March 1, 1995

SRC Research Report 135a

DeckScape: An Experimental Web Browser

Marc H. Brown and Robert A. Shillner

digital

Systems Research Center
130 Lytton Avenue
Palo Alto, California 94301

Systems Research Center

The charter of SRC is to advance both the state of knowledge and the state of the art in computer systems. From our establishment in 1984, we have performed basic and applied research to support Digital's business objectives. Our current work includes exploring distributed personal computing on multiple platforms, networking, programming technology, system modelling and management techniques, and selected applications.

Our strategy is to test the technical and practical value of our ideas by building hardware and software prototypes and using them as daily tools. Interesting systems are too complex to be evaluated solely in the abstract; extended use allows us to investigate their properties in depth. This experience is useful in the short term in refining our designs, and invaluable in the long term in advancing our knowledge. Most of the major advances in information systems have come through this strategy, including personal computing, distributed systems, and the Internet.

We also perform complementary work of a more mathematical flavor. Some of it is in established fields of theoretical computer science, such as the analysis of algorithms, computational geometry, and logics of programming. Other work explores new ground motivated by problems that arise in our systems research.

We have a strong commitment to communicating our results; exposing and testing our ideas in the research and development communities leads to improved understanding. Our research report series supplements publication in professional journals and conferences. We seek users for our prototype systems among those with whom we have common interests, and we encourage collaboration with university researchers.

Robert W. Taylor, Director

DeckScape: An Experimental Web Browser

Marc H. Brown and Robert A. Shillner

March 1, 1995

Publication History

This report appears in the proceedings of the *Third International World-Wide Web Conference*, held in Darmstadt, Germany, April 1995, published by Elsevier as a special issue of **COMPUTER NETWORKS AND ISDN SYSTEMS**.

A two-page summary of the work described in this report appears in the Conference Companion proceedings of the ACM *1995 Conference on Human Factors in Computing Systems (CHI'95)*, held in Denver, May 1995.

Author Affiliation

Rob Shillner is currently a Ph.D. candidate at Princeton University. The work described here was performed while he was a research intern at SRC, during the summer of 1994. Rob's email is `ras@cs.princeton.edu`, and the URL of his home page is `http://www.cs.princeton.edu/~ras/`.

Abstract

This report describes DeckScape, an experimental World-Wide Web browser based on a *deck* metaphor. A deck consists of a collection of Web pages, and multiple decks can exist on the screen at once. As the user traverses links, new pages appear on top of the current deck. Retrievals are done using a background thread, so all visible pages in any deck are active at all times. Users can move and copy pages between decks, and decks can be used as a general-purpose way to organize material, such as hotlists, query results, and breadth-first expansions.

Overview

Mosaic [1] and the various Web browsers it has inspired [5][6], use a depth-first navigational model. At any point in time, the user is “at” a particular node on the Web, having arrived there by following a path of nodes from some root. The user can choose to leave the current node either by following an outgoing link or by going back to the previous node in the path from the root. After going back, the user can also choose to go forward to the next node on the most recent path from the root.

Most Mosaic-inspired browsers support other navigation methods in addition to these primitives; for instance, the user can jump to different URLs using the “Hotlist” and “Open URL” dialogs. Most browsers also offer multiple open windows, each with its own depth-first visitation stack. However, with the exception of Netscape [4] and InternetWorks [3], the browsers are single-threaded, so while one window is downloading a page, all of the windows owned by the browser become inactive.

We have developed DeckScape, an experimental browser for exploring new methods of navigating and organizing pages on the Web. DeckScape centers on the metaphor of a *deck*: a collection of Web pages, of which only one is visible at a time. When the user clicks a link on a page, a new Web page appears on top of the deck, obscuring the page that was previously visible. The user can leaf through a deck’s pages one at a time, jump to the top or bottom of a deck, or move to any particular page by choosing its name from a list of the deck’s current contents. The browser itself consists of multiple decks, all in a single top-level window. Users can move, resize or iconify decks, move or copy pages between decks, start new decks, delete decks or pages, and so on. The contents of decks persist between invocations of DeckScape.

The key benefit of the deck abstraction is that it provides a way to organize material. For example, a user can keep the home pages of all of his or her colleagues together in a deck named “Colleagues,” or keep several Mosaic-style hotlists, each in its own deck. DeckScape further uses decks to return the results of certain operations, such as “expand all the links on this page.”

DeckScape is also multi-threaded. In particular, fetching a new page occurs in the background, in a separate thread. Thus, unlike single-threaded browsers, traversing a slow link or downloading a large file does not freeze the entire application. All decks remain active and ready for browsing, and multiple links can be traversed concurrently.

(Of course, one could start up multiple instances of a single-threaded browser. This would have the advantage that, when traversing a slow link or downloading a large file, the other instances would remain active. However, multiple instances have the drawback of increasing the amount of computer resources consumed. Also, multiple instances are completely independent of each other, so it is not possibile to share the information among instances.)

A Tour of DeckScape

Deck Basics

When DeckScape is first run, the user sees a window containing a menu bar and a large open area. This window forms the *workspace* in which the user positions decks and pages.

Choosing “New Deck” from the File menu produces a new deck with a default name, containing a single document: the user’s home page. Clicking links adds more pages to the deck. Buttons let the user shuffle through the deck’s contents or go to the top or bottom of the deck. The user can create more decks and use them to follow different links. This behavior is similar to that of a traditional browser: creating windows, clicking links and moving forward and back.

Fig. 1 shows DeckScape with a single deck. The deck, named “WWW’95”, has six pages in it. The user is looking at the second page in the deck, whose URL is `http://www.igd.fhg.de/www/www95/program.html`.

DeckScape retains all pages until the user explicitly discards them, while a traditional browser retains only those pages on the path from the root to the current page. For example, if a user starts at page A, then traverses some pages (including B) and ends at C, both DeckScape and a conventional browser keep copies of all the pages from A to C. However, if the user then backs up to B and chooses a new link, a traditional browser discards all of the pages after B up to and including C. On the other hand, DeckScape keeps all of those pages and inserts the new page into the deck just after B.

This design allows users to quickly switch back and forth between two or more pages which do not lie on one convenient path from the root, but rather lie on different branches of a tree. Traditional browsers would have to download and parse each page on each traversal, while DeckScape allows the user to flip quickly through the deck’s contents without refetching any pages. (DeckScape has a “Reload” command to refetch a page, rather than use the page from its internal cache.)

Once a deck has been built up, a user can modify the contents of a deck in several ways. Clicking a document’s “D” button deletes the document from its deck. Dragging a document from one deck to another moves the document between decks. Holding down an appropriate modifier key while dragging copies the document.

Figure 1: DeckScape with a single deck.

Clicking a deck's "Merge" button and dragging to another deck merges two decks by adding all of the pages from the first deck into the second one, immediately after the second deck's current page.

"Away" Pages

DeckScape also offers the ability to temporarily remove a page from its home deck. To pull a page away from a deck, the user drags the page into the workspace background; the page then appears in a window separate from its home deck. The page is still a member of the deck, but it is *away* from the deck rather than *in* it. Later, the user can issue the deck's "Gather Up" command to bring the "away" pages back to the deck, or he or she can drag the pages back to the home deck (or a different deck) manually.

The ability to pull a page away from its deck allows the user to simultaneously view two or more pages from the same deck. It is often useful to have certain pages, such as glossaries or reference pages, visible for an extended period, even while following another chain of links. DeckScape allows users to drag such pages off to the side and continue following links on the main body of the deck, leaving the "special" pages easily accessible.

Fig. 2 shows DeckScape with three decks, "Ongoing SRC Research Projects," "Nifty home pages," and "Palo Alto stuff." The narrow window in the lower-left is showing an "away" page from the "Palo Alto stuff" deck:

When the user follows a link on an "away" page, the resulting new page appears on the main body of the deck. This behavior is useful when one page, such as a table of contents, has many links in which the user is interested. Ordinarily, clicking a link on such a page would cause a new page to cover up the table of contents, so that the user must dig through the deck each time he or she wishes to follow a new link from the table of contents. However, if the user were to pull the table of contents away from the deck, then the page would always be handy for following new links: the resulting pages appear on the deck and do not cover up the table of contents.

Similarly, the user could click many links on an "away" page in rapid succession, causing many new documents to appear on the deck. Since DeckScape is multi-threaded, the user need not wait for one download to complete before clicking another link. The user can browse through the resulting new documents immediately, even before all of the downloads have completed. No traditional browser can support this type of Web exploration because in a traditional browser clicking a link always makes a new document cover up the page containing the link.

Browsing in this manner is particularly effective when combined with a deck's "Make Link Index" command. This command finds all of the links on each page in

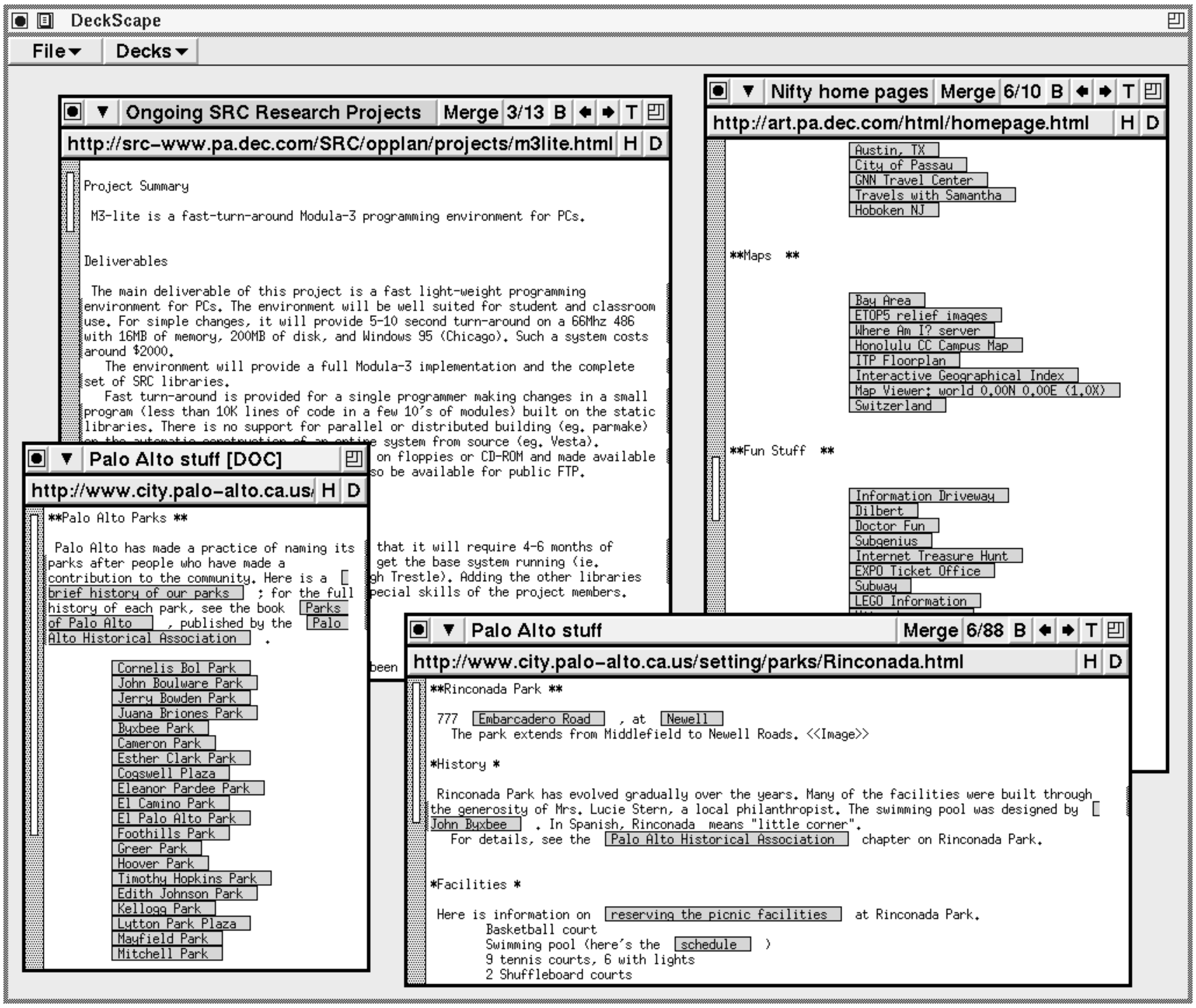

Figure 2: DeckScape with three decks, one of which has an “away” page.

a deck, then adds to the deck a new page containing all of the links in alphabetical order. The new page provides an index of all the links accessible from any page in the deck. The user can then drag the index page away from the deck and click a series of links, browsing through the resulting documents as they appear on the main body of the deck.

Another use for “away” pages is for creating new decks. If a user wants to start a new deck from a particular page, he or she can drag the page away from its home deck, then issue the page’s “Make New Deck” command. A new deck appears, containing the page.

Organizing Information with Decks

Decks can be used to organize information found in the Web. Since decks’ contents are automatically saved and restored when the DeckScape application exits and restarts, users can use decks to help find pages that they have visited before. For instance, if a user frequently visits a particular Web server, he or she can set up a deck to contain pages from that site, and use the deck to access the site, rather than follow a series of links from a home page.

Another use of decks is to organize hotlists. DeckScape has a special hotlist deck, and each document has a “copy to hotlist” button. When a user comes across an interesting page, he or she can click the “H” button to copy the page into the hotlist deck. Users can also use ordinary decks as hotlists by manually copying interesting documents into them; thus, each user can have many hotlists, organized by whatever criteria are appropriate.

In Fig. 3, the hotlist deck is in the lower center. The hotlist is like any other deck, except that it cannot be renamed or deleted by the user. The “Goto Page...” dialog allows the user to select and jump to any document in a deck, either by its title or by its index.

Acquiring Information with Decks

DeckScape also uses decks to return the results of certain operations. For example, issuing a page’s “Expand One Level” command causes DeckScape to traverse each link on the page and place the resulting documents in a new deck. The link traversals all occur in the background, so the user retains control of the application and can even begin browsing pages and traversing links in the new deck before all of the pages have been fetched.

In Fig. 4, the deck entitled “Expanding 82 links” is the created by issuing the “Expand One Level” command on the home page for the Systems Research Center

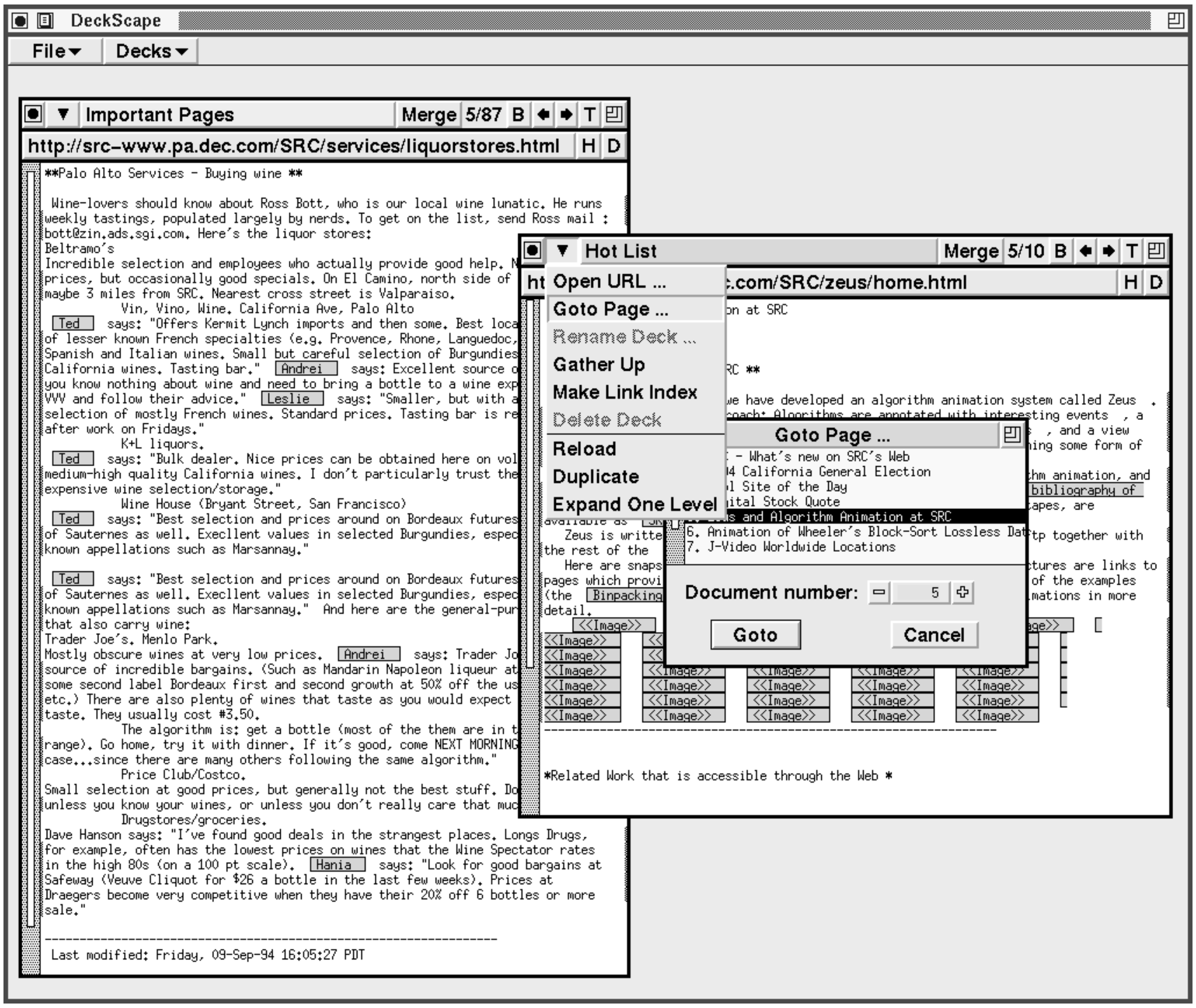

Figure 3: Hotlists in DeckScape.

(the page displayed in the "Deck 12" deck). It is common to issue the "Expand One Level" command on the home page of Web sites in order to create a new deck that contains the pages most relevant to the home page. These pages can then be rapidly traversed with a single button. When the screen dump was taken, 79 of the 82 links had been retrieved, and the user was looking at the 24th page in the deck.

Global search is another operation which returns its results in a deck. After the user enters the text to be found, DeckScape searches through all the pages in each deck. It copies the pages that contain hits and makes a new deck containing the copied pages.

In Fig. 5, the deck labeled "Search Results" contains a copy of each page from any deck matching the string "animation."

Implementation

DeckScape is implemented in Modula-3 [2] and consists of about 3500 lines of code. The system makes extensive use of Modula-3's standard libraries, including the threads package, user interface toolkit, and persistent data structures facility.

DeckScape's primary components are object classes that correspond to portions of the visual interface; these visual classes have names ending in "VBT". The VBT classes, along with other non-visual classes, constitute a hierarchy of abstractions which make it possible to easily integrate the browser's functionality into other Modula-3 applications.

The remainder of this section describes the modules comprising the implementation.

- A `WorkspaceVBT` is the main application window; only one is ever created and it is installed in a top-level window. A `WorkspaceVBT` provides the global menu bar and the space where the user positions decks and documents. The `WorkspaceVBT` maintains lists of all of the decks and "away" documents.

- A `WSObjectVBT` is an abstract class whose subtypes are objects that can appear in the workspace, namely decks and "away" documents. No objects of type `WSObjectVBT` are ever created; `WSObjectVBT` exists so that operations that are common to both decks and "away" documents (such as iconify, raise and lower) can be defined.

- A `DeckVBT` (a subclass of `WSObject`) is a deck: it contains the deck's title bar, browsing controls, sizing, iconifying and dragging widgets, and a menu of miscellaneous commands, as well as space for displaying a document.

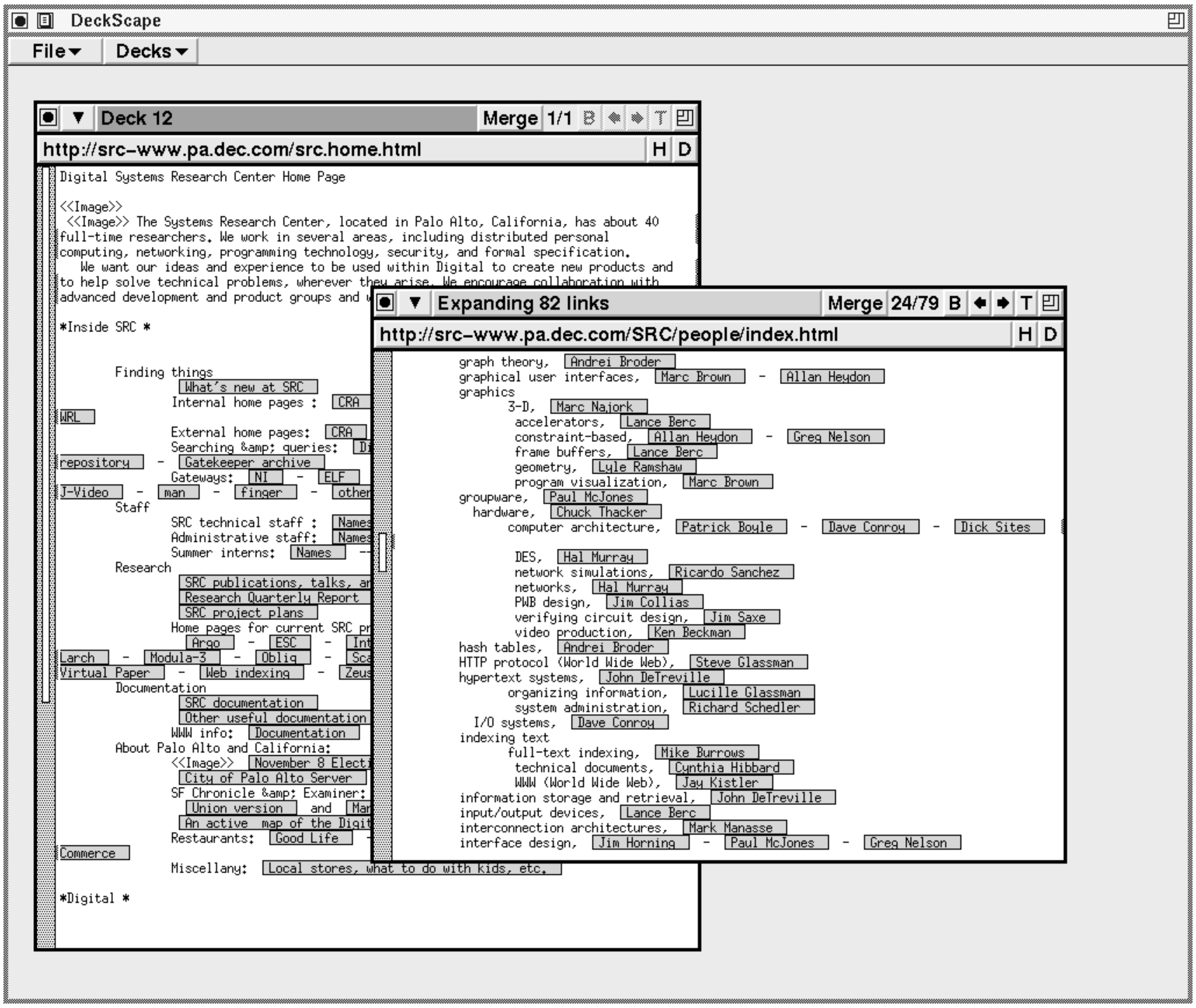

Figure 4: DeckScape's "Expand One Level" command.

- An `AwayVBT` (a subclass of `WSObject`) represents a document away from its deck: it contains a title bar for repositioning the “away” document, sizing, iconifying and dragging widgets, and a menu of miscellaneous commands, as well as a space for displaying a document.

- A `DocVBT` displays a Web page. It contains the buttons to delete itself or copy itself to the hotlist, as well as the draggable banner used to move the document between decks or “away” from its home deck. The `DocVBT` also contains space for the document’s contents, displayed by a `URLVBT`.

- A `URLVBT` is an abstract class for displaying data fetched from a URL. At present, only two subtypes are defined: one to display plain text (called a `PlainVBT`) and one to display HTML (called an `HTMLVBT`).

- A `PlainVBT` displays a plain text document.

- An `HTMLVBT` displays the contents of an HTML page. It allows the user to scroll through the page and traverse links by clicking. `HTMLVBT` is still in the very early prototype stage; it does not support multiple fonts, sizes and styles, nor does it support inline images and forms.

- An `HTML` object is an abstract syntax tree for an HTML document. `HTML` objects are produced by the `Parser` module and used by `HTMLVBT` objects.

- The `Web` module fetches a document from a given URL.

- The `Parser` and `Lexer` modules produce `HTML` syntax trees from HTML source text retrieved by `Web`.

Conclusions

This report has described DeckScape, an experimental Web browser. Decks provide a flexible way to organize Web pages, in many of the same ways that modern folder-based mail readers (e.g., `xmh` for Unix) improve on previous generation tty-oriented mail programs (e.g., `/usr/ucb/mail` for Unix). However, DeckScape is lacking essential Web-browsing features such as inlined images, forms, multiple fonts, and external viewers. Some of these deficiencies will be easy to address, but others will take quite a bit of effort. Because of these deficiencies, DeckScape is not in daily use, even for the authors.

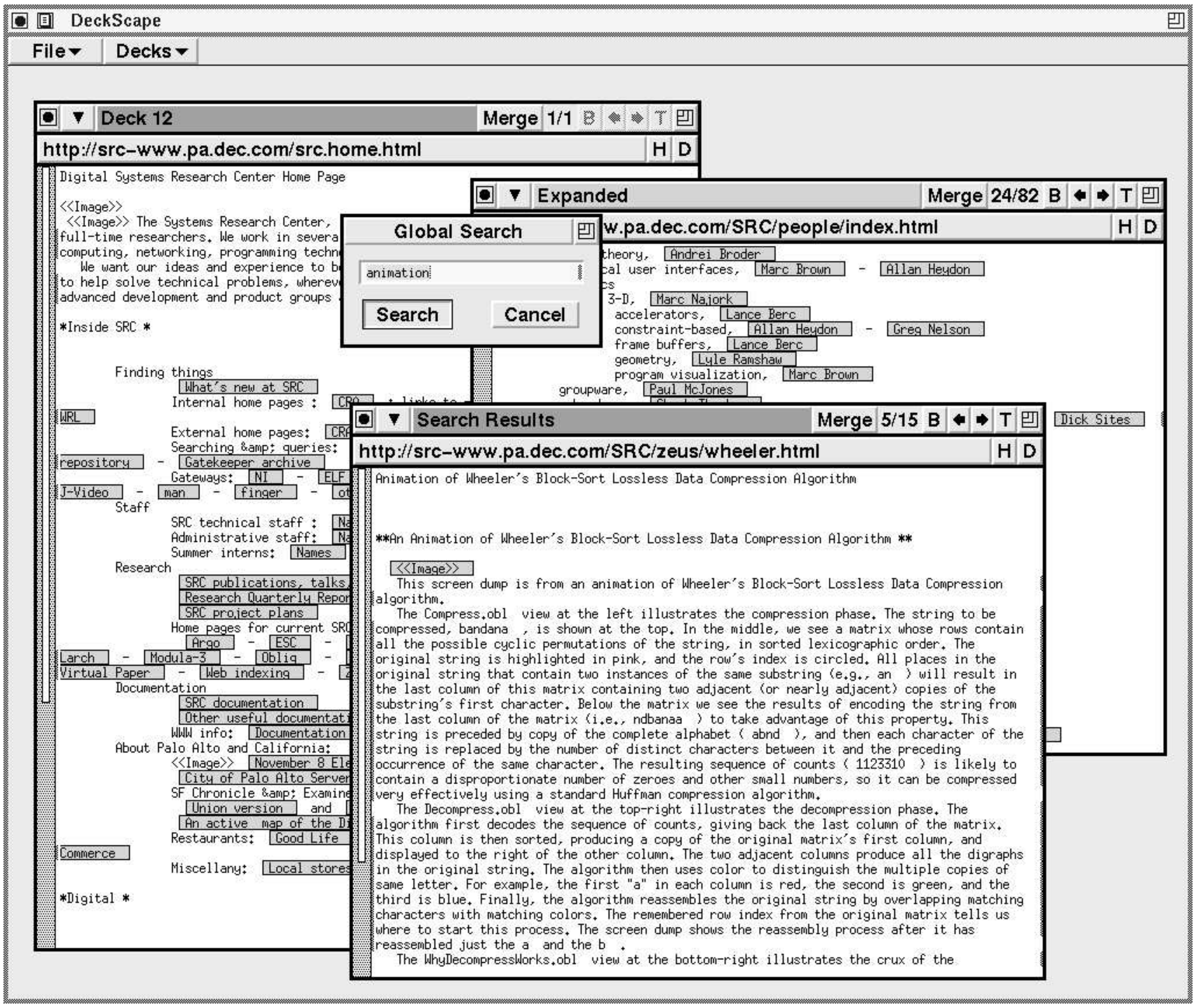

Figure 5: The results of a search command are returned as a deck.

We do not claim that DeckScape is *the* correct way to browse the Web and organize pages in the Web. Discovering and quantifying the strengths and weaknesses of decks are challenges for the future. We hope that the ideas introduced in this report will help to advance the standard for navigational and organizational capabilities of Web browsers.

Acknowledgments

Bill Weihl and Paul McJones helped improve the clarity of this report. Lucille Glassman implemented the `Web` module. Steve Feiner provided a wealth of ideas for improving the system, most of which are still on our "todo" list unfortunately.

References

[1] About NCSA Mosaic for the X Window System.
`http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/help-about.html`

[2] Modula-3 home page.
`http://www.research.digital.com/SRC/modula-3/html/home.html`

[3] Welcome to BookLink.
`http://www.booklink.com/`

[4] Welcome to Netscape!
`http://home.mcom.com/home/welcome.html`

[5] World Wide Web Frequently Asked Questions.
`http://sunsite.unc.edu/boutell/faq/www_faq.html`

[6] WWW Client Software products.
`http://info.cern.ch/hypertext/WWW/Clients.html`