SmartView: Enhanced Document Viewer for Mobile Devices

Natasa Milic-Frayling
Ralph Sommerer

November 15, 2002

Technical Report
MSR-TR-2002-114

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

SmartView: Enhanced Document Viewer for Mobile Devices

Natasa Milic-Frayling & Ralph Sommerer
Microsoft Research Ltd, Cambridge, CB3 0FB, United Kingdom
{natasamf,som}@microsoft.com

Abstract

Web pages with complex layout do not display well on small screens. They are difficult to read on mobile devices such as PDAs or Web enabled phones because they require extensive horizontal scrolling.

SmartView is a browser feature that performs partitioning of an HTML document content into logical sections. Individual sections can be selected by the user for viewing independently from the rest of the document. The selected portion of the document is presented in a detailed view with modified layout for optimal reading or meeting other user's or device requirements.

The SmartView interface enforces the concept of a document by allowing the user to view both the document overview (e.g., a zoomed-out version of the document, a document thumbnail, etc.), indicating the logical decomposition of the document, and a detailed view of the selected section of the document.

SmartView further enhances the user's browsing experience by providing additional information on the document content. This information annotates logical sections and is derived from the document itself or supplied by external services. For example, SmartView assists the user in searching and browsing by providing feedback on the relevance of the page and its constituent parts with respect to a previously issued query to a search engine. The feedback is provided in two forms: as indicators of the number of hits on the overview presentation of the page and as term highlights within detailed view of a selected portion of the page.

SmartView can be implemented on the device as an integral feature of the browser or as a (local or remote) service that provides the decomposition of a page and query processing on the client's behalf.

1. Introduction

The ability to display digital documents on a variety of devices has become an issue of high importance in on-line document publishing. In particular with the proliferation of Web enabled mobile devices such as PDAs and smart phones there is a great demand for efficient dynamic modification of the document layout in order to accommodate the user's viewing preferences or device capabilities. Web pages are typically designed to be viewed on desktop screens and therefore require a certain minimal screen space which mobile devices cannot provide.

Since there does not yet exist a fully generic document description format that allows flexible and adaptive layout of document content on various devices, Web site authors are faced with the following choices: either they restrict their design ideas to the few options that render equally well on all or most of the potential target devices, or they create different content pages specifically for the use with particular target devices. Currently, far too little published material on the Web is suitable for mobile devices, despite the fact that most of today's mobile phones have Internet capabilities. In fact, it is worth noting that mobile Internet devices already vastly outnumber stationary devices like desktop computers.

In order to cope with existing online material, the browsers on PDAs are applying a number of scaling heuristics to provide a reasonable viewing of Web pages. Unfortunately, the right trade-off between readability of the content and the amount of horizontal and vertical scrolling required to view a page is very difficult to achieve.

On the other hand, restricted view and need for extensive scrolling has been linked to impoverished performance in information seeking tasks on small devices in comparison to the performance using devices with standard screen size [1].

We should mention that there has been a host of other approaches for dealing with this problem by providing summaries of document contents or extracts of information delivered by the services [2], [3]. While the reduction of delivered content has its benefits in many situations we are looking at the problem of preserving the content and the original characteristics of the document as much as possible.

SmartView is a prototype application that addresses the issue of displaying HTML documents on small devices in a novel way. It analyzes the layout of an HTML document and partitions it into logical sections that can further be selected by the user and viewed independently from the rest of the document.

In the following sections we describe in detail the design and implementation of the SmartView feature for PDAs and show how it could be used to enhance user's information access capabilities. We also give an overview of related research and conclude with the plans for future work.

2. SmartView Design and Implementation

HTML lacks the generic means of expressing common layout features such as multiple columns, sidebars, etc., that have been commonly used in designs of today's Web sites. Therefore, in order to implement design ideas and create two dimensional page layouts with several text flows and appropriate spacing between them, Web site authors usually resort to using HTML tables with fixed column widths and small blank images to obtain the proper spacing between various elements of the page (see [4]).

All of this results in rigid, inflexible, fixed-size Web page layouts that require certain minimal screen space and cannot be re-flowed to accommodate smaller screens, such as those of mobile devices. Indeed, Figure 1 depicts the front page of a news site (background) and, on the top, the portion of the page that can be seen using the integrated web browser on a Pocket PC([5]).

Note that the link bar on the left occupies more than half of the screen width of the Pocket PC, and that the scroll bars indicate the requirement of both vertical *and* extensive horizontal scrolling to see other parts of the page. It can also be seen that the main text body of the page (central column) is too wide to fit on the screen, thus requiring horizontal scrolling to read the text.

Note that simple pages without complex layout (e.g., pages having a simple flow of text) do not usually pose serious display problems on small devices because in such cases the browsers (including those on mobile devices) typically format the text to fit the width of the browser window.

SmartView approach recognizes the importance of the intended layout of the content, as specified by the author, and the fact that the Web pages typically involve a number of coherent logical units of the content.

Figure 1: Web site with complex design as seen on a Pocket PC

While in the HTML implementation these units are not explicitly marked we discover them by analyzing the structure of the page layout and allow the user to select each unit for viewing independently from the rest of the document. Because these portions are usually simple non-structured HTML fragments, they can be re-flowed easily to accommodate the narrower screen of mobile devices.

By selecting the browser's SmartView option, the page currently displayed in its window is analyzed and decomposed into logical segments based on geometric features of the layout, e.g., the table structure that defines the position of the page elements. Using this analysis, a thumbnail image is displayed with superimposed regions indicating the segments of the page discovered during analysis (Figure 2, image on the left).

The thumbnail image provides an overview of the Web page and serves as a user interface control to access its logical segments. If a region on the thumbnail is tapped with a stylus, the corresponding segment is extracted and displayed in the browser window appropriately re-flowed to is interested in (and leave out areas which appear

Figure 2: Web page thumbnail indicating the logical segments (left). Detailed view of a selected segment (right), displayed for optimal viewing and reading.

segment view and read the sections that he or she fit on the screen without the need for horizontal scrolling (Figure 2, image on the right).

The user can quickly switch back and forth between the thumbnail overview and the detailed to be of minor interest such as, for example, the link bar on the left or the advertisements on the right).

While in SmartView mode, all links executed in the detail view are pulled through and processed by the SmartView facility, resulting in a thumbnail view of the linked page with indicated page decomposition.

2.1 SmartView Page Analysis and Decomposition

Page analysis and partitioning used in SmartView relies only on geometric properties of the HTML page and is, therefore, language independent. The geometric properties of page elements are established by completely downloading the page, including all images, and formatting it to a standard page width suitable for viewing on a desktop computer (e.g., 800 pixels wide). From this layout, a thumbnail image is created, sized to fit the target screen of the device.

The page structure is then analyzed by recursively traversing the document object model (HTML DOM [6]) of the page.

In the current prototype we consider the sizes and arrangements of tables, cells within tables, and forms but for a more detailed analysis additional elements can be used. Depending on the sizes and arrangements of these elements, applying few simple heuristics based on elements' widths and heights, the algorithm determines whether a table or cell is to be bookmarked as a "logical section" or whether processing is continued recursively.

The result of the analysis is a vector of nodes (tables, cells within tables, etc.) in the document model, each representing a logical section. If such a section is requested for viewing, an HTML document corresponding to the page fragment is created by extracting the HTML representation of the node and all its contents. The extracted segment is wrapped in the HTML code representing the path from the root of the document model down to the node. In this manner a minimal, yet structurally consistent HTML document is created and then displayed by the Web browser on the device simply as any other HTML document.

The resulting document fragment is usually simple enough that the browser can re-flow it in a way that no horizontal scrolling is required to read it.

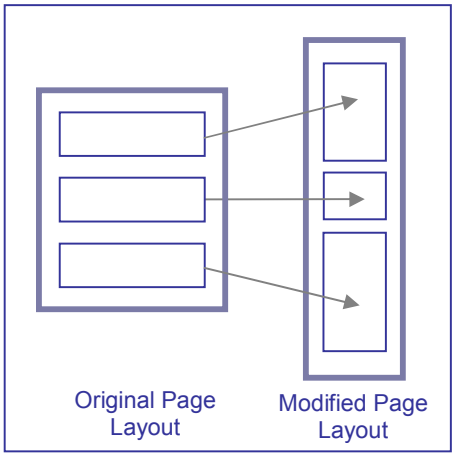

Figure 3. SmartView layout modification for two textural paragraphs and image in the middle.

Nevertheless, the width of the top-most element of the fragment document (for example the cell or table node that represents the logical section) is explicitly bounded to the width of the browser window.

The resulting document fragment is usually simple enough that the browser can re-flow it in a way that no horizontal scrolling is required to read it. Nevertheless, the width of the top-most element of the fragment document (for example the cell or table node that represents the logical section) is explicitly bounded to the width of the browser window.

Figure 3, for example, shows a logical segment containing three sub-elements, two paragraphs and an image, that is re-flowed in order to fit in a narrower window. Note that the paragraphs have become "higher" because their text lines are shorter to fit in the window. In contrast, the image in the middle is smaller because it may have been scaled down in order to keep its proportions intact.

In a few well defined cases, the SmartView prototype even adjusts the layout of a section by breaking it up completely and re-flowing it to fit in the window. Figure 4, for example, shows a logical section containing two sub-elements, for example two cells of a table, one containing a sidebar, and the other the main text. The layout is modified by changing the side-by-side arrangements into a top-down arrangement, because the latter is more effective on small screens.

2.2 Implementation Aspects

The current SmartView implementation relies on a service that performs the analysis of the page layout and page partitioning, thumbnail creation, and layout modification on behalf of a client.

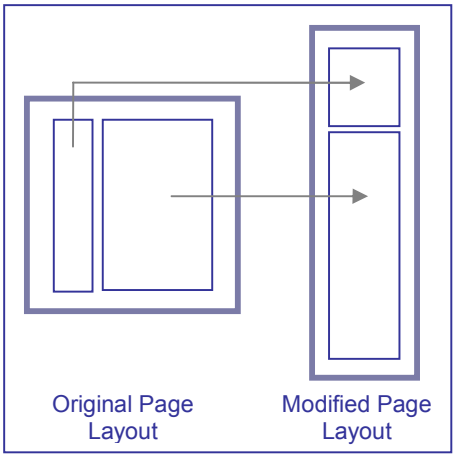

Figure 4. Layout modification for a side menu and the main text area

With new releases of the browser software for PDAs it will be possible to implement SmartView feature completely on the device. It is likely that even the thumbnail overview could be replace by a zoomed out version of the live page, displayed in the scaled down browser window.

Currently we have implemented two types of remote services that provide SmartView functionality: first one as a specialized *SmartView service* that performs full page analysis upon request and the second one tied to the Web server that hosts the original page: *extension of the server* that stores and delivers analysis of the page as requested by the client (analysis is automatically created during document authoring or publishing).

The user interface of the SmartView client on the mobile device consists of an HTML page and scripts which forward all corresponding requests to the SmartView server.

SmartView Service Implementation

As the user of the mobile device makes a request for a "smart view" of a Web page, the server downloads the page, creates a thumbnail image of the page, performs the analysis and partitioning of the page, and sends to the browser on the device a thumbnail image of the page with the partition details (see Figures 5 and 6). The partition is then annotated with context-specific information if available, such as the number of hits on a page with regards to the last issued search query.

When the user selects a particular section on the thumbnail, the server responds by extracting the HTML code of the desired section, creating a new HTML document that satisfies the new layout specifications, and delivering the new document to the device's browser for display.

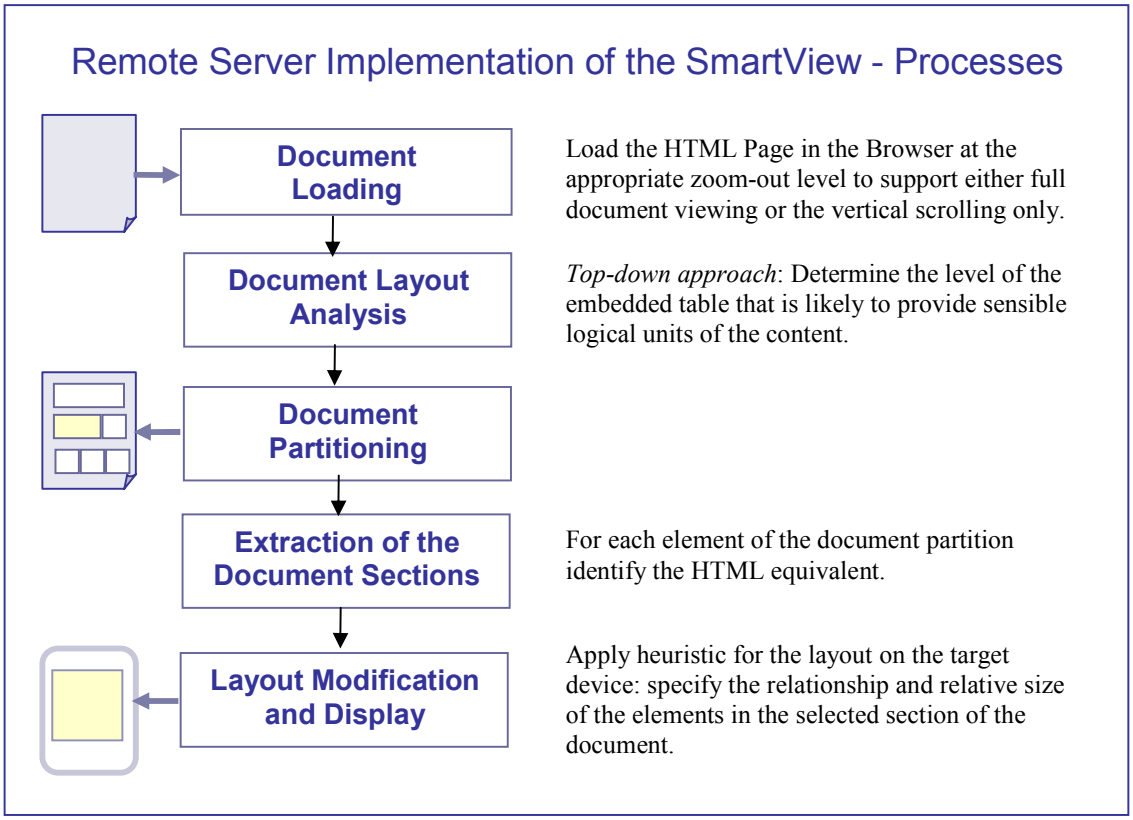

Figure 5. Steps involved in creating SmartView of a page

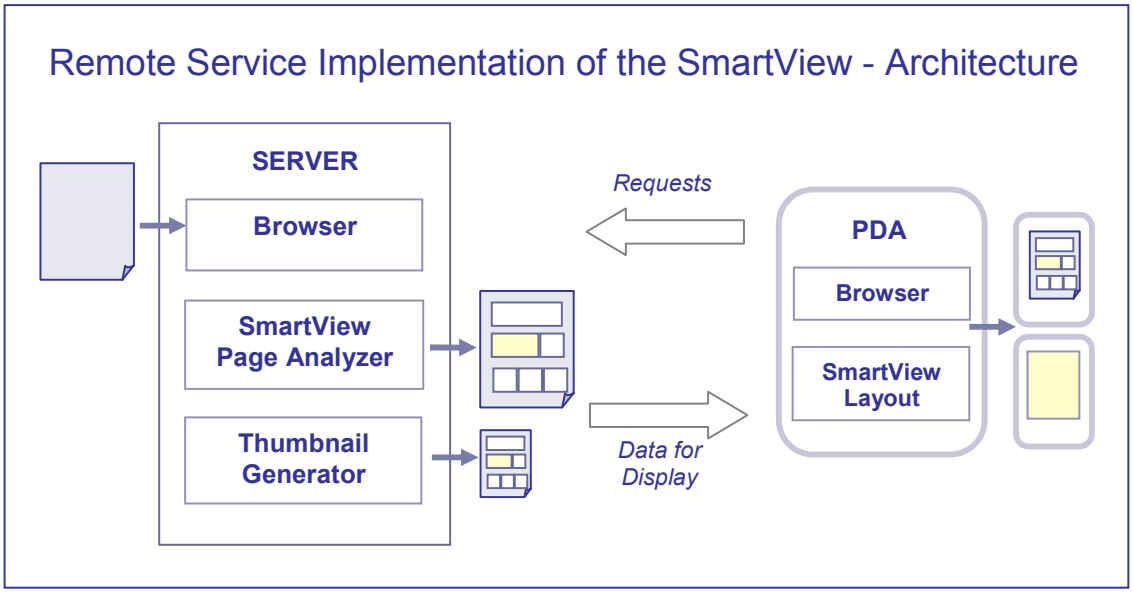

Figure 6. Architecture for the remote server implementation of SmartView

Note that the SmartView server only handles requests and performs operations related to the actual "smart viewing" of a page. In particular, it does not relay any requests for linked page elements to the original Web server (as a proxy server would) but instead modifies all URIs in HTML fragment documents to point to the original Web server. All linked elements are therefore pulled directly from their hosting server.

Document loading and layout is performed by an instance of Microsoft's Internet Explorer (IE) that is hosted in the SmartView server. When the document is complete, the server makes a copy of the document model and performs layout analysis and partition on the copy. Working on the copy prevents the SmartView server from interfering with IE's proper operation and frees the latter for subsequent requests.

The server maintains a cache of document analyses which enables it to serve requests for logical segments of pages for a certain period of time without having to redo the analysis.

SmartView Extension of the Web Server

Document analysis described above in relation to the designated SmartView server is in this implementation performed at the publishing time, on the page hosting Web server. Thus, the creation of the thumbnail image and analysis of the document need to be performed only once for a (static) page and stored on the Web site for consumption by client applications.

This approach provides savings in the processing time and band-width. Indeed, the thumbnail overview and document partitioning are ready for consumption by the client and at no time does the device need to upload the entire page. It is only the portions of the document, those explicitly requested by the user for viewing, that are provided to the client.

This approach is motivated by the related research work on automatically generated meta-data for Web publishing (see [7]).

Client and Local Service Implementation

The client application requests the "smart viewing" of a page and various logical segments of a page using specially formulated HTTP requests. Implementation by a (remote) service makes this type of facility available to a variety of devices which lack the appropriate computing capabilities to perform partitioning of complex Web pages, for example, smart phones.

However, a local implementation on a mobile device, either as a local service through which the browser connects to the Internet or as an integral part of the browser, can be more efficient.

Indeed, a local service can benefit from the same caching facility that serves the browser, thus avoiding the need to download separate SmartView fragments. Conversely, a browser-integrated SmartView facility can apply the necessary document partitioning and re-flowing in-place, without even having to load a fragment document into the browser.

3. Applications

SmartView provides a simple yet effective way of making Web pages with complex layout available to mobile devices by providing a page overview, partitioning the document into logical sections and giving users direct access to those. However, the partitioning of the document can also serve as a framework for providing additional information on different aspects or sections of a document. As such it is used in combination with various services and client applications to enhance information access experience of the users.

Additional information that can be provided on the logical sections (henceforth called *annotation*) is not restricted to document dependent or *static* annotations stored with or derived from the document. Instead, it can include context dependent or *dynamic* annotations that have no inherent association with the document.

Static annotations can include properties such as the role of various logical sections within the document. For example, a section in the SmartView of a page could be annotated to indicate whether it is a menu or whether it contains a user interface for a service, or similar. Such annotations can be generated at the authoring or publishing time, if the document format provides the facilities to express such generic structural properties, as suggested in [7].

Furthermore, note that such document dependent properties do not necessarily need to be stored *within* the document. Rather, they can be supplied by the server (or service) that hosts the document as a *separate* meta-document that a client requests from the server only if required (see [7]).

Dynamic annotations, on the other hand, are properties that can relate to the user's current

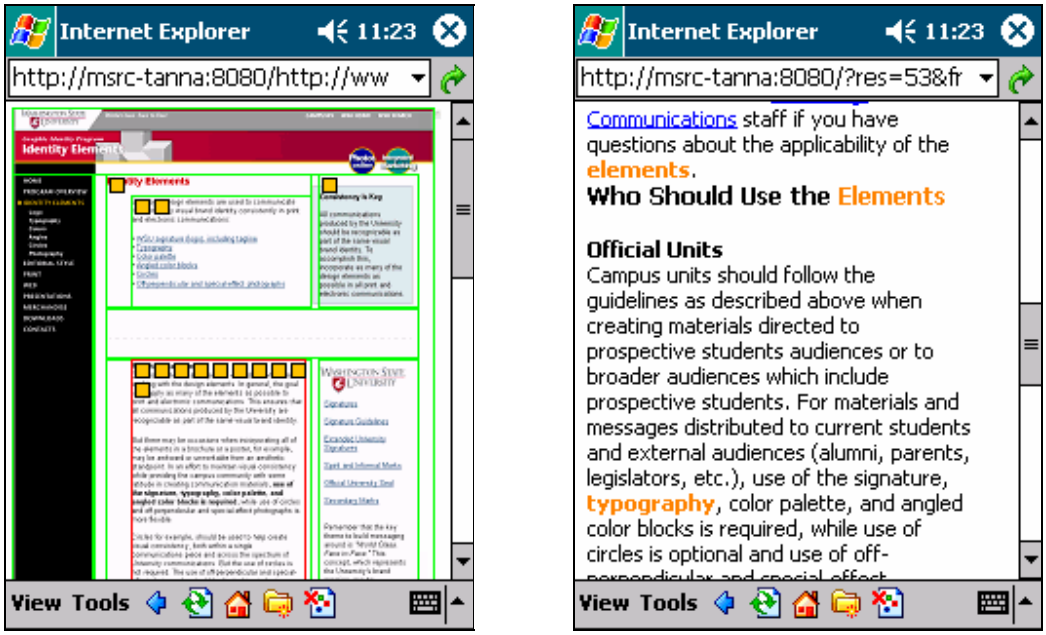

Figure 7. SmartView and Search (for "elements of typography"): The logical segments annotated with the number of hits (left). The selected segment with highlighted query terms (right).

browsing context rather than the document itself. Here we elaborate in more depth on such context dependent annotations, in particular, a coupling of SmartView with Internet search services.

SmartView and Search

This extension of SmartView functionality is motivated by the observation that search users are sometime concerned not only with the question of which documents in the result set are truly relevant to their query but also which *part* of a particular document contains relevant information. This is even more important on mobile devices due to limitations related to the screen size and need to have a quick and direct access to relevant parts of a document.

Similarly to topic modeling in ([8],[9]), if the user issues a query to a search engine the query terms are captured by the browser, i.e., extracted from the text box of a search engine's Web page, and stored. The query terms are then used in two ways to assist the user in assessing the relevance of documents found by the search engine.

First, logical sections in SmartView's thumbnail overview are marked with small squares to indicate the number of query term "hits" found in that particular section (see Figure 7, image on the left). Second, in the detail view, query terms are highlighted (see Figure 7, image on the right).

Query analysis and hit highlighting is done in a way similar to the prototype in ([8],[9]) but not only on the level of a full document but its logical sub-sections. By indicating the number of hits within logical sections user can quickly direct their attention to the most promising parts of a document found during search.

Section marking and highlighting remains active for all subsequent SmartView browsing activities, even if the user browses away from the result documents found in the query (i.e., until explicitly turned off). Thus, all further documents can still be examined and analysed with respect to the previous query. The reason for such a simplified context enforcement is to reduce the interface complexity. Extensions to more sophisticated query management and modeling of the user's topics of interest could be easily made.

4. Related Research

Most of the attempts to overcome the problem of document display on small devices are focused on presenting the user with information contained on the page in some other form rather than the page itself. For example, in [2] and [3]

the authors focused on devising methods to create suitable summaries of a single or multiple pages and present those to the user. While there are many benefits of this approach, we believe that there is a value in delivering the content of a page as originally designed by the author and, at the same time, allow the user to pick and chose what might be of interest for detailed viewing. Furthermore, we believe that in many situations discrepancy between the page views on different devices can hinder the optimal use of information.

We should point out that research on automatic extraction of the document architecture, including the logical and reference structure, has been conducted to some degree in the context of optimizing text authoring tools (see [10]). Future extensions of the SmartView to other document formats would certainly include some of these more sophisticated document analyses.

Furthermore, the idea of various partial views of Web documents has been explored in the context of collaborative Web Browsing in [11]. However, this work is restricted to Web pages created within specialized XML framework designed to enable multi-device and multi-user viewing of document content. Since it relies on specialized XML tags and XML splitting policy defined by the author, this approach is not easily extendible to the general HTML Web pages. On the other hand, in our approach based on the automatic analysis of HTML pages we can easily encode the results of the analysis into HTML to avoid repeated processing of the page as well as further enhance the document with the ability to receive the device specifications from the device placing a request.

Modification of a document layout to accommodate various devices has also been explored in the context of e-book research (see [12]). In the current prototype, we implemented rather simple layout adjustments, in comparison to the e-book prototype techniques, using heuristics for re-flowing content of HTML tables that are typically used to implement the layout of Web pages.

Bitstream's *ThunderHawk* facility [13] employs font technology to make Web pages accessible on small devices. It reduces the overall extent of a page, in particular its width, by using specially crafted extremely narrow screen fonts and by scaling down the size of images on the page. The resulting page is then displayed in landscape mode, to obtain a slightly wider screen (PDAs often have a portrait-shaped screen that is higher than wide). The ThunderHawk facility is an effective way of reducing the size of Web pages while retaining their complex layout, but it doesn't allow the user to 'zoom in' into regions of interest. Furthermore, users might find it difficult to sustain the required effort to read a text rendered using a very narrow font. SmartView, however, displays the selected section in full size for comfortable reading, re-flowed to fit the width of the window.

The browser on Palm [14] handheld computers employ a technology called *Web Clippings* to view pages on its small screen. The Web Clippings technology requires that web site authors develop special *Palm Query Applications* (PQAs). These contain small pieces of information, tailored specifically to fit in Palm's small screen. SmartView, in contrast, automatically and *on-the-fly* creates "clippings" of arbitrary Web pages, which users can view individually. Furthermore, SmartView clippings can be annotated with user and context specific information as we saw in the search scenario.

Future Work

Focus of our future work will be on exploring various mobile information access scenarios in which SmartView technology may increase effectiveness of the applications and services. This will be coupled with extensive user evaluation of the SmartView technology and prototype modifications based on the user feedback. Another objective is to explore the ways of extending the SmartView functionality to a variety of other document formats.

References

[1] Jones, M., Marsden, G., Mohd-Nasir, N., and Boone, K. Improving Web Interaction on Small Displays. In the Proceedings of the 8th World Wide Web Conference, 1999.

[2] Buyukkokten, O., Garcia-Molina, H., Paepcke, A. and T. Winograd. *Power Browser: Efficient Web Browsing for PDAs*. In Proceedings of the ACM Conference on Computers and Human Interaction 2000 (CHI'00), 2000.

[3] Buyukkokten, O., Garcia-Molina, H., Paepcke, A. *Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices*. In the Proceedings of

the Tenths International World Wide Web Conference (WWW 10), 2001.

[4] David Siegel, *The Web Is Ruined and I ruined it.* http://xml.coverpages.org/siegelRuined.html.

[5] Pocket PC http://www.microsoft.com/mobile/pocketpc/.

[6] HTML DOM http://www.w3.org/DOM/.

[7] DX Format: Towards Web Publishing with Rich Meta-data [submitted for publication]

[8] Milic-Frayling, N. and R. Sommerer. MS-Read: Context Sensitive Document Analysis in the WWW Environment. User Modeling in the Web Environment. *Microsoft Research Technical Report*: MSR-TR-2001-63, http://research.microsoft.com/scripts/pubs/trpub.asp, 2001.

[9] Milic-Frayling, N. and R. Sommerer. MS-Read: User Modeling in the Web Environment. *In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2001.

[10] Iwai, I., Doi, M., Yamaguchi, K., Fukui, M., and Y. Takebayashi. *A document Layout System Using Automatic Document Architecture Extraction.* In the Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'89), 1989.

[11] Han, R., Perret, V., and M. Haghshineh. WebSplitter: *A Unified XML Framework for Multi-Device Collaborative Web Browsing.* In the Proceedings of the ACM Conference on *Computer Supported Cooperative Work (CSCW'00),* 2000.

[12] Thacker, Ch. and R. Sommerer. A Prototype Electronic Book. (Unpublished, available on-line at http://research.microsoft.com/~som), 1999.

[13] Bitstream *ThunderHawk* http://www.bitstream.com/wireless/client_index.html.

[14] Palm http://www.palm.com.