From Desktop To Phonetop: A UI For Web Interaction On Very Small Devices

Jonathan Trevor, David M. Hilbert, Bill N. Schilit
Fuji-Xerox Palo Alto Laboratory
3400 Hillview Avenue
Palo Alto, CA 94304 USA
Tel: +1 650 813 7220
E-mail: {lastname}@pal.xerox.com

Tzu Khiau Koh
Xerox Singapore Software Center
16 Science Park Drive #02-04
The Pasteur, Singapore 118227

E-mail: kohtk@pal.xerox.com

ABSTRACT

While it is generally accepted that new Internet terminals should leverage the installed base of Web content and services, the differences between desktop computers and very small devices makes this challenging. Indeed, the browser interaction model has evolved on desktop computers having a unique combination of user interface (large display, keyboard, pointing device), hardware, and networking capabilities. In contrast, Internet enabled cell phones, typically with 3-10 lines of text, sacrifice usability as Web terminals in favor of portability and other functions. Based on our earlier experiences building and using a Web browser for small devices we propose a new UI that splits apart the integrated activities of link following and reading into separate modes: *navigating* to; and *acting* on web content. This interaction technique for very small devices is both simpler for navigating and allows users to do more than just read. The M-Links system incorporates modal browsing interaction and addresses a number of associated problems. We have built our system with an emphasis on simplicity and user extensibility and describe the design, implementation and evolution of the user interface.

KEYWORDS: Web browsing, wireless web, web phone, PDA, transducing, transcoding.

INTRODUCTION

Today most people interact with the Internet using a desktop computer running a Web browser such as Internet Explorer or Netscape. Their interaction involves downloading and viewing HTML documents that include content (text, images, and user interface components) as well as links to other Web content (including HTML, audio, video, Adobe PDF, and Microsoft Office files). The user, viewing both content and links, will rapidly alternate between reading and link following. When a user clicks on a link to a non-HTML document, the browser invokes a client-side plug-in application to display and in some cases allow manipulation of the link content.

Although the browser interaction model is well suited for the desktop computer, it is not well suited for "phonetop" computers. This is because the usability of the browser model depends on characteristics of desktop PCs, such as their large user interfaces (displays, keyboards, pointing devices), considerable computing resources (CPU, storage, operating systems), and high bandwidth network connectivity. Small devices possess few of these characteristics. Nevertheless, small portable Internet devices are growing in popularity as Web terminals, and users expect to be able to access and interact with the same rich multi-media content available at their desktops.

Thus, designers must consider how to bridge the feature gap between desktops and very small devices. There are a number of methods available for putting web content onto small displays. For the most part, these rely on the notion of *transducing* (i.e., converting and transforming) Web content to fit the small UI. It is our experience with building and using an early transducing system, along with observations of colleagues' browsing activities, which led us to develop a complementary technique. Our approach is to not only transform the content, but also to transform the integrated activity of Web browsing into separate and simpler modes.

We begin by summarizing the common methods for putting web content onto small devices and introduce the technique of modalized web browsing. We then describe how our system generates a *navigation interface* and an *action interface* for a range of very small devices. We conclude with our experience using the system and plans for future work.

PUTTING WEB CONTENT ONTO SMALL DISPLAYS

Mobile web devices span a range of capabilities as shown in Figure 1. The largest displays can be found on PDA class devices, like the Palm or Pocket PC, which are capable of displaying many lines of text and graphics in a single screen. The mid-range displays, such as the NeoPoint 1000, can show around 10 short lines of text with more limited graphics capability. At the lower end are phones like the

Samsung SCH-3500 with 4 lines of text and a maximum of 48 characters in total. We characterize very small devices as being sub-PDA in size, at the middle or low end of this range.

Make Model	Network	Markup	Screen Size (WxH)	Dimensions (WxHxD)
Palm Pilot VII		HTML Gray	160x160 pixels	190g 133x83x19mm
RIM 950	Mobitex	WML Gray	132x65 pixels	142g 63x89x23mm
NeoPoint NP1000	CDMA PCS	HDML WML	120x160 pixels 11x24 chars	181g 140x54x25mm
NEC N209i	TDMA	CHTML Gray	108x82 pixels 9x6 chars	86g 90x46x19mm
Mitsubishi D209i	TDMA	CHTML Color	96x90 pixels 8x7 chars	63g 125x40x15mm
Sony CMD-Z5	GSM	WML HTML	96x72 pixels 4x17 chars	82g 88x49x21mm
Samsung SCH-3500	CDMA	HDML WML	96x32 pixels 4x12 chars	154g 112x52x25mm

Figure 1: Characteristics of some small and very small wireless devices.

The methods for displaying Web information on small displays fall into four categories: scaling; manual authoring; transducing; and transforming.

Scaling

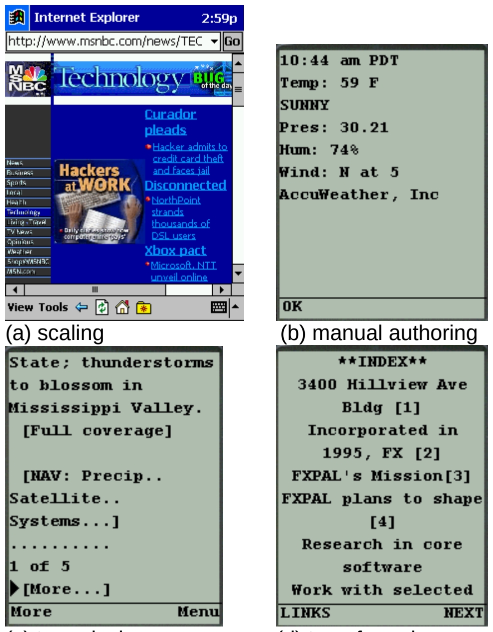
Web devices with high-resolution color displays, such as the Pocket PC with Pocket Explorer, support an experience that is closest to the desktop (figure 2a). The Pocket PC is capable of rendering many types of web content "full-size" and uses scrollbars to reposition the view within the web page. However, as most Web content is designed for 15" displays with at least 800x600 pixels, displaying full-sized pages on small devices results in lots of scrolling by the user. Alternatively, Web content can be scaled-down in the viewer, using a "fit to screen" feature, giving the user an interactive overview of the page. While scaling can reduce scrolling, it also reduces readability and ease of interaction. As the device's screen size and resolution decreases towards those of very small devices, such as Web phones, these problems become increasingly significant.

Manual authoring

For devices with lower resolution and smaller sized displays, the best user experience is achieved by using web content that has been manually authored (or re-authored) by professional web and graphic designers for the specific target device (figure 2b). Manual authoring allows the content to be laid out or summarized appropriately, and the interaction design can take into account known device limitations and idiosyncrasies. Unfortunately because of the labor required, only a small fraction of Web content has been manually authored for any particular device.

Transducing

Automated techniques for re-authoring web content have become popular because they are both cost effective and allow access to content maintained by third parties.


Internet Explorer
2:59p
http://www.msnbc.com/news/TEC
Go
MSNBC
Technology
BUG of the day
Curador pleads
Hacker admits to credit card theft and faces jail
Hackers at WORK
Disconnected
NorthPoint strands thousands of DSL users
Xbox pact
Microsoft, NTT unveil online
View Tools

```
10:44 am PDT
Temp: 59 F
SUNNY
Pres: 30.21
Hum: 74%
Wind: N at 5
AccuWeather, Inc

OK
```

```
State; thunderstorms
to blossom in
Mississippi Valley.
 [Full coverage]

 [NAV: Precip..
Satellite..
Systems...]
.........
1 of 5
▶[More...]
More              Menu
```

```
     **INDEX**
 3400 Hillview Ave
     Bldg [1]
  Incorporated in
   1995, FX [2]
 FXPAL's Mission[3]
FXPAL plans to shape
        [4]
 Research in core
     software
 Work with selected
LINKS           NEXT
```

(a) scaling (b) manual authoring

(c) transducing (d) transformeing

Figure 2: Techniques for displaying Web content on small devices content

Transducing is a basic automation technique that involves translating HTML and images into compatible formats (many small devices use a different markup language than HTML). So, for example, a client device can indirectly request the content through a proxy system, such as Mobile Google [1] or Wingman [2]. Such proxies retrieve the required content, transduce it into native formats, and compress and convert images to match device characteristics (figure 2c). Once transduced, the user sees a web page as a set of screens because most pages are too large to be transferred or rendered on the device as a single screen due to display size and network protocol limitations. Transducing systems, such as AvantGo[3], may also work off-line, giving users access to information when they are not connected to the network.

Transforming

In addition to making Web content compatible with client device mark-up language, transducing systems can also *modify* content to transform the structure, or experience, of interacting with the content (figure 2d). For example, the Digestor system [4] attempted to mimic the expert Web designer if they were faced with the task of re-authoring web pages for PDAs by performing semantic compression and layout modification of Web pages. A web page was split into multiple sub-pages (each better suited for the smaller display) and new navigation links were added to navigate around the sub-pages.

MODAL WEB BROWSING FOR VERY SMALL DEVICES

It was our experience with Digestor that motivated this research and inspired us to seek new ways to improve the Web experience on very small devices. Digestor credibly transduced content for a range of small device types but broke down on very small devices. The problem is that transducing a desktop-sized UI into many pieces for display on a mobile phone-sized UI inevitably results in a complicated structure that is difficult for users to understand and navigate.

Trying to use Digestor on mobile phones led us to re-examine the desktop user's Web experience and the desire to support that experience. We realized that much "browsing" involves following links and reading, or more generally, navigating to information and then using it. Since small devices have such limited user interfaces, it is very difficult for users to perform both of these activities at the same time. Thus, we realized it would be more appropriate to take the integrated activity known as browsing, and divide it into two separate modes: navigation and reading. This led us to separate the underlying link structure from the content to make the dual tasks of locating and reading content more manageable on very small devices. Although modal interfaces on large computers are often unnecessary and sometimes inappropriate, for small devices they can provide a significant advantage because they reduce the number of interactors and information a user needs to see at any one time. Power Browser [5] adopts a related approach, but for larger PDA-sized devices.

While separating navigation from reading is useful, observations of our colleagues' desktop browsing habits showed that there is clearly more to Web browsing than navigation and reading. A common sequence kept recurring: users would navigate through sequences of pages following links until they found something of interest. They would then perform any number of actions including reading, but also mailing, printing, saving, and even translating. We realized that an appropriate interface for very small devices not only separates navigation from reading, but also generalizes from *reading* to the ability to perform *any number of different actions* on link content.

The separation of interface concerns into two modes enables a much simpler user experience and increases the ways in which users can act on content. However, modifying the interaction to support distinct modes creates new problems:

Link naming: separating links from page content means removing contextual cues that can help users understand where links will lead. This is especially obvious when links have uninformative labels such as "click here". Separating the text surrounding such links from the links themselves limits the user's ability to anticipate where the link will lead them. Thus, we developed link-naming algorithms to improve link label quality.

Non-linked data: much content of interest to mobile users, such as phone numbers and addresses, are often not linked using HTML tags, and thus will not appear in a navigation view that only shows links. Thus we developed server-side data detectors to convert such useful bits of information into explicit links that users can manipulate much like other links, for instance to place a call (in the case of a phone number) or to get directions (in the case of an address).

Link overload: there are often more links on Web pages than lines available for displaying them on very small devices. To alleviate this problem, we developed link categorization algorithms to separate links into categories to aid in navigation and reduce clutter.

Unlimited content types and actions: the set of actions a user may want to apply to a given link depends on the link's type, and there are virtually unlimited content types and possible actions. Due to limitations inherent in very small devices (limited user interfaces, processing, bandwidth, and plug-in applications), we designed a mechanism to allow users to exploit network-based "plug-in" services to perform various actions. We made these services openly extensible to allow third party developers to associate new services with links based on link attributes, such as MIME type.

The remainder of this paper discusses, in more detail, how we solved these problems in our implementation of a modal web-browsing interface, allowing users to perform a wider variety of actions on a wider variety of content types than previously possible on very small devices.

THE M-LINKS MODAL INTERFACE

Before presenting the interfaces, it is worth considering the features of a representative target device. Figure 3 shows the input features of a Neopoint 1000 phone [6]. Two "soft buttons" can be programmed to execute HTTP requests when pressed and are labeled on the screen immediately above them (e.g. OPEN and TOOLS). Users can input text through the phone's keypad and use the thumb pad to select an entry in a list or move the input cursor on the display. Finally the "B" button goes back one page and "M" brings up the micro-web browser's menu.

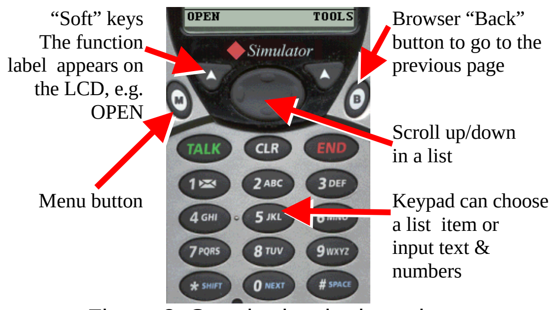
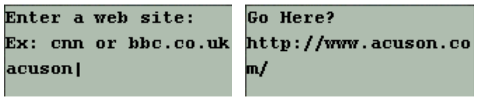

"Soft" keys
The function label appears on the LCD, e.g. OPEN
Menu button
OPEN
TOOLS
Simulator
Browser "Back" button to go to the previous page
Scroll up/down in a list
Keypad can choose a list item or input text & numbers

Figure 3: Standard web phone input

Figures 4 and 5 show a sequence of interfaces, or pages, produced by the M-Links system for the Neopoint. Again this is typical of the displays provided by many web phones, with capabilities to display 10 lines of text and a small set of icons.

The home page for M-Links (figure 4 left) prompts the user to enter the name of a web site they want to visit. Because entering a URL using a keypad is slow and difficult, the user only needs to provide the core domain element of the URL, such as the company name. M-Links expands the domain by prefixing it with "www" and appending various top-level domain extensions ("com", "co.uk" and so on) until a valid site is determined. The expanded URL is returned to the user (figure 4 right) who can confirm that is the correct site, or press the "back" button and specify the site or page more precisely.



```
Enter a web site:
Ex: cnn or bbc.co.uk
acuson|
```

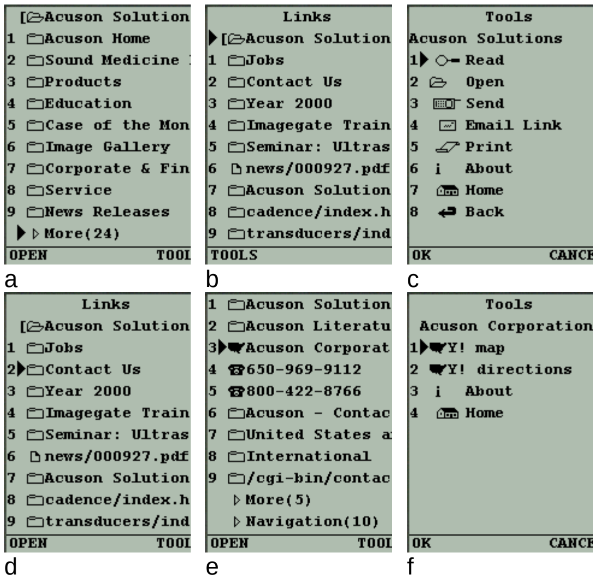
```
Go Here?
http://www.acuson.co
m/
```

Figure 4: Getting started with the M-Links interface

Figure 5a shows the navigation interface for the Acuson Solutions home page. The interface is streamlined to only support the interaction of navigating through the web site for some content. Therefore the content (or text) of the page is removed and the system supplies only the parts of the page necessary to support navigation - the links. These are represented with closed folder icons in a "list" like interface that is simple to comprehend and quick to navigate. Selecting a link from the list using the thumb pad or pressing the corresponding keypad number traverses to that linked page. In this example there are more links in the web page than lines on the phone so the system splits the list into separate pages. A 'More' entry is made available in the 10th list slot and shows how many more links are available on the current page. Selecting this entry, or pressing '0', causes more links to be displayed (figure 5b).

A "document" icon indicates links to non-HTML web content, such as PDF or multimedia. The title of the page whose links are being displayed is shown top-most with an open folder icon. Selecting "tools" when this item is selected causes the interface to switch modes to support the users selection of an action to execute on that web content. The M-Links interface changes to display possible actions that the user can perform on this link (figure 5c). Pressing the "right" soft key returns the user to the navigation mode (figure 5d). The user continues to dig down through the web hierarchy by pressing '2' and gets to Acuson's contact information page.

M-Links generates categories for the links on a page to keep the list to a manageable length, and to keep the interface simple. One of these, "navigation" (figure 5e bottom) collects links that are repeated across many pages on a web site. Web designers produce such links to help users navigate a site.



a

```
 [Acuson Solution
1 Acuson Home
2 Sound Medicine
3 Products
4 Education
5 Case of the Mon
6 Image Gallery
7 Corporate & Fin
8 Service
9 News Releases
  More(24)
OPEN          TOOL
```

b

```
     Links
[Acuson Solution
1 Jobs
2 Contact Us
3 Year 2000
4 Imagegate Train
5 Seminar: Ultras
6 news/000927.pdf
7 Acuson Solution
8 cadence/index.h
9 transducers/ind
TOOLS
```

c

```
       Tools
Acuson Solutions
1    Read
2    Open
3    Send
4    Email Link
5    Print
6 i  About
7    Home
8    Back
OK           CANCE
```

d

```
     Links
 [Acuson Solution
1 Jobs
2 Contact Us
3 Year 2000
4 Imagegate Train
5 Seminar: Ultras
6 news/000927.pdf
7 Acuson Solution
8 cadence/index.h
9 transducers/ind
OPEN          TOOL
```

e

```
1 Acuson Solution
2 Acuson Literatu
3 Acuson Corporat
4 650-969-9112
5 800-422-8766
6 Acuson - Contac
7 United States a
8 International
9 /cgi-bin/contac
  More(5)
  Navigation(10)
OPEN          TOOL
```

f

```
       Tools
 Acuson Corporation
1 Y! map
2 Y! directions
3 i  About
4    Home
OK           CANCE
```

Figure 6: The M-Links interface on a web-phone

M-Links uses the link as a basic unit of manipulation, but sometimes information the user wants to use may not be explicitly linked in the HTML. M-Links solves this by scanning the text, using server side data-detector to create new links, and then shows these links in the list of "actionable" items. Figure 5e shows three detected links, a physical street address and two telephone numbers. Choosing "Tools" for the street address provides a list of actions that can accept addresses, such as mapping tools (figure 5f).

The M-Links navigation interface exploits the user's familiarity with desktop file selection dialogs. The Satchel system [7] also utilizes this metaphor. However, M-Links does not operate on a well-structured and named file hierarchy but rather on the hypertext structure of the World Wide Web, and allows an unlimited number of actions to be associated with the content of that structure.

THE M-LINKS SYSTEM

Figure 6 shows the M-Links architecture. The system retrieves documents from the Web using the HTTP protocol, allows users to navigate and apply services to Web content, and delivers a suitable user interface to a variety of small wireless devices using HTTP. There are three main processing components: (1) the Link Engine, which creates the navigation interface; (2) the Service Manager, which creates the action interface; and (3) the Multi-Device User Interface Generator, which converts the interfaces into forms suitable for the requesting device and browser (e.g., HDML).

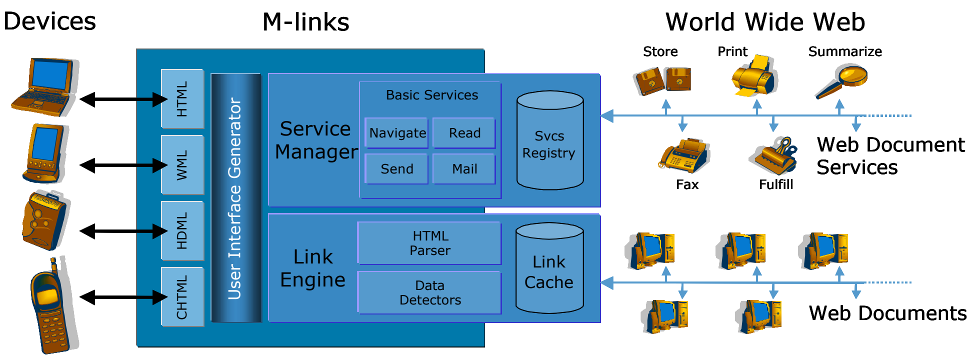


Figure 6 The M-Links architecture

GENERATING THE NAVIGATION INTERFACE

The Link Engine is responsible for processing web pages into a “link collection” data structure. The Link Engine works with a Link Cache where link collections for each processed page are stored. A request for the navigation interface for a web page involves these steps: (1) an HTML parser creates a parse tree from the page content; (2) the text elements in the parse tree are scanned by various data detectors for patterns (e.g., phone numbers and addresses) and new links are created; (3) the links are categorized; (4) each link on the page is extracted and added to the page’s link collection; and (6) the link collection data passed to the multi-device user interface generator which creates a page to be returned to the device.

Page Parsing

The M-Links system uses a variant of the fast HTML parser used by Digestor to construct a tree of HTML tags and text elements in an HTML page. All text in the page, between or around the elements, is also represented in the tree as special “text” nodes. Figure 7 llustrates how the parser would represent an HTML fragment.

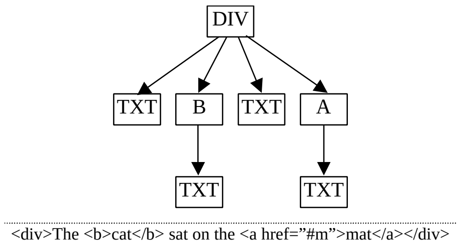


Figure 7: Parse tree representation of an HTML fragment

Many web pages make extensive use of HTML frames. Whenever the parser encounters a FRAMESET element it recursively calls itself to iterate through each FRAME element. As each frame is loaded and parsed, its representation is inserted into the tree, which in effect flattens the frame structure.

When designing M-Links we decided not to support HTML scripting elements such as JavaScript or VBScript. In part this is justified because scripts are often concerned with the rendering of the page, such as highlighting an image under the cursor. Since the M-Links interaction with a web page is very different than the desktop interaction, these script elements are not useful. However, other scripting elements perform structure or content changes to the page, such as generating interactive navigation menus. In practice, ignoring these elements is not a significant problem for two reasons. First, authors of many scripted pages also provide “hidden” or extra links for non-script supporting browsers. These are picked up by our HTML parser and fed directly into the link engine. Second, since most search engines also ignore scripting elements, most sites provide alternative pages for accessing the site.

Data Detection

Server-side data detectors provide an extensible mechanism for identifying elements of a web page on which users may want to perform actions, but which are not represented as an HTML anchors or tags. The current system has two such detectors, for phone numbers and addresses. M-Links data detectors are functionally similar to Data Detectors [8] and the selection recognition agent [9] but run on the M-Links server as opposed to on client devices.

Once the page has been parsed the hierarchy is passed to each data detector in turn. When a detector finds a match it modifies the hierarchy to insert a virtual “link” or data detection element that spans the text (and possibly other elements) containing the data in the hierarchy. Each data detection element is assigned an “HREF” property that contains target information about the virtual link. For an address this is the street address (as extracted). For a phone number this is a “WTAI:” URL identifier that would allow a web phone to dial the number when “followed”.

Link Extraction and Naming

Once the parse tree has been constructed and modified by data detectors, links are extracted and given appropriate display names for the interface. The Link Engine generates the link collection by traversing the tree looking for elements with an HREF property (such as <A> elements and data-detected links) or AREA elements.

The Link Engine then employs a link naming algorithm to determine concise but meaningful link labels to present to the user. The algorithm identifies a set of different possible labels for each link and assigns each a quality value representing how "good" or meaningful that label is. The lowest quality label is the link's URL itself. The highest quality label is assumed to be the title of the document at the link's destination (for HTML pages) as page authors generally make titles meaningful for bookmarking. Other label sources include: the anchor text of a link; the alt-text associated with an image link; and the link's relative URL path (excluding the host name and so forth).

When different links (to different documents) share the same name label the algorithm discards the label, moving to the next highest quality label in the links label set. The new labels are checked again to ensure the uniqueness of the new labels, guaranteeing a distinct label for each different link appearing in the final user interface.

Link Categorization

After links are labeled, the Link Engine categorizes them, which has two main benefits. First, it allows the system to present a much more streamlined interface. By hiding links in certain categories from the user the interface becomes less cluttered and more compact. Secondly, the user can quickly focus or direct their navigation actions by exploiting the presence of these categories. For example, if the user wants information about how to get to or contact a company it is probably accessible through a link on some navigation bar on the company web site. Currently two types of categories are determined: off-site and navigation.

Off-site

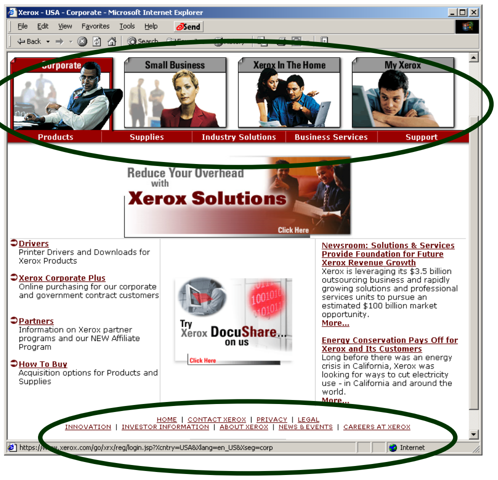
An "off-site" category is assigned for links that refer to documents at a different web site from the document being processed. For example, when processing (1) below, links to (2) are considered on-site while links to (3) are not.

1. http://abc.here.com/index.htm
2. http://def.here.com/docs.htm
3. http://abc.there.com/main.htm

The categorization algorithm begins by extracting the server's domain name from the URL and discarding the protocol, port, and other components of the URL. To avoid miscategorizing URLs that indicate multiple servers on the same site (e.g. 1 and 2 above), the algorithm constructs a "site identifier" for each URL by working backwards from the top-level domain (TLD) collecting up to two domain name components if it's a general, or gTLD (e.g., com, edu, gov, int, mil, net, org) and up to three components if it's a country code, or ccTLD (e.g., au, fr, uk, etc.). These site identifiers (e.g., "here.com" and "there.com") can then be compared against the site identifier for the page being processed ("here.com") to separate off-site from on-site links.

Navigation

A "navigation" category is used to classify links that are used throughout a site to navigate from anywhere to common index pages. Figure 8 shows links from Xerox's home page considered navigation links. Navigation links are identified using a number of page layout heuristics. The categorization algorithm examines adjacent links and attempts to verify: (1) that they reside on the same hierarchical level in the parse tree, (2) that any intervening text between them is identical and acceptable (e.g., "|", "-", or "]["), and finally, (3) that the transitions, or intervening "paths" in the parse tree, between them are identical and acceptable (e.g., they occur in adjacent table cells or in the case of links with image anchors, may be separated by line-breaks <BR> or paragraph tags <P>). The set of links at the bottom of figure 8 alls into the navigation category because of the recurring "|" character between anchors at the same depth in the page hierarchy.


Xerox - USA - Corporate - Microsoft Internet Explorer
File Edit View Favorites Tools Help Send
Back Search
Corporate
Small Business
Xerox In The Home
My Xerox
Products | Supplies | Industry Solutions | Business Services | Support
Reduce Your Overhead with Xerox Solutions
Click Here
Try Xerox DocuShare... on us
Click Here
Internet

Drivers
Printer Drivers and Downloads for Xerox Products

Xerox Corporate Plus
Online purchasing for our corporate and government contract customers

Partners
Information on Xerox partner programs and our NEW Affiliate Program

How To Buy
Acquisition options for Products and Supplies

Newsroom: Solutions & Services Provide Foundation for Future Xerox Revenue Growth
Xerox is leveraging its $3.5 billion outsourcing business and rapidly growing solutions and professional services units to pursue an estimated $100 billion market opportunity.
More...

Energy Conservation Pays Off for Xerox and Its Customers
Long before there was an energy crisis in California, Xerox was looking for ways to cut electricity use - in California and around the world.
More...

HOME | CONTACT XEROX | PRIVACY | LEGAL
INNOVATION | INVESTOR INFORMATION | ABOUT XEROX | NEWS & EVENTS | CAREERS AT XEROX

https://www.xerox.com/go/xrx/reg/login.jsp?Xcntry=USA&Xlang=en_US&Xseg=corp

Figure 8: Navigation links on www.xerox.com

Because the parse tree represents Web pages as containment hierarchies, the hierarchical level for each link can be determined by simply walking up the parse tree. If the hierarchy level is equal, then the intervening text and paths between navigation candidates is examined. Links with textual anchors (indicated by the <A> tag) or image anchors (indicated by the <IMG> tag) must occur in sequence with identical intervening text and paths between them. However, since the paths that are acceptable between <A> links are not the same as the paths that are acceptable between <IMG> links, there is a look-up table that lists acceptable paths per link type. Finally, links occurring in image maps (indicated by the <AREA> tag) are assumed to provide site navigation functionality and therefore are included regardless of inter-link paths. We have found that while these heuristics are not foolproof, they are acceptable for the majority of the web sites we've examined.

GENERATING THE ACTION INTERFACE

The Service Manager creates the action interface, which presents "tools" that users can use once they have located content using the navigation interface. These are not limited to reading or displaying the content on the device, and include desktop-like operations such as saving the content, emailing a link to a friend, or faxing and printing.

The action interface can support "compound" interactions implicitly exploited by desktop users. For instance, selecting a data-detected address link and retrieving directions in M-Links is analogous to copying an address from one Web page and pasting it into the Yahoo! Maps page to get directions on the desktop. In many respects the action interface works like "cut and paste" between applications on the desktop computer – taking a link from the browser's web page and pasting it into an application, whether that application is another browser (with an web-based service page), or some email the user is preparing, or the user's file system and so on.

When the user selects a link in the navigation interface and chooses "Tools," M-Links receives the request and passes the link to the Service Manager. The Service Manager constructs the action interface by adding actions appropriate to the selected link. There are a number of actions that are applicable to all links: "Read" allows the user to view the content of the page around the link (see below); and "About" returns a page showing the properties of the link's target, such as the URL, size and mime-type. The Service Manager evaluates a set of rules that each service specifies against the attributes of the link (e.g. the MIME-type of the link), the characteristics of the user's client device (e.g. the markup the device supports), and the user's identity (their email address). If all the service rules are satisfied then the service is added to the action interface. In this way a service developed to play audio files will appear only when an audio file is the selected link. When the list of actions becomes too large for one page, the last entry in the list is replaced with "More" which requests the next page of services.

There are two sources of services available to the M-Links architecture: general and content provider. General services are Web-based services hosted on particular sites that have previously been identified to the system. Users can specify which services they would like to see for which kinds of links. Content provider services allow web site owners to control the services available for links to their web site. These are specified in a "services.xml" file at the root directory relative to the link. This allows a content provider to include a set of customized services for their content. For example the web site www.patents.com might provide a service for overnight delivery of high quality patent documents.

Defining and Extending Services

The service specification document is the mechanism by which both general and provider specific services are introduced to the system and appear in the action interface. Figure 9 shows an abbreviated version of the XML-based specification for an email service. There are three main sections in a service specification:

1. The *rule* section describes when the service should be presented to the user as a possible service to be invoked. The section includes predicates that declare the required attributes of the link, the type of client device, and the identity of the user.
2. The *execution* section describes what URL should be executed when the service is selected. The service is activated using an HTTP request with a number of parameters, including the link to operate on, the user identity and so on.
3. The *presentation* section is separated into subsections for different languages. Each language element provides short and long labels to present to the user, as well as a longer description. A set of icon tags provides links to various graphical elements that can also be used when displaying the service to the user.

The XML format for service specification allows one or more files to be included, and allows other external specifications to be referenced using URLs which are resolved and substituted during the validation of the XML.

```
<service-group ID="email-group">
  <service ID="email">
    <rule>
      <accept match="any">
       <client-accepts>.*text/html.*
       </client-accepts>
       <client-accepts>.*text/x-hdml.*
       </client-accepts>
     </accept>
     <reject>
       <source-mimetype>URL/.*
       </source-mimetype>
      </reject>
    </rule>
    <execution>
        <execute>/mailto</execute>
    </execution >
    <presentation>
      <language ID="en">
       <service-name>email document
       </service-name>
       <short-name>Email Link</short-name>
       <description>
          Email a URL or the URL
          and its contents to yourself
          or a friend.
       </description>
     </language>
     <icon output-format="text/x-hdml">
          envelope1
      </icon>
    </presentation>
  …
  </service>
…
</service-group>
```

Figure 9: An email service specification

Adding new services to M-Links system is as simple as submitting a URL with the location of the service description file. The M-Links system then fetches and checks the description for validity, and adds it to the service registry. The next time a service menu is generated for a link the service rules will be checked and the service can appear in the menu.

We have experimented with a number of general services, which can be broadly categorized into: reading, for displaying content on the device; sending, for moving the link or content to a user via email, or WAP-alerts; printing for getting faxed or printed copies of content either at work or while mobile; and mapping, to get directions and print out maps of an address. While the various reading services enable link content to be displayed, such as PDF or HTML we remain convinced that reading is not the primary action people want to perform on content on such small devices. However it is often necessary to “check the contents” before proceeding with another action such as faxing and this is one of the most common uses of the reading services.

MULTI-DEVICE INTERFACE GENERATION

M-Links supports a variety of different compact user-interfaces to handle the variety of small devices that may access the system. For example, HDML and WML markup is used by web-phones, and HTML by most palm-sized devices. However, while the markup supported by the various client browsers on these devices differs, the actual underlying functional and interaction model of the interface is very similar. For example, all the different markup language interfaces provide a screen where the user can input a web site to be navigated. The UI Generator exploits this shared functionality using a combination of “template markup files” and program inheritance. Together these support a multi-view interface for the different device types.

The Multi-Device User Interface Generator outputs the final user interface by: (1) identifying the type of device making the request; (2) determining the appropriate type of response markup; (3) extracting various pieces of information from the request (such as the HTTP headers); (4) dispatching it to the relevant interface markup handler (for the identified markup type) to generate the interface; (5) returning the markup to the device. The same sequence is used for both the navigation and action interfaces.

Although the final markup may be different from device to device, the actual variables involved remain the same. Therefore M-Links employs a simple template substitution technique across most of the different interface screens in a similar manner to many web-based applications like BSCW [10]. The same base interface code is responsible for generating the variable values from request to request for that interface. These values are substituted into named fields that have been inserted into the template for a given screen. Figures 10 and 11 show two different templates used to generate the same interface in HDML and HTML.

Substitution on templates also enables the system to support many languages in addition to supporting many devices. The multi-device interface generator uses knowledge of the font encoding for the target web page to encode link labels in the font intended by the author (figure 12). Users can also specify in what language M-Links should be used to return the various menus, prompts, and messages. Selecting a different language causes the UI generator to simply choose a different set of markup templates (marked up using the users preferred language). When no equivalent language-specific template can be found, the system returns the English (default) version.

```
<HDML VERSION=3.0 MARKABLE=TRUE>
<ENTRY DEFAULT="%(defaulturl)" KEY=u FORMAT=*x>
<ACTION METHOD=GET TYPE=ACCEPT LABEL="GO"
       TASK=GOSUB FRIEND=TRUE DEST="%(next)?u=$u">
<LINE>Enter a web site:<LINE>Ex: cnn or bbc.co.uk
</ENTRY>
</HDML>
```

Figure 10: An HDML template markup file for the site address input screen.

```
<html>
<head>
<title>Input</title>
</head>
<body bgcolor="#FFFFEE">
<h2><img border="0" src="mlinks.gif"></h2>
<table border="0" width="530" height="100">
  <tr>
    <td width="523" height="51">
      <h2>Web Links</h2>
    </td>
  </tr>
  <tr>
    <td width="523" height="40"><b>
    Enter the name or URL of the web site where you
    would like to visit :</b></td>
  </tr>
</table>
<form method="get" action=%(next)>
<table border="0" width="530" height="50">
  <tr>
    <td width="465" height="11">
      <input name="u" size="65" value=%(defaulturl)></td>
    <td width="465" height="11">
      <input type="submit" value="Go">&nbsp;&nbsp;</td>
  </tr>
  <tr>
    <td width="465" height="27"><i>
    &nbsp;(E.g. cnn or bbc.co.uk)</i>
  </td>
  …
</body>
</html>
```

Figure 11: An HTML template markup file for the site address input screen

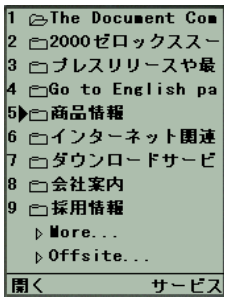

1 The Document Com
2 2000ゼロックススー
3 プレスリリースや最
4 Go to English pa
5 商品情報
6 インターネット関連
7 ダウンロードサービ
8 会社案内
9 採用情報
More...
Offsite...
開く サービス

Figure 12: Font encoding in the M-Links interface

Finally, in more complex interface situations, where the supported functionality or methods of interaction differ significantly between different devices and markup languages, the base interface generator is sub-classed by the markup handler to extend and tailor the functionality.

In generating both the navigation and action interfaces, we were forced to contend with strict restrictions on web phones on the byte size of pages that phones can fetch and display (part of the WAP standard). As a result, rather than embedding actual URLs in our interfaces, we use link offsets (in the navigation interface) and unique link identifiers (in the action interface) to reduce the size of the pages sent to client devices. Consequently, in the action interface, the Service Manager directs requests back to M-Links identifying the selected service and target URL using its unique identifier. The manager then redirects these HTTP requests to the appropriate web server as determined by the service description. This final redirected request identifies the URL target, the user, a return URL (back to the system once the operation completes), and a number of optional parameters. Identifying the link and service using small identifiers nearly halves the byte size of the action interface, allowing more actions to be displayed per page and a quicker transmission time.

IMPLEMENTATION, EXPERIENCE AND DISCUSSION

Our implementation of the M-Links system runs on a Java Servlet engine under Microsoft's IIS web server. We have been running the system continuously on a Pentium III processor with 256 MB of memory connected to a T1 network connection for over 6 months.

Performance

In this configuration the latency for generating the link interface for a cached page is approximately 2 seconds. The link cache stores the link structure derived by the Link Engine (which may be in memory or, in the worst case, has been serialized to disk) and therefore the only significant processing that occurs is the re-generation of the interface from the various multi-device interface templates in response to the request.

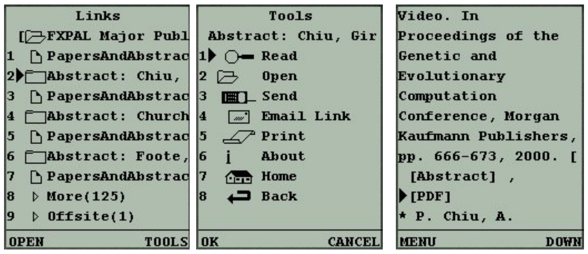
Where the page had not been pre-cached it needs to be fetched and processed. Consequently the time taken to service such requests is significantly longer. The processing itself, for an average page size, takes approximately 3 seconds, so when a web site returns the page quickly to M-Links (in 3 seconds for example) the total latency from receiving the request to returning the reply is around 6 seconds. Unfortunately we have found a number of sites perform very badly in returning pages - to the extent where the device making the original request can occasionally time out waiting for a response from M-Links (which is waiting for a response from the actual page's web server). Re-submitting the same request generally succeeds immediately as M-Links continues to attempt to fetch and process the page in the background. Fetching and processing is a "one time" time penalty that only occurs on the first request for an un-cached page (or a page that has changed since being caching). Subsequent requests for the same page perform with cached latency.

To reduce the initial latency penalty for many common web sites, we seeded the link cache with pages using a crawler and parsing pages off the Yahoo directory. A useful side-effect of this seeding is that we could provide a set of HTML pages (which could be parsed and presented by M-Links itself) which acted as a set of pre-built bookmarks for our users, removing the need to enter URLs in many cases.

It is worth pointing out that wireless devices (even on a wireless LAN), often experience significant delay in establishing a network connection for the request and receiving the data in response, and most web phones only support a connection speed of 14.4 KBps. Consequently we have found that for web sites that return pages even moderately quickly, the differences in perceived performance between cached and un-cached requests is negligible.

Additional Link Context

It became clear in early prototyping that the separation of the links from the content produced an interface that is clean, simple and vastly easier to use than any interface relying on transducing or transformation alone. However a consequence of this separation was a loss of context or information that normally surrounds a link on a page. This loss was especially pronounced on the data-detected phone numbers links - is the number a fax number or a phone number?



```
        Links
 [FXPAL Major Publ
1  PapersAndAbstrac
2 Abstract: Chiu,
3  PapersAndAbstrac
4  Abstract: Church
5  PapersAndAbstrac
6  Abstract: Foote,
7  PapersAndAbstrac
8  More(125)
9  Offsite(1)
OPEN          TOOLS
```

```
        Tools
 Abstract: Chiu, Gir
1      Read
2      Open
3      Send
4      Email Link
5      Print
6      About
7      Home
8      Back
OK           CANCEL
```

```
Video. In
Proceedings of the
Genetic and
Evolutionary
Computation
Conference, Morgan
Kaufmann Publishers,
pp. 666-673, 2000. [
 [Abstract] ,
 [PDF]
* P. Chiu, A.
MENU           DOWN
```

Figure 13: Tightly integrating a "reading" view with the link view

To address this problem the M-Links system more tightly integrated the HTML reading service to provide a "read around" feature (figure 13). This was provided in the interface through an option on the tools menu or via an additional programmable key on some web phones. Pressing the key or selecting the action toggled the view of the page to display the page text surrounding that link.

Portal Sites

Portal sites present a significant problem for our model of interaction where links are the primary component of the navigation interface. Sites like Yahoo or Excite often contain pages with over 200 links, which equates to at least 20 different M-Links pages. Finding an appropriate link in such situations can be time consuming. Transformation systems, like Digestor, perform better for these types of pages as they naturally separate areas of the page into distinct pages and provide an index into those areas – which mirrors the structure of how groups of links are laid out on portal pages. Web pages with large numbers of links remain an open problem with our solution.

Styles of Browsing

Finally, it is worth noting that our modal interface best supports *directed* browsing – where the user is following links in an attempt to find some particular content. More *casual* browsing, following links to see if there is anything useful on a site, does not work well in this model (and we would argue casual browsing cannot be well supported very small devices due to the inherent display restrictions). This is because the activities of navigating and acting are so closely intertwined a modal interface requires a user to keep switching modes to see if the page was of interest, and if not to select a new link to follow. In practice, the M-Links system supported this as well as other transducing or transformation solutions. The reading action transduced HTML pages and allows users to follow links in the content and continue the reading action on the new page.

FURTHER WORK

The M-Links architecture can easily be extended to support more types of link categorization than offsite and navigation. Categories could be based on the link destinations MIME type (e.g., PDF files, MPEG files, MP3 files) as well as based on layout characteristics (e.g., links separated into frames, table rows, columns and cells). By exploiting categories we may be able to further tighten the user's interaction and make finding useful content on large (through filtering), or poorly laid out web sites faster. Research is required to determine when such categories such be displayed or used, and we are concerned that the proliferation of categories may actually lead to more cumbersome interface.

From observations of use it is clear that for most people inputting text using a keypad, even in small amounts, is non-intuitive and a frustrating experience. We have started work on integrating the maturing technologies of speech recognition and VXML [11] to allow the system to seamlessly support mixed media input to dialogs and forms, as well as alternative forms of output.

CONCLUSION

Our experiences with the Digestor system and problems with other methods for putting web content onto small devices led us to consider how we could give users a *simple* interface for accessing and manipulating all types of web content, not just HTML. This paper presented an alternative approach that could work with existing methods, to make the user's browsing experience more *modal*.

We identified two different modes involved when browsing the web: navigation and action. The M-Links system supports these modes in separate interfaces. By focusing strictly on navigating the link structure, we were able to produce a simple list-based navigation interface that separates the content out. The content remains accessible through the action interface, allowing many different types of action other than reading to be executed on the content.

However adopting a more modal approach to presenting web content on very small devices creates new problems: link naming; non-linked data; link overload; supporting unlimited link types and actions. M-Links addressed these problems using link-naming algorithms to improve link label quality; server-side data detectors to convert such useful bits of information into explicit links; algorithms to separate links into categories; and a flexible XML-based mechanism for associating multiple (third-party) web-based services with links based on link attributes.

REFERENCES

1. Mobile Google, search engine for WAP and Palm devices, http://mobile.google.com/.
2. Armando Fox, Ian Goldberg, Steven D. Gribble, David C. Lee, Anthony Polito, Eric A. Brewer. Experience With Top Gun Wingman, A Proxy-Based Graphical Web Browser for the USR PalmPilot. Proc. IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98), Lake District, UK, Sept. 1998
3. AvantGo, web browsing and caching for wireless devices, http://www.avantgo.com/.
4. Timothy Bickmore and Bill N. Schilit. Digestor: Device-Independent Access to the World Wide Web. Proceedings from the Sixth International World Wide Web Conference (Santa Clara, 1997).
5. Orkut Buyukkokten, Hector Garcia-Molina, Andreas Paepcke, and Terry Winograd. Power Browser: Efficient Web Browsing for PDAs. In Proceedings of CHI 2000.
6. Neopoint1000 phone details http://www.neopoint.com/
7. Mik Lamming, Marge Eldridge, Mike Flynn, Chris Jones and David Pendlebury, Satchel: providing access to any document, any time, anywhere. ACM Transactions on Computer-Human Interaction, Vol. 7, No. 3, 2000.
8. Bonnie A. Nardi, James R. Miller, David J. Wright: Collaborative, Programmable Intelligent Agents. Communications of the ACM, Vol. 41 No. 3, March 1998.
9. Milind S. Pandit, Sameer Kalbag. The selection recognition agent: Instant access to relevant information and operations. Proceedings Intelligent User Interfaces '97. New York: ACM Press.
10. Bentley, R., Appelt, W., Busbach. U., Hinrichs, E., Kerr, D., Sikkel, S., Trevor, J. and Woetzel, G. Basic Support for Cooperative Work on the World Wide Web, International Journal of Human-Computer Studies: Special issue on Innovative Applications of the World Wide Web, Spring 1997
11. The Voice eXtensible Markup Language (VXML) Forum, http://www.vxml.org/