Hunter Gatherer: Interaction support for the creation and management of within-web-page collections

Conference Paper · January 2002

DOI: 10.1145/511446.511469 · Source: DBLP

5 authors, including:

m.c. Schraefel
University of Southampton
294 PUBLICATIONS **5,058** CITATIONS

SEE PROFILE

David Modjeska
Canadian Institute for Health Information
24 PUBLICATIONS **295** CITATIONS

SEE PROFILE

Shengdong Zhao
National University of Singapore
125 PUBLICATIONS **2,569** CITATIONS

SEE PROFILE

Hunter Gatherer: Interaction Support for the Creation and Management of Within-Web-Page Collections

m.c. schraefel,[1] Yuxiang Zhu,[1]David Modjeska,[2] Daniel Wigdor,[1] Shengdong Zhao[1]
[1]Dept. of Computer Science | [2]Faculty of Information Systems
University of Toronto
Toronto, Canada
{mc |dwigdor |yuxiang|shengdong}@dgp.toronto.edu
modjeska@fis.toronto.edu

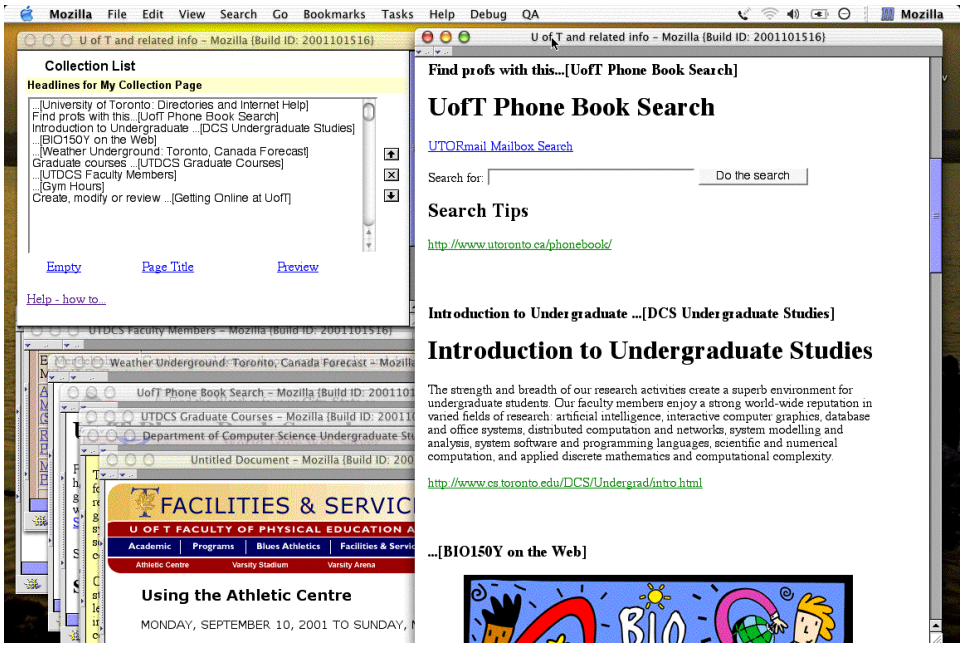

Figure 1. Hunter Gatherer at work. A sample collection page is on the right. Below each component in the collection is the link to the component's source page. Each component has a default, editable title. Collections can contain any web page element: shown here are images, forms and text. In the upper left is the List/Edit window to monitor the collection as it is being created. In the lower left are the pages from which the collection was created. A video demonstration is available at http://shaka.dgp.toronto.edu/hg/overview

ABSTRACT

Hunter Gatherer is an interface that lets Web users carry out three main tasks: (1) collect components from within Web pages; (2) represent those components in a collection; (3) edit those component collections. Our research shows that while the practice of making collections of content from within Web pages is common, it is not frequent, due in large part to poor interaction support in existing tools. We engaged with users in task analysis as well as iterative design reviews in order to understand the interaction issues that are part of within-Web-page collection making and to design an interaction that would support that process.

WWW2002, May 7–11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

evaluations of the tool that evolved from that process, and the future work stemming from these results, in which our critical question is: what happens to users' perceptions of web-based resources and their web-based information management practices when they can treat this information as harvestable, repurposeable data, rather than as fixed pages?

Categories and Subject Descriptors

H5.4 Hypertext/Hypermedia–Architectures, Navigation, User issues. H5.2 User Interfaces–Prototyping.

General Terms

Design, Experimentation, Human Factors.

Keywords

Web-based interaction design, information gathering and management, attention, collections, transclusions

1. INTRODUCTION

Studies of Web-based information interaction such as [2][5], have generally dealt with a Web page as the smallest unit of consideration. Task analysis carried out in a user study reported in [13] indicates that users, however, regularly need to deal with smaller units, that is, information components from *within* Web pages. The study found two things: (1) that Web users want to be able to make collections of information found from within Web pages, but that (2) users only infrequently make such collections, in large part because of poor interaction support for this activity. For instance, bookmarks, referencing entire pages often capture more than the desired data; this forces users first to load and then to sift through multiple pages to attempt to find the desired material. Text editors cause users to shift attention between the information gathering task in the browser and the information management task with the editor. With editors, users often forget or neglect to label the collected component with a title or the URL of the source page, making later access to the original material difficult, degrading the value of the collection over time.

Despite these shortcomings, those surveyed still expressed a need to create collections from material within Web pages. Scenarios for such collections are easy to imagine: a journalist might want to build a collection of different newspaper coverage of the same story. A student might build a heterogenous collection to reflect her current term, including courses, professors, gym hours and so on.

We developed Hunter Gatherer (HG) both to support this kind of within-Web-page collection making and to investigate how this novel interaction design might affect Web-based information practices. Hunter Gatherer (Figure 1) blends the transparency of bookmark capture for component selection, with the support of an editor for revising collections. The tool also automates the inclusion of a contextual, editable header/annotation for each component, and grabs the URL of the source page for that component (Figure 1, right), so that users can return to the source document at any time. Our interaction goal for Hunter Gatherer's design is to let users, rather than the tool, determine which information activity they wish to focus on: gathering, management or contemplation of the collection. Our software goal has been to create a tool that integrates with the browser and utilizes web-based protocols so that the user does not require additional software to carry out these tasks. Our larger research goal is to use this tool to help us investigate both perceptions of and expectations of what might be called *information flexibility* in an information space that has previously defined the smallest unit of information to be the Web page. We wish to investigate how this might change once users have tools which can support information harvesting, in which they can replant or repurpose information elements from one context into ones of their own devising.

Hunter Gatherer is the result of an iterative process of user-based design, surveys and evaluation. This paper describes the most recent version of the artifact, the associated interaction design, and its evaluations. We begin with a discussion of Web-based collection management tools research and illustrate where this work does not address the interaction problem most relevant to within-Web-page collection making: shifting focus between information capture and post-capture information management. We follow this with a discussion of our prototype tool development. We present our evaluation and consequent evolution of the tool over several iterations. Finally, we report on lessons learned from these evaluations, and describe how the results have helped to refine our understanding of the tasks we hope to support, and the steps we wish to pursue in future work.

2. RELATED WORK

Our research investigates the problems faced by Web users who wish to carry out two related tasks: to gather information components from a variety of Web sources and to manage that gathered information. When we focus on information gathering on the Web, we foreground the process that Marshall *et al.* [9] refer to as "information triage," the act of moving through a variety of sources to determine quickly whether they are of potential worth. The sticking point occurs when, on making such a determination, we wish to capture the component identified for retrieval. When users are engaged in information triage, they currently lack a method for putting the identified components into a collection *without* needing to make the collecting activity a foreground task. While there has been much work done on the management of Web-based document collections (which we discuss below), there has been less work on the interaction activity of placing the identified information from the source into the collection. Therefore, our work has focused especially on the latter process.

2.1 Bookmarks and Visualization

Our design model for the kind of transparent interaction that we wish to emulate has been bookmark-making. Bookmarking is well integrated with most Web browsers. The user engages a simple command key sequence, or makes a menu selection, and the current page is added to a list of bookmarks. With slightly more concentration on the bookmark task, users can shift focus to more specific information management tasks: many bookmark tools, for instance, support adding bookmarks directly to specific folders within the bookmark list. Such interaction supports a gradient of task focus, from peripheral attention to main focus. While bookmarking supports this multiple attention level for interaction, its failure to help users retrieve information effectively from bookmarks has been well discussed in Abrams *et al.* [1]. To deal with the shortcomings of bookmarks for retrieving information, several research and commercial applications have been developed. While not completely applicable to our research, there are related findings from that web-based work which inform ours.

Card, Robertson and York's WebBooks [2] is an early example of an application for bookmark visualization. In this work, the entire Web page is always available, eliminating the requirement for a user to load each interesting bookmark iteratively. Collections of pages are visualized as books, where pages in the collection can be quickly "flipped through." While the WebBook eliminates the need for users to load pages, it still focuses on a complete Web page as the artifact of value.

More recently, Robertson *et al.* developed the Data Mountain tool to let users arrange bookmarks as page of thumbnails on an inclined plane. Compared with Internet Explorer's Favorites bookmark tool, participants were able to retrieve pages more quickly and with fewer errors [12]. Czerwinski *et al.* extended this work; they demonstrated that the name and the location of a bookmark on the plane were the two factors most important for successful retrieval; a page's thumbnail image was less important [5].

Amento, Terveen, Hill and Hix's TopicShop work [2][14] draws particularly on the Data Mountain research for letting users manage collections of sites on a given topic. In this case, an algorithm developed for TopicShop captures candidate sites, which become available to a user in a multi-paned window. In the site profile pane, for instance, a list of sites shows miniature thumbnails of the page, along with relevant site characteristics, such as name and number of links in and out of the page. This information helps users decide if they wish to visit the site. Users can then drag chosen sites into a "work area." The site is represented here as a thumbnail. Thumbnails can be "piled" into groups; groups are in turn reflected in the site profile window. Evaluation participants found this multi-view approach to evaluating and organizing collections to be TopicShop's most effective feature.

Once again, the Web page is the entity of value. This makes sense in the case of TopicShop, as the entire page or site is desired overall, since, by design, the pages collected are themselves either all "on topic" (e.g., a fan site) or are collections of links to such sites. It is not clear if the TopicShop algorithm could be extended to capture, for instance, a more heterogeneous notion of topic, as in the preceding student scenario. There, "My Term" as a topic might reflect an associative set of components such as courses and student loan information, rather than clusters of similar information.

2.2 Editors

Some editors such as Microsoft Windows' Front Page and Netscape Navigator's Communicator are better integrated for the within-Web-page collection process than basic text editors or even some word processors. Both applications let users open a blank, editable page into which they can drag content, including images, from the browser to the editor. Users can then edit the collected information in any way they wish. Unlike bookmark managers, the editor page makes all the collected components readily apparent to a user looking at the file. The file can be saved to a server via the editor's integrated FTP support. Users can also access the URL of any collected image. The same cannot be said, however, for any collected text. Unless the URL is specifically grabbed, that information is not captured. Similarly, the user must label the content themselves, since no page information (such as page title) travels with the copied content. Word processors such as Microsoft Word support drag and drop of both text and images from Web pages into files; plain text editors support text capture.

2.3 Hybrids: Spatial Hypertext

In Spatial Hypertext, which predates the emergence of the Web, the notion of the page, *per se,* does not exist. Documents are always already collections of data objects, like one's own notes on a topic, or references to other works. These data objects are manipulated in a 2D visualization space, so that the space in which a user creates a hypertext is also the space in which that document is viewed. This is a more elastic version of hypertext than what the Web currently supports. By way of intermediary, Mark Bernstein's Web Squirrel,[1] is a tool that attempts to bring some of the data object vs. Web page approach to Web practice, though its main use is for annotating bookmarks rather than capturing components within pages. Web Squirrel lets users create and copy information (such as URLs) into a Web Squirrel file. The data is represented as squares to be directly manipulated in a 2D space. The objects can then be arranged and annotated. Agents sift through information in a collection (or "farm" in Squirrel parlance) and suggest connections among collected objects. Like bookmark lists, which only reveal a page title, not the page content, the Web Squirrel boxes hide annotation/link information attached to them. Also, only one box's information can be revealed at a time. As well, while users copy and paste text information from a Web page into Web Squirrel, the source URL for that text is lost unless the user also grabs the URL and drops that into the application. This URL will then show up as a distinct box from the text. Finally, Web Squirrel does not capture images or other media.

2.4 Overview

With the exception of a hybrid tool like Web Squirrel and the Spatial Hypertext work that informs it, Web-based research has focused on managing whole Web pages and sites, rather than on the discrete content within a Web page. Even in Spatial Hypertext with its emphasis on capturing one's own annotations, however, there is little consideration of the interaction of getting content from one context to another. We wish to expand the research to consider this interaction aspect of the movement among information gathering, capture and reflection, and how that can be supported in a web-based approach.

3. Hunter Gatherer Design Process

Our main goal for Hunter Gatherer has been to support the collection making interaction process for collecting within-Web-page components. To determine how best to do this, we carried out the task analysis, tool comparison and an initial prototype design review [13].

3.1 Goals

From our tools and task analysis, and prototype design review, we determined 3 requirements for Hunter Gatherer.

- First, the addition of components to collections must be as transparent as highlighting text.
- Second, the interaction must support user-determined, not tool-forced focus shift among component selection, addition, monitoring, and management.
- Third, the collected components must automatically capture enough contextual information for the collection to be immediately valuable for the user.

In the following sections, we present an overview of the artifact to support this process, and its evaluation in terms of these three goals.

[1] http://www.eastgate.com/squirrel/FAQ.html

3.2 Description of the Tool and Architecture Overview

3.2.1 Browser Integration

Hunter Gatherer is a browser-based, not a stand-alone application. By integrating Hunter Gatherer within the browser in a manner similar to browser support for bookmarking, we are able to minimize the forced divided attention [15] introduced by shifting between one application (the browser) and another (the editor); between information triage and management. Our approach is also proxy based. This means that the user does not have to download additional software to access the tool. While not perfect, the proxy approach also lets us support multiple operating systems and browsers simultaneously. Further, our interest is in the potential impact of supporting within-Web-page collection making on Web information practices. Multiple OS support lets us deploy the tool over a wide user space for this assessment.

3.2.2 Relation to Open Hypermedia

Hunter Gatherer collections are created by rendering references to a collection of addresses for the components within the Web pages. This means that there is no copying of content; only referencing of content addresses. This strategy closely emulates the Open Hypermedia concept of creating collections of smaller-than-page-size elements for what [7] refers to as "pick-up" styled, or arbitrary and user-determined, collections of components. By referencing locations within documents, HG Collections may also be framed in Open Hypermedia terms as user-defined (or user-authored) composites of anchors, as recommended by Halasz, "constructed by reference rather than by value" [8, p355]. We describe the benefits of this approach following an overview of the tool's architecture.

3.3 Basic Architecture

In the current system, once the client browser makes a request for a page, that page is run through a server-side process to convert the HTML to XML-compliant XHTML. Once the page is in XHTML, we can use XML's Document Object Model's tree structure for the document to determine the location of a particular component selected by the user. We have 2 methods to identify components for selection: one is by page element, such as a paragraph, indicated by the XHTML tags like or <td></td>. The second method is to use XML's associated XPath to identify entities *within* elements, so that in some text a user can select, for instance, the last "e" of "some" and the first "t" of text. This latter method emulates the act of highlighting a portion of a Web page for copying. In the current iteration of Hunter Gatherer, we have discovered a number of incompatibilities across systems for within-element text selection, so have temporarily taken this approach off line.

Once a user indicates a selected component is to be added to the collection, the server process either (a) creates a new collection Web-page if one is not already in use or (b) adds the component to the active collection. The component has a default, editable title assigned to it, consisting of the source page's title and a few keywords from the component. We also use the URL part of the component address to create a URL for each component to take the user back to the component's source page. The collection can then be represented as what we call an Aggregated URL. For instance,

http://[server]/examples/servlet/Collection_b?aurl=http%3a%2f%2fwww%2eutoronto%2eca%2fphonebook%2f%23H1%231%234%23Find%20profs%20with%20this...[UofT%20Phone%20Book%20Search]%7chttp%3a%2f%2fwww%2eutoronto%2eca%2fphysical%2ffac%5fserv%2ffacilities%5fsub%2fACentre%2ehtml%23P%234%231%23...[Gym%20Hours]&pagetitle=U%20of%20T%20and%20related%20info

represents an AURL with 2 components, the title of each component is in bold. The final attribute of the URL is the title for the collection itself which will appear in the title for the Web page containing the collection.

Portability. In emailing or otherwise sharing Collection AURLs, each user can view and non-destructively edit the collection, since editing only changes an AURL, and one user's changes to an AURL has no impact on another's.

Dynamic Components. The referenced-based approach to collections makes collections dynamic. If a user includes a component for the local weather, each time the page is loaded, the user will see the latest forecast; reference a bank account balance, it will show up as the current balance. In some cases, it may be necessary to construct methods to let users identify which components are important to be set as static and which can remain dynamic. For now, we are interested particularly in focusing on better understanding the interaction between component selection, capture and management rather than considering the long-term archival properties of a collection. That said, dynamic versus static raises interesting questions about location for static material with respect to where the static material is stored. It is relatively simple to save the collection as a local HTML file that will keep the HTML attributes, like links, in the page alive, but that reintroduces a user-side problem for future retrieval of the file. A server side solution would likely require a network Web disk approach. We are looking into the design of this extension.

Relative Addressing and Bumping. The Document Object Model (DOM) of web pages lets us access locations within a Web page relative to the root of the document. For instance, a page may have two elements after the document root, paragraph A, A , and paragraph B, B . If the user selects paragraph B, we initially used the location of paragraph B in the document tree to create the address for that component in the collection's AURL. This approach had one potential drawback: if an author adds a new paragraph between A and B, the new paragraph becomes B, and the old B becomes paragraph C. We call this effect "bumping." If the user previously collected paragraph B, they would now have the new paragraph B in their collection. Our solution (implemented after our initial field trials) came from the Annotation community [3]. In annotation, one of the goals is to keep an annotation associated with a particular component, even if that component is moved within a document. We have recently adapted Phelps and Wilensky's Robust Intra-document Location algorithms for reattaching annotations to altered components [11] to keep track of "bumped" components. We have yet to formally quantify the success rate of this approach to component tracking, but informally, the technique has proven highly robust and will be part of our Prototype 2 evaluations.

It may be important to note, however, that such robustness is not a key priority for interaction design evaluation. In our field trials, losing components by being "bumped" in this way has not shown up as a concern for users. We do not have enough data yet to know whether or not this is because most collections reflect structurally static pages, so bumping is an infrequent occurrence, or if the collections themselves are being created for shorter term projects, rather than archival purposes, so that if a page changes structurally, users have not encountered this problem showing up in their collections.

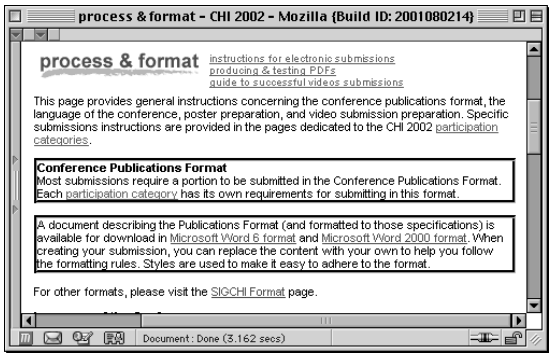

Figure 3. Component Selection. As the user holds the control key and drags the cursor over the page, available components are indicated by borders appearing around them. By holding the control and shift key, users can select multiple components.

Transclusions. By referencing components with AURLs rather than by copying the content, Hunter Gatherer embodies a version of Nelson's Transclusions [10]. Translusions propose creating and publishing hypermedia documents by reference in part so that authors can control both private and public organization and publication of information resources. While the issues of intellectual property raised by letting a user reference parts of a page outside its own (potentially banner-added) context are outside the scope of this paper, one could imagine a method of extending Hunter Gatherer to support authorizing Web sites/pages/components for publication within public or private collections with something like a robots.txt file, or by implementing Nelson's own Transcopyrigtht [10].

4. PROTOTYPES

We now turn to a description of our first alpha-distributed prototype and the evaluation of its interaction.

4.1 First Alpha Prototype

After our initial task analysis, we created a first proof-of-concept prototype to evaluate the concept in a design review with 26 participants [13]. The prototype allowed us to demonstrate the concept of within-page capture as well as the AURL for rendering component collections as new web pages. That prototype relied on the authoring within web pages of specific anchors: if the author had defined a element within the web page and given that ID or Title attribute, Hunter Gatherer could collect the div-wrapped content as components. The results of the design review suggested that we were on the right track with the tool and interaction, but that supporting only author-defined components within web pages would limit the viability of the tool. To address this problem we developed our alpha prototype to support both author-defined and user-determined component selection. We used this first alpha in both lab evaluations and field studies.

4.2 Component Selection in the Alpha Prototype

There are three steps to collect a page component in Hunter Gatherer: (1) select the component to be collected (Figure 3); (2) with that component selected, press the "a" key; (3) a dialog box appears (Figure 4) asking if the user wishes to add the component or not. The user can click "ok" or press the return key to approve the collection. We plan to make this last step part of a user's tool preferences, since in our design reviews, some users wish to be asked to confirm a selection; others did not. The current default is to ask. The user can continue to add components in this manner. Any component that can be displayed in a Web page can be added to a collection, from images to applets.

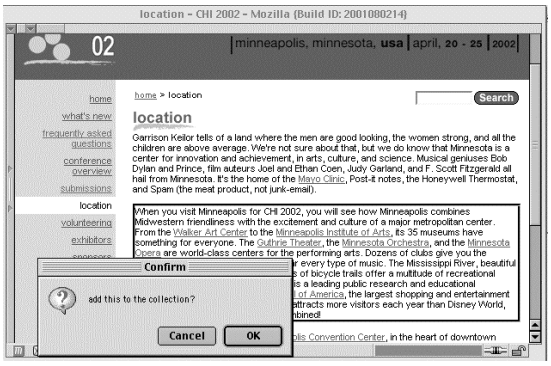

Figure 4. The user has selected a component (indicated by the border around the selection) and hit the "a" key to add the component to a collection. The dialog box then appears to ask the user to confirm the addition.

The selection and add process is relatively transparent. It does not require the user, after selecting a component, to shift attention from the browser to an editor application, paste content into that application's file, go back to the browser, copy the URL, go back to the editor, paste the URL, add a note to contextualize the component, save the file, go back to the browser and refocus on hunting for the next component. The user simply identifies a component to be added; the system automatically adds the component to the collection; creates an editable title for component that, by default, contains the title of the source page of the component. The process also automatically adds the URL as a link back to the source document. By automating these steps, users can focus their main attention on their information gathering task until *they decide* to shift that focus to a different task.

Prototype Selection Note. The visual feedback for selecting a part or parts of a Web page is indicated by borders around elements (Fig. 3) rather than by highlighting. As users, we are used to interpreting

highlighting as something that can be edited to a fine-grained level. Since the first prototype could not support this degree of selection fully, we opted to use borders to indicate what is selectable, since such bounding boxes are less likely to be interpreted as being as refinable as highlighting. In our latest prototype, users can select components down to the level of a character within a word. We will evaluate whether we should keep both modes of selection indicators: highlighting and bounding boxes, or simply use highlighting only.

4.3 Collection Interaction

When the user first selects a component to be added to a collection, a small window, the List/Edit view, opens (Figure 5). This window displays a list of the components in the collection which allows a user to monitor the growth of that collection. As soon as a component is added to a collection, the name of the component is added to the List/Edit view. As a browser window, the List/Edit view can be closed or partially occluded by moving any other window over it, or it can be arranged to be peripherally available as shown in Figure 1 above. Figure 1 shows the List/Edit view visible beside the main browser window.

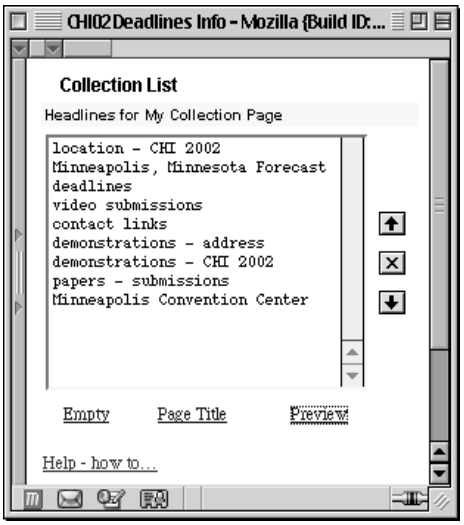

Figure 5. List/Edit View of a Collection. This collection contains components from multiple Web sites. Users can sort, delete or rename components listed, and rename or preview the collection.

The List/Edit view as a separate window lets users determine the degree to which they wish to monitor a collection: each time they add a component after a collection has been initiated, the List/Edit View window does not come to the front, but stays where placed. Indeed, in the first design review of the initial prototype [9], the ability to adjust the "focus" of the List/Edit View to monitor collection state was seen to be an essential feature for the tool. If the user wishes to move task focus from adding components, to the collection, to dealing with the collection itself, they can do so via the List/Edit View. This window for monitoring collection state also acts as the editor palette for the collection. Users have several editing options available: they can rename a component, sort components in the list, delete components from the list, give the collection a title and preview the collection in a browser window.

4.4 Collection View

When the user selects Preview from the List/Edit View, a new browser window opens, displaying each of the components represented by the list, in the order in which they are displayed in that list. With both List/Edit View and Collection View open, as in TopicShop, users have two ways to visualize the collection simultaneously. As shown in Figure 1, right, each component appears with an automatically generated header: the title of the component's source Web page. The component also appears in the Collection with the source page URL as a link. At any time, the user can click that link to open the source page for that component. Likewise, any links within the captured component behave just as they would in the component's source page.

4.5 Gradations of Interaction: Focus

Throughout the collection making process with the prototype, the user can move among hunting for sources, selecting components from those sources, adding those components to a collection, editing the content of a collection, previewing the collection, and saving a version of the collection (by making a bookmark, for instance, of the current collection AURL). If the user at a later point wishes to return to a collection, they load its AURL, which may be done by selecting a bookmark for a collection or by pasting the AURL from an email message into the browser's Location area. To edit the collection further, the user clicks the "edit" link from the collection page, and a List/Edit View window of that collection opens, listing all its components. The user can continue to view or revise that collection. By having all views as browser windows, the user determines which part of the collection making activity they wish to foreground, keep in the background or have peripherally available, simply by arranging the browser's windows.

5. EVALUATION

In order to asses how Hunter Gatherer meets the requirements for collection, focus shift and continued value, we initiated 2 evaluations: an experiment to assess the tool's efficiency and a field study to gain insight into how a new way of working with Web-based information may fit into daily practice. The experiment was designed to assess tool efficiency and effectiveness compared with current best practice: if the tool is not more effective/efficient than existing methods, then there would be little reason for users to adopt the tool. Because we want to deploy the tool widely, the tool must be efficient and robust. The field study, on the other hand, was designed to assess tool *affect* in the context of Web-based information management practices. This largely self-reporting study would be our starting point to understand how to quantify tool use/impact on these information practices.

5.1 Alpha Prototype Experiment

5.1.1 Design and Methodology

We set up a 2x2, within-subjects study to test the efficiency of Hunter Gatherer compared to an editor for creating collections. To reduce learning curve noise in the data for the editor-based collections, we choose Microsoft Word as the most familiar editor among participants. The first factor in the experiment was tool (Hunter Gatherer vs. Word); the second factor was data set (Web pages on a Chemistry program; Web pages on a Physics program).

We first ran a pilot study with five participants, refined the protocol, and ran the formal experiment with 12 participants, representing a mix of technical and non-technical undergraduate and graduate students at the University of Toronto.

At the start of the evaluation, users were given 15 minutes training time with Hunter Gatherer. Users were then asked to build two collections, each from a given set of bookmarks, *to be clear enough to be used by someone else.* This direction was motivation to use the tools' editing capability to create the most effective collection possible within the time constraints. We alternated which tool a participant would use first, Word or Hunter Gatherer. To reduce potential learning effects, we prepared two similar collections of bookmarks, one on the Chemistry program and one on the Physics program at the University. The pages for each set were taken from the same general Web sites, so that pages were similar but for content. So, for each tool, the participant used similarly structured data with distinct content. Participants were given 5 minutes with each set of 3 bookmarks to familiarize themselves with the content of the pages before each tool trial. Participants were then given 15 minutes to build a collection from the bookmarks that would (a) explain how to get a minor in the given subject, (b) list and describe the required courses, and (c) show the course instructors for those course for the term. The experiment let us test HG in terms of our 3 requirements: (1) the efficiency of component addition (2) the effectiveness of HG in the complete collection making cycle (3) the immediate legibility of the resulting collection.

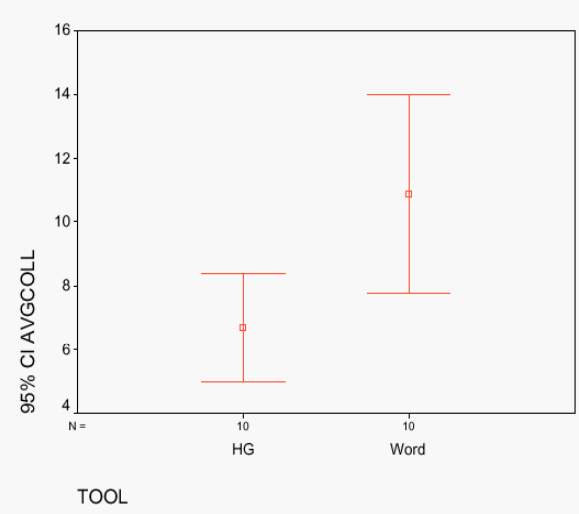

Figure 6. Hunter Gatherer significantly more efficient in component addition than Word.

5.1.2 Empirical Results

A one-way within-subjects ANOVA showed a significant effect of tool type (collection time ($F = 5.730$, $p < .040$) in comparing average component collection time using HG and Word. Participants required an average of 6.70 seconds using HG and an average of 10.9 seconds using Word (Figure 6). The effect of the content variable was non-significant and was pooled over.

5.2 Observations

General Observations. First, despite practice with the Hunter Gatherer tool in which we also demonstrated that each component captured contained a default header and source page URL, only 3 participants, when using Word to build a collection, included the URL of the source page for a given component. The collections, on average, had over a dozen components. The participants who included URLs did so for only a few components, and each of them had used Hunter Gatherer as their first collection making tool.

Word-specific Observations. In creating collections in Word, many participants over-captured the information required from the Web page, and then edited the extra material out from the collection file. Also in editing, Word was more efficient than Hunter Gatherer for revising component headers. Headers in Word could be edited directly in the collection, whereas Hunter Gatherer required participants to move to the List/Edit view to enter a dialog box to make a change. This motivated our revision of the tool to support editing of the component headers directly in the Collection View

HG-specific Observations. In the post-evaluation questionnaire, most users reported that they would prefer highlighting components to collect them, in addition to having the bounding box as methods for component selection. Participants also commented that sorting components in collections was "easier" in Hunter Gatherer than in Word. Similarly, in being asked what the best feature of Hunter Gatherer is, 10 out of 12 participants reported the automatic capture of the component's URL.

5.3 Analysis

We have met our first design requirement to make the addition of a component as efficient as selecting text in a browser. Though participants expressed a desire to have highlighting as a selection method, HG selection performance was significantly better than with Word. The Hunter Gatherer method is also more effective than Word for component addition, since HG automatically adds both a header for the component and the URL for the source page, the latter addition indicated by users as the most valuable attribute of the tool.

The alpha prototype only partially met our second design requirement to support user-determined focus shift among collection tasks. Header editing in the prototype forced users to concentrate on the tool, rather than the task: double clicking a header title in List/Edit view and ok'ing a change in a header dialog box is less transparent than editing the header directly in a file. Our observations also indicate that users want to be able to make a first pass at component selection, and then edit the components further, after they have been collected. We may be requiring users to focus too much on precise component selection when they would rather be focusing on faster, more general initial "information triage" as described above, and edit further, later.

With the automatic capture of the URL, the prototype partially met our third design requirement for automatic capture of enough information for the collection to continue to be useful to the user. Our initial evaluation showed us that the headers, which consisted of only the source page title, were not descriptive enough to be

automatically useful if many components came from the same page(s), which was the case in our trials. This lack of distinction in the List/Edit view rendered parts of the collection immediately less useful.

We describe the second prototype resulting from this analysis in the section Prototype 2, below. First, we describe the field study that went on in parallel with the lab evaluation of the first prototype.

5.4 Field Study

5.4.1 Design and Methodology

Andrew Dillon, discussing Process, Outcome and Affect as alternative evaluation measures to Effect, Efficiency and Satisfaction, suggests that the affect of a design – whether a user experiences the interaction as empowering or frustrating – is critical for understanding and improving the interaction design [6]. With Hunter Gatherer, we are interested to know if, given an efficient and effective interaction, the tool itself will become an affective part of Web information practices. To begin to answer this question, we have followed Dillon's suggestions for evaluating affect: we have given the tool to participants to explore "free style." Participants in the four week field study were asked to try the tool over a month, to answer a weekly set of questions about their tool use, and to share example collections made during that time. Participants learned how to use the tool via a Web page, describing its features and known bugs. There were 14 participants from a wide variety of disciplines. Each identified themselves as "tolerant" of alpha software and expressed an interest in using the tool.

5.4.2 Overview of Results and Analysis

All but one participant reported that they like the tool and make collections with it. Participants tended to make specific collections for specific purposes, rather than making more casual, serendipitous collections. In other words, preliminary findings indicate that users form the intent to collect first, and then use the tool, rather than using the tool for more casual perusal. We do not have the data to say whether this deliberate approach is because of any particular attributes of the tool, or due to preconceptions about information gathering. Kinds of collections made, however, were diverse. One participant made a collection of components from a variety of financial information sites, which, he recorded, he consults daily, since the components are dynamic and he wants only current financial information. Another participant in Medicine has a collection on a particular disease profile that he wishes to publish for participants at an upcoming conference. Another has gathered components for course lecture notes. The intent of these collections suggests that they will have a relatively long shelf life and possibly high return use. Only one participant, a reporter for a national television network, indicated being interested in making collections for shorter term purposes such as collecting background sources for upcoming stories. We will quantify both collection making and return to collection rates in the follow-up study.

When asked specifically if the concept of making collections from within-Web-pages had become a technique that was now part of their way of thinking about managing Web-based information or not, most users responded that the tool/concept had indeed become part of their way of thinking about gathering information on the Web. Only one participant reported discovering that he did not find a need to make within-Web-page collections. Indeed, many of the participants regularly emailed design suggestions that would make the tool more effective for them, most of which reinforced our findings in the lab evaluations, such as: more descriptive default headers, fewer steps to edit those headers and highlighting as well as bounding boxes for component selection.

Surprisingly, we did not hear any concern about several parts of the prototype we had anticipated: bumping/dynamic components and collection layout. In the first case, as described in the Architecture section above, if an author adds a component before a component previously collected, that will offset which component is represented in the collection. Our first prototype, used in the field trial, did not have the Robust Linking adaptation to manage bumped material. No one, however, reported having had this experience of content being bumped. Similarly, no one when asked reported having problems with dynamic vs. static components. That is, participants seemed to understand that components in a collection behaved similarly to bookmarks: material referenced may change. Indeed, so far it has only been participants in design discussions who have suggested that we must support "saving static versions of a collection." While this feature seems to make sense intuitively, it is not a feature that any of our field trial users requested. That said, our second prototype does have a "save" option to save a static version of a collection as a fixed HTML page. Our next evaluation will instrument this feature so that we can measure frequency of use. Further, no one reported wanting to be able to resize or reposition components in the collection view. One participant in particular said that he knew what was in a collection – he'd made it – and he could use the Find command in the browser to locate an element quickly in a collection, if required.

5.5 Prototype Two

Based on the formal experiment and field study, we have created a second prototype with the following revisions:

- *default headers:* the default header created by the collection operation now creates headers for components that contain both keywords from the component selection and the title of the component's source page. This means that components from within the same page are distinguishable from each other.
- *direct editing of headers:* users can now double click on a header directly in the Collection page to edit the header.
- *post collection component deletion:* users can now over-capture components (for instance, grab three paragraphs rather than 1 paragraph) and then, at their discretion, delete unnecessary elements within a component from within the collection view.
- *single component viewing:* by shift-clicking on a component in the List/Edit view, a user can render that single component without having to render the entire collection.

Overall, we anticipate that these revisions will bring Hunter Gatherer closer to supporting our design requirements for

transparent collection making. Informally, the majority of our field study participants have continued to use the tool and are now using the second prototype. The response has been favourable, with some participants suggesting that the changes have both improved interaction with the tool and (perhaps consequently) increased tool use.

6. Conclusions and Future Work

6.1 Conclusions

In this paper, we discussed the problem of selecting and managing smaller-than-page-sized information components when interacting with Web-based documents. We presented results from needs assessment and tools analysis, which showed that current tools either over-capture the desired information, or require users to divide their attention between knowledge discovery and knowledge management, and that this divided attention can compromise performance of either task. To begin addressing this problem, we presented Hunter Gatherer, a tool and architecture to support Web-based, within-page component collections. We presented how the tool supports collection interaction. The tool supports browsing, sorting, addition and deletion of components. Each component also has a link back to its source document for reference. Since collection information is stored in its URL, collections can easily be shared and retrieved. We then presented results of formal lab experiments and field studies which have helped us improve the tool. We will re-evaluate the new prototype to confirm whether or not these design revisions have indeed improved tool efficiency and efficacy. The previous field trial was too short to let us know if tool use had a significant effect on the perception of the web page as a less fixed information source as opposed to one in which the user could determine context for presenting information. We are designing a new protocol to use with our revised tool in a longer and larger field study to see if we can better capture tool effect.

6.2 Future Work

Context and Navigation. While our design motivation for Hunter Gatherer has been to support fine-grained component collection, a side effect of this work has become of increasing interest. This is the effect of collection making on navigation. Collection making foregrounds the idea that users can focus their attention on knowledge discovery rather than on both location tracking and information management of selected information. Since the components in collections have back links to their source documents, the collection becomes a loose map for the user-constructed hyperspace represented by the collection. Thus, a user can potentially refer to the collection to re-situate themselves in a previously discovered context of interest. We are looking at methods to strengthen this process of associative navigation with Hunter Gatherer. One of these approaches we refer to as Back++.

Back++: History in Context. In observing participants use Hunter Gatherer, both in the lab and in hands-on demonstrations, we discovered that our tool may still be too course-grained to support the collection-making process optimally. Our tool is binary about selections: an element is either part of a collection or it is not, just as a bookmark is either added to a list or it is not.

We are just completing development of a prototype that will allow users to indicate a "maybe" state for a selected element, as well as support multiple views of the collection. These views would allow users to view the "definite includes," definite includes with maybes, and definite includes, maybes and anything else viewed on that path (effectively, the history list) – or any combination of these views. At any time components can be moved among categories of definite, maybe or part of history. Pages will also show up as frequently as they are visited: for instance, page A visited before collected component Y shows up before component Y in the list, and then shows up again after collected component Z, if it was visited after Y. Because of this integrated view, the path of the user's travels in creating the collection will be immediately apparent. These views can be rendered in list form in the List/Edit view, or rendered completely in the Collection view. What we wish to evaluate two effects in particular with this extension to the HG tool: (1) do these views create an automatic, transparent and associative map of one's information gathering sessions, and if so, in what contexts is this especially useful? and (2) does the support for "maybe" improve the value of the tool by lessening the need to make a seeming "absolute" decision at selection time, or does it have the opposite effect, introducing more decisions a user feels the need to make during information triage? We look forward to reporting the results of these tests.

Web-Testing Hypermedia Assumptions. The Web, besides becoming the default information resource, is a powerful test bed for understanding and evaluating information interaction design. By embodying attributes like composites from Open Hypermedia in browser-based systems, we have the opportunity to test Hypermedia concepts like the user as simultaneous Reader/Editor/Author.

6.3 Overall

Our first studies with Hunter Gatherer have helped us both to improve the tool's efficiency, as well as to understand better users' expectations for Web-based information interaction support. These studies have already shown us that the required tasks may be more subtle in the both the capture and reflection processes (*do I want to keep this? Maybe. Show me versions of the collection based on scenarios of definites, maybes, everything)* and more tolerant in the representation process *(layout refinements are largely unneeded)* than anticipated. Consistent with our early hypothesis however, is the observation that users do find value in being able to create their own information contexts (like collections) for information access and for reflection on that information, especially when support for this process is well-integrated with the browsing process.

7. ACKNOWLEDGEMENTS

Thanks to Adele Newton, Michael Milton and Tim Dinesen for their support through the Bell University Labs funding program. Thanks to Graeme Hirst and Alberto Mendelzon and Peter Nürnberg for their feedback throughout the project's development, and to Steve Feiner, Dan Olsen, and Kelly Booth for their feedback at UIST2001's open demo sessions. Thanks to the participants in our evaluations, as well as to those who reviewed, commented on and copy-edited iterations of this paper. The work is better for your insights.

8. REFERENCES

[1] Abrams, D., Baecker, R., Chignell, M. Information archiving with bookmarks: personal Web space construction and organization in Conf. Proc. on Human Factors in Computing Systems, 1998, pp. 41–48.

[2] Amento, B., Terveen, L., Hill, W., Hix, D. TopicShop, enhanced support for evaluating and organizing collections of Web sites in Proc. of the 13th Annual ACM symposium on User interface software and technology, 2000, 201–209.

[3] Brush, A.J., Bargeron, D., Gupta, A., and Cadiz, J.J.. Robust Annotation Positioning in Digital Documents. Proc. CHI 2001 (Seattle, WA, 2001), ACM Press, 285-292.

[4] Card, S. K., Robertson, G. G., York, W. The WebBook and the Web Forager: an Information Workspace for the World Wide Web. Conf. Proc. on Human Factors in Computing Systems (Vancouver, Canada, April 13–18, 1996), 111.

[5] Czerwinski, M., van Dantzich, M., Robertson, G.G., Hoffman, H. The contribution of thumbnail image, mouse-over text and spatial location memory to Web page retrieval in 3D. Sasse A. & Johnson, C., Eds. HCI–Proc of Interact '99 (Edinburgh, Scotland, 1999), IOS Press, 163-170.

[6] Dillon, A. Beyond Usability: Process, Outcome and Affect in Human-Computer Interaction. Presentation to Faculty of Information Studies University of Toronto, 23/03/01.

[7] Garzotto, Franca, Mainetti, Luca, Paolini, Paolo. Adding Multimedia Collections to the Dexter Model. Conference on Hypertext and Hypermedia (Edinburgh Scotland, Sept. 19–23, 1994), 70-80.

[8] Halasz, Frank G. Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems in Proceeding of the ACM conference on Hypertext, 1987, pp. 345–365.

[9] Marshall, C. C., Shipman, F. M., Coombs, J. H. VIKI: Spatial Hypertext Supporting Emergent Structure. Proceedings of the 1994 ACM European Conf on Hypermedia Technology, 1994, 13–23.

[10] Nelson, T. H. "A Literary Structure with Two Fundamentally Different Means of Connection." Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning and Deep Re-Use. http://www.sfc.keio.ac.jp/%7Eted/XUsurvey/xuDation.html.

[11] Phelps, T., and Wilensky R. Robust Intra-document Locations, Proc. of the 9th World Wide Web Conference, (Amsterdam, May 2000).

[12] Robertson, G., Czerwinski, M., Larson, K. Robbins, D., Thiel, D., van Dantzich, M. Data Mountain: Using Spatial Memory for Document Management. Proc. of UIST '98, 11th Ann. Sym. on User Interface Software and Technology, ACM Press, 153–162.

[13] schraefel, m.c. and Zhu, Yuxiang. Preliminary Requirements Gathering for the Design of User-determined, Within-page, Web-based Collections. Tech Report, CSRG-433, DCS, U of Toronto, 2001.

[14] Terveen, L., Hill, W., Amento, B. Constructing, Organizing, and Visualizing Collections of Topically Related Web Resources. ACM Trans. Comput-Hum. Interact. 6, 1 (Mar. 1999), 67–94.

[15] Wickens, C. D., Hollands, J. D. Engineering Psychology and Human Performance, 3rd Ed., Prentice Hall, 2000.