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In *Proc CHI '09*, ACM (New York, NY, USA, 2009), 1589–1598.
4. Chi, E. H., Pirolli, P., Chen, K., and Pitkow, J. Using information scent to model user information needs and actions and the web. In *Proc CHI '01*, ACM (New York, NY, USA, 2001).
5. Ekstrand, M., Li, W., Grossman, T., Matejka, J., and Fitzmaurice, G. Searching for software learning resources using application context. In *Proc. UIST '11*, ACM (New York, NY, USA, 2011), 195–204.
6. Fourney, A., Mann, R., and Terry, M. Characterizing the usability of interactive applications through query log analysis. In *Proc. CHI '11*, ACM (New York, NY, USA, 2011), 1817–1826.
7. Fourney, A., Mann, R., and Terry, M. Query-feature graphs: Bridging user vocabulary and system functionality. In *Proc. UIST '11*, ACM (New York, NY, USA, 2011), 207–216.
8. Fourney, A., and Ringle Morris, M. Enhancing technical Q&A forums with CiteHistory. In *Proc. ICWSM '13* (2013).
9. Goldman, M., and Miller, R. C. Codetrail: Connecting source code and web resources. *J. Vis. Lang. Comput. 20*, 4 (Aug. 2009), 223–235.
10. Hartmann, B., Dhillon, M., and Chan, M. K. HyperSource: bridging the gap between source and code-related web sites. In *Proc. CHI '11*, ACM (New York, NY, USA, 2011).
11. Hendy, J., Booth, K. S., and McGrenere, J. Graphically enhanced keyboard accelerators for GUIs. In *Proc. GI '10*, Canadian Information Processing Society (Toronto, Ont., Canada, 2010), 3–10.
12. Jones, W., Bruce, H., and Dumais, S. How do people get back to information on the web? how can they do it better. In *Proc. INTERACT '03* (2003), 793–796.
13. Kim, J., Nguyen, P. T., Weir, S., Guo, P. J., Miller, R. C., and Gajos, K. Z. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *CHI '14*, CHI '14, ACM (New York, NY, USA, 2014), 4017–4026.
14. Lafreniere, B. *Task-Centric User Interfaces*. PhD, University of Waterloo, Waterloo, ON, Canada, Apr. 2014.
15. Lafreniere, B., Grossman, T., and Fitzmaurice, G. Community enhanced tutorials: Improving tutorials with multiple demonstrations. In *CHI '13*, CHI '13, ACM (New York, NY, USA, 2013), 1779–1788.
16. Lafreniere, B., Grossman, T., Matejka, J., and Fitzmaurice, G. Investigating the feasibility of extracting tool demonstrations from in-situ video content. In *CHI '14*, CHI '14, ACM (New York, NY, USA, 2014), 4007–4016.
17. Li, I., Nichols, J., Lau, T., Drews, C., and Cypher, A. Here's what i did: Sharing and reusing web activity with ActionShot. In *Proc. CHI '10*, ACM (New York, NY, USA, 2010), 723–732.

18. Matejka, J., Grossman, T., and Fitzmaurice, G. Patina: Dynamic heatmaps for visualizing application usage. In *Proc CHI '13*, ACM (New York, NY, USA, 2013), 3227–3236.
19. Matejka, J., Li, W., Grossman, T., and Fitzmaurice, G. CommunityCommands: command recommendations for software applications. In *Proc. UIST '09*, ACM (New York, NY, USA, 2009), 193–202.
20. Morris, D., Ringel Morris, M., and Venolia, G. SearchBar: a search-centric web history for task resumption and information re-finding. In *Proc. CHI '08*, ACM (New York, NY, USA, 2008).
21. Obendorf, H., Weinreich, H., Herder, E., and Mayer, M. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. In *Proc. CHI '07*, ACM (New York, NY, USA, 2007), 597–606.
22. Pirolli, P., and Card, S. Information foraging. *Psychological Review 106*, 4 (1999), 643–675.
23. Pongnumkul, S., Dontcheva, M., Li, W., Wang, J., Bourdev, L., Avidan, S., and Cohen, M. F. Pause-and-play: Automatically linking screencast video tutorials with applications. In *Proc. UIST '11*, ACM (New York, NY, USA, 2011), 135–144.
24. Schwarz, J., and Morris, M. Augmenting web pages and search results to support credibility assessment. In *Proc. CHI '11*, ACM (New York, NY, USA, 2011).
25. Sparrow, B., Liu, J., and Wegner, D. M. Google effects on memory: Cognitive consequences of having information at our fingertips. *Science 333*, 6043 (2011), 776–778.
26. Teevan, J., Adar, E., Jones, R., and Potts, M. A. S. Information re-retrieval: repeat queries in yahoo's logs. In *Proc. SIGIR '07*, ACM (New York, NY, USA, 2007), 151–158.
27. Willett, W., Heer, J., and Agrawala, M. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov. 2007), 1129–1136.