PageTailor: Reusable End-User Customization for the Mobile Web

Nilton Bila, Troy Ronda, Iqbal Mohomed, Khai N. Truong and Eyal de Lara
Department of Computer Science, University of Toronto
Toronto, Canada
{nilton, ronda, iq, khai, delara}@cs.toronto.edu

ABSTRACT

Most pages on the Web are designed for the desktop environment and render poorly on the small screens available on handheld devices. We introduce Reusable End-User Customization (REUC), a technique that lets end users adapt the layout of Web pages by removing, resizing and moving page elements. REUC records the user's customizations and automatically reapplies them on subsequent visits to the same page or to other, similar pages, on the same Web site. We present PageTailor, a REUC prototype based on the Minimo Web browser that runs on Windows Mobile PDAs. We show that users can utilize PageTailor to adapt sophisticated Web sites, such as Amazon, BBC and MSN, for browsing on a PDA. Moreover, the customizations remain effective for up to a year, even as the content of pages is updated, and can be reused across similar pages, limiting the customization effort required to browse a site.

Categories and Subject Descriptors

C.5.3 [**Computer System Implementation**]: Microcomputers—*Portable devices*; H.4.3 [**Information Systems Applications**]: Communications Applications—*Information browsers*; H.5.4 [**Information Interfaces And Presentation**]: Hypertext/Hypermedia—*User issues*

General Terms

Algorithms, Human Factors, Performance, Experimentation

Keywords

Mobile Web, Customization, End-User, Small Screen

1. INTRODUCTION

Today, browsing the Web on a handheld device, such as a smart phone or a PDA, is an unpleasant experience [20, 21]. Most Web sites are designed for the desktop environment and render poorly on the small screens available on handheld devices. For example, figures 1(a) and 1(b) show the homepage of the BBC Web site as it renders on a desktop computer and a PDA, respectively. The limited screen size of the PDA causes significant frustration to users as they have to perform considerable scrolling to locate items of interest on the page or may find it difficult to see content that has been reduced in size.

Some content providers support handheld devices with technologies such as WML [34] and cHTML [36], where either device-specific or low quality versions of pages are handcrafted for display on small screens. Unfortunately, these approaches incur significant overhead as they require content providers to maintain multiple versions of their content to support a plethora of devices. As a result, adoption of these techniques has been limited to a small set of high-traffic Web sites that can afford the high cost of hand-tailoring content for mobile clients; and even then, support is limited to a few popular devices and content is updated at a lower frequency than the desktop version of the Web site.

This paper introduces Reusable End-User Customization (REUC [1]), a technique that lets end users adapt the layout of Web pages to the limited screen size of handheld devices. REUC lets users remove, resize or move Web page elements as part of their normal browsing activity. REUC records the user's customizations and automatically reapplies them on subsequent visits to the same page or to other, similar pages, on the same Web site. REUC supports even pages with content that is updated regularly, such as in Web portals and news sites, and requires users to customize only a small number of pages in order to browse an entire site.

REUC takes advantage of the observation that content in commercial Web sites tends to be generated from templates that do not change very often. Content providers have strong incentives for keeping the structure of their Web pages consistent because stable page layout provides for a familiar user experience and promotes Web site branding [18].

We describe PageTailor, a REUC prototype that runs on Windows Mobile PDAs. We implemented PageTailor as a plugin for Minimo, the mobile version of the popular Firefox Web browser. PageTailor is fully self-contained and executes stand alone on the PDA. It supports dynamic Web pages with rich JavaScript functionality, as well as pages with content that is updated regularly. PageTailor allows users to customize a page incrementally, over multiple sessions, and can reapply user customizations across pages of the same site. Figure 1(c) shows a version of the BBC homepage that

[1]Pronounced reuse.

MobiSys'07, June 11-14, 2007, San Juan, Puerto Rico, USA.

was customized with PageTailor by a user interested in news headlines.

Results from controlled experiments conducted with PageTailor show that users are able to successfully customize sophisticated Web pages from five real-world sites: Amazon, BBC, MSN, eBay, and Flickr. For a large majority of participants in our experiments, PageTailor successfully reapplies the users' customizations for at least a month, and in some cases for over a year. Moreover, PageTailor succeeds in reapplying the users' customizations to at least 75% of other pages which serve the same function in the site. On average, users take 10 minutes to customize a page; a time commitment that participants in our experiments deemed acceptable for customizations that can be reused.

This paper makes four contributions: (i) it describes REUC a technique that lets end users adapt the layout of Web pages to the limited screen size of handheld devices; (ii) it describes PageTailor, a PDA-based REUC prototype which demonstrates that end-user customization of Web pages is feasible for popular Internet sites using existing hardware and software; (iii) it shows that even for the case of pages with content that changes over time, page layout is stable enough to support the reapplication of end-user customizations for a period of a month or longer; and (iv) it shows that end-user customizations can be successfully reapplied across pages in the same Web site, thus limiting the number of pages that a user has to manually customize in order to browse the site.

The remainder of the paper is organized as follows. Section 2 introduces Reusable End-User Customization as a technique to adapt Web pages for mobile devices. Section 3 describes PageTailor a prototype REUC implementation that runs on a PDA. Section 4 describes the controlled experiments we conducted to evaluate PageTailor, and Section 5 presents the results of this evaluation. Section 6 discusses related work, and Section 7 concludes the paper.

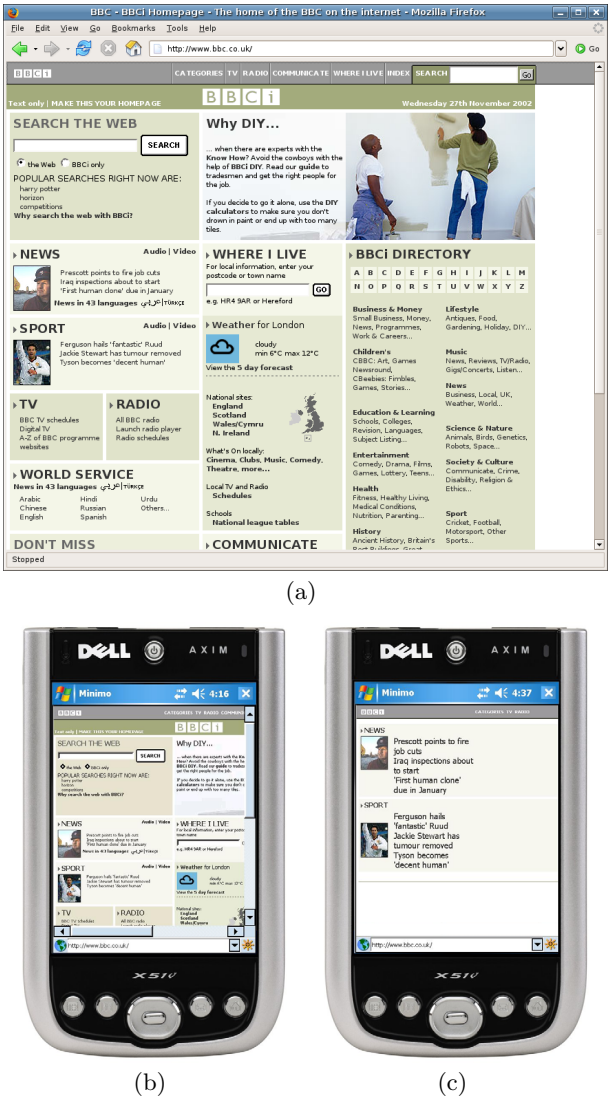


Figure 1: The BBC page as rendered on 1(a) a desktop and 1(b) a PDA. The same page is shown in 1(c) after it has been customized with PageTailor.

2. REUSABLE END-USER CUSTOMIZATION

We introduce Reusable End-User Customization (REUC), a technique that allows end users to adapt the layout of Web pages originally designed for the desktop, to the limited screen size of handheld devices, such as smart phones and PDAs. In our technique, users customize Web pages on the mobile device itself, using the same view of the content they use for regular browsing, by directly manipulating graphical objects on the screen to move, remove or resize elements on the page. Under the covers, REUC translates the user's customizations into operations that mutate the structure of the Document Object Model (DOM) [37] of the Web page being adapted. Thus, users do not require programming experience nor need to be familiar with the page's intimate structure in order to customize it.

The content and layout of modern Web pages is controlled by a combination of HTML, CSS [35], and client-side scripts mostly written in JavaScript. Typically, the browser first renders the static portion of the page based on the HTML and CSS descriptions, generating a DOM representation of the page, and then executes any associated JavaScript code which can dynamically add or reformat page elements. Customizations in REUC are done after the page has been fully rendered by the browser. This approach lets REUC support modern commercial Web sites that include sophisticated Web pages whose content is increasingly being controlled by JavaScript.

Because REUC requires user intervention, users are unlikely to benefit much while they are actively involved in customizing a Web page, as customization requires them to perform additional activities that are not directly related to their main purpose for visiting the page (e.g., restructuring a news article and then reading it is likely to require more effort than just reading the article). Instead, users benefit from their customizations when these customizations are automatically reapplied on subsequent visits to the same page or to other similar pages. We expect that REUC will be most beneficial in support of frequent tasks that involve repeated visits to the same Web sites or pages (e.g., reading news from our favorite site, going over the list of new Jazz CD releases, checking the weather forecast for city where we live). Conversely, REUC may be less applicable in the

case of arbitrary Web browsing involving unique visits to a random set of Web pages.

REUC succeeds in reapplying customizations across different pages in a Web site, and on different versions of a dynamically generated page (i.e., versions of Toronto's weather forecast page for different days), because the content in commercial Web sites tends to be generated from a small number of templates that serve as the basis for a much larger set of pages. Moreover, these templates do not change very often as content providers have strong incentives for keeping the structure of their Web pages consistent. Thus, whereas the content of a given page may change frequently, the structure of the page tends to remain stable. REUC allows users to customize a Web page incrementally over multiple sessions, and reapplies these customizations in sequence, on future visits to the page.

Notwithstanding, reapplying modifications to different versions of a page or across pages of a Web site is far from trivial as even small modifications can significantly affect a Web page's DOM representation, and content providers do make changes to their pages from time to time, e.g., adding a new banner image to advertise items on sale. The remainder of the section discusses the techniques that REUC uses to reliably reuse customizations over different versions of a page and across different pages. In the discussion, we use the following terms to describe REUC and evaluate the performance of the techniques presented. An *operation* is a single adaptation instruction performed by the user to an object (e.g. increasing the dimensions of an image once). A *customization* is a sequence of operations performed by the user to a single Web page in one session. *Success rate* is the fraction of operations from a customization which apply to a given page. Success rate indicates how well REUC is able to locate Web page objects and reapply the user's operations. For example, a 70% success rate for a Web page indicates that REUC is only able to locate objects required to perform 70% of the operations in a customization. A customization applies *successfully* to a page if applying the customization to the page generates a layout equivalent to that obtained when the user initially created the customization, and its application is beneficial to the user.

2.1 Reuse on the Same Page

REUC groups operations performed by the user on a given Web page into a customization, and associates the customization with the page's URL. For every operation, REUC records its type (e.g., move, remove, resize) and the page objects to which it is applied. On subsequent visits to a URL, REUC recovers all associated customizations from permanent storage and attempts to reapply them to the page.

There are several possible approaches for identifying the object to which an operation should be applied. First, we discuss simple approaches and explain why they are ineffective. Then, we detail two optimizations that REUC uses.

One option for locating an object that is conceptually straightforward involves keeping track of object identifiers (IDs) during the initial customization. To locate an object on later visits, the system can query objects by their IDs. Although all HTML tags have a predefined ID attribute, this technique does not work well in practice because Web designers rarely provide identifiers to objects. Moreover, if designers were to make use of IDs, then supporting reusable customizations would require that such IDs are unique and are employed consistently over time, such that they always refer to the same content or replacement thereof, as the content of the Web page changes.

Instead, REUC records for each operation the *path* from the root of the DOM tree to the object that is the target of the operation. This approach is robust to replacement of content, as the path will point to the correct location, where the new content now appears. Reuse of customizations on subsequent visits to a page involve traversing the DOM tree as specified by the instruction path of each operation, and applying the specified operations. Unfortunately, this *simple* approach does not work very well in practice as changes to the path to an object are common, for example as new content is added.

For example, Figure 2(a) shows success rates achieved in reapplying a customization to the Amazon homepage over time. The customization consists of 13 operations which were performed by one user on the homepage. The customization was obtained from the longevity user study, which we describe in Section 4. The results labeled *simple* report the fraction of operations retained with the simple DOM traversal approach we described. After 19 days from when the user made the adaptations, only nine (70%) of those adaptations are retained. The failures to accurately apply user operations are due to changes to object locations, however, those objects are still present. Specifically, the simple approach was unable to locate an account information table displayed at the bottom of the page, which contains information about passwords, shipping, tracking purchases and product search. On day 19, Amazon had added a disclaimer regarding their sale of personal and health products. This disclaimer was placed in a new table immediately before a form object surrounding the account information table. The addition impeded the simple approach's ability to locate that form object, and subsequently the target table. In the user's customization, she moved the account information table to the top of the page and so, subsequent operations that depend on the success of this operation, also resulted in failures. Interestingly, by day 27 the approach is able to apply all operations again as the disclaimer had been removed from the page.

2.1.1 Search

To address the false negatives observed with the simple DOM traversal algorithm, we augment the traversal with a depth first search. When traversing the DOM tree, as specified by an instruction path, REUC detects a path change when it is unable to proceed to the next object it has been instructed to traverse. In the Amazon example above, a change in path would be detected when the traversal leads to the new disclaimer table, instead of the form element expected. At this point, a search can be performed on neighbouring objects to the disclaimer table to find alternate paths. In our example, this search would enable REUC to find the expected form element immediately after the disclaimer table, and then proceed with the traversal.

In more complex cases, an expected object can be relocated because another object of similar type is added at its initial location. For example, instead of adding a disclaimer table, Amazon could have added another form. In this case REUC would not detect a failure until after it has traversed the new form and was unable to locate the target table. In such cases, when REUC is unable to locate an object at

a given tree level, it backtracks and searches for alternate paths one level above. The backtracking process can continue until the root of the document is reached, as long as a suitable alternate path is not found. This behaviour is desirable since the detection of path changes may be delayed for several levels. From experiments, we have encountered pages in which a change in paths occurs at tables near the document root, however they are not detected for a number of subsequent traversals.

The above algorithm can, for some objects no longer present in the page, lead to searches which explore the entire document tree. To improve performance, REUC limits the search space such that at any tree level only the nearest neighbours to the expected object are searched. Thus, although the search algorithm described has, in the worst case, a time complexity $O(n)$, where n represents the number of objects in the page, on the average case its performance is better. The benefits of the search over the simple traversal are illustrated in Figure 2(a), with the label *search*. Clearly, search improves success rates, as can be seen, for example, in day 19. Consequently, false negative rates for this day also drop considerably.

2.1.2 Context

One weakness of the algorithms introduced so far is that the only mechanisms used to verify that the object located is in fact one that should be modified are based solely on the path traversed and the type of the object. Thus, in our example, if the traversal lead to a table, there is no way of knowing that this is in fact the account information table. In fact, for day 19, one operation applied to the page with the traversal algorithm augmented with search was a false positive, as can be seen in Figure 2(c). A false positive is the application of an operation to an incorrect object. In her initial customization, the user removed a section of the page showing a featured product of the day. In the reapplication, REUC removed instead a different section which presents ongoing sales. We enhance the algorithm by ensuring that whenever a search is performed REUC uses context information about the object located to determine whether it is in fact the target of the operation. This context includes information about the number and types of child objects, as well as type information about the target's parents.

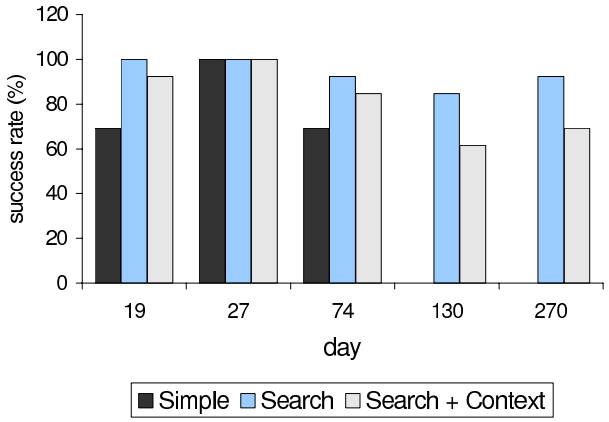
Figure 2 shows that augmenting the DOM traversal with both search and context achieves very low false negatives while maintaining little or no false positives. In Figure 2(c), for example no false positives are reported with *search + context* for day 19. In fact, very few false positives are reported over the course of the 270 days under this algorithm.

2.2 Reuse on Similar Pages

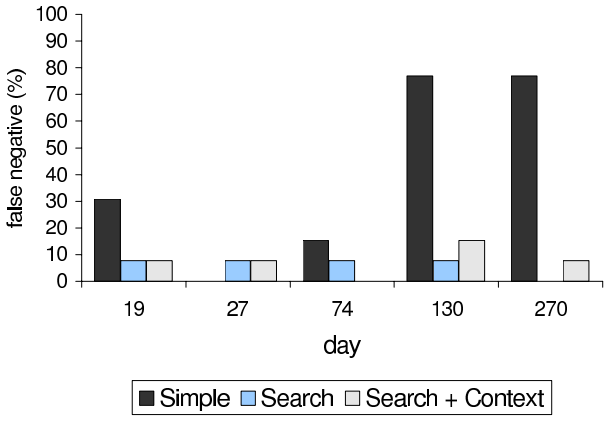
REUC can apply customizations to pages that are similar to those initially customized by the user. We next discuss how REUC identifies which customizations apply to a new page.

2.2.1 General Comparison of Document Structure

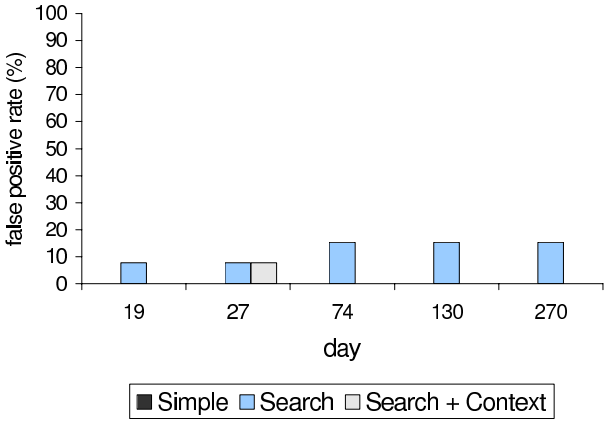
One possibility is to treat the problem of matching Web pages to customizations as that of clustering documents produced from the same template. Once we know that a new document originates from the same template as one the user customized, we can apply customizations made to the earlier document onto the new one. Clustering of structured docu-



(a) Success Rate



(b) False Negatives



(c) False Positives

Figure 2: Performance of object location algorithms. The figure shows that search and context improve success rates when reapplying customizations to a Web page over time.

ments is a problem addressed in the information science and information retrieval literature. By treating Web pages as trees, we can take advantage of well known algorithms that compute tree edit distances, the minimal cost to transform one tree into another [12, 13, 10, 6, 38, 33]. The fastest of these operate with time complexity $O(n_1 n_2)$, where n_1 and n_2 represent the number of objects in each document. We have experimented with one fast algorithm, described in [12], and found it to consistently require several minutes to compare two average sized pages of nearly 1200 objects on a reasonably powerful desktop computer. On a mobile device this translates to tens of minutes. Moreover, the space complexity of the algorithm is also high as it builds matrices of size $n_1 n_2$.

It is clear that such algorithms would not work well in this context, as users cannot be expected to wait for tens of minutes before they are told that no customizations apply to the page on screen. The challenge with these algorithms (in the context of our application) is their generality. They take into consideration an entire document, rather than the parts which are of interest for the customization, and this is time consuming.

2.2.2 *Customization-Based Structural Analysis*

Instead of comparing documents for the purpose of recognition, it is more efficient to compare a page against the instruction paths specified in the customization operations. In this approach, we determine the success rate of a customization on the new page. If the success rate exceeds a certain threshold, then the customization is considered successful for that page. The success rate cannot be determined statically, as customization operations may depend on the successful execution of other, non-idempotent operations. For example, one operation may depend on the relocation of an object by previous operations, and we cannot determine whether such operation applies to the page without executing all preceding operations on which it depends. Thus, when the user loads a new page, the system attempts to apply all available customizations using the algorithm described in Section 2.1, and then decide which customizations should be associated with the page. This speculative application of customizations is not performed on the actual page as it would result in a deformed page, so rather, we test each customization on a shadow copy of the page's DOM tree. This shadow is a much lighter version of the DOM tree as objects only contain the few attributes needed for customization operations. Only when we identify successful customizations we apply those directly to the document. At this point we also associate the URL of the new page with the customizations so a future visit to the page does not require an expensive structural analysis, as a simple comparison of URLs will suffice.

To improve responsiveness, we restrict the recognition process to include only customizations associated with the domain name for the page. This is a reasonable restriction given that customizations are likely to be relevant only across pages that serve similar functions. Moreover, in our experience the likelihood of finding pages with similar structure on different sites is minimal.

Employing the full algorithm described earlier when testing customizations onto pages that match poorly can lead to unnecessary searches which are detrimental to the responsiveness of the system. Thus, since REUC only needs an indication of whether it should apply a given customization, it can employ a restricted version of the search. This algorithm restricts the number of searches for a given path traversal such that at most one search is allowed. So, if a search is performed at a given level of the path, no search is allowed at subsequent levels. Intuitively, this restriction limits support to pages in which the overall structure is unchanged, except that at one level an object may have been moved either by addition of new objects or removal of old ones. It places a bound on the search at the expense of success rates: the algorithm may report higher false negatives (inability to locate existing elements) than the unrestricted search. As we discuss in Section 5.5, experimental results indicate that the restriction on search has only a limited impact on REUC's ability to match Web pages to customizations. However, the improvements in responsiveness are significant, as shown in Section 5.4.

We determined empirically that at 70% success rate, customizations are still beneficial to users, and thus we use this as a threshold to identify successful customizations. In other words, we apply a customization to a page if the paths specified in 70% or more of its operations are present in the page. Further discussion on how this threshold is determined is presented in Section 5.3.

2.3 Discussion

Allowing users to modify the layout of Web pages can have both negative and positive implications for content providers. On the one hand, if content providers have incentives to control the layout of content, they may not receive this approach well. As such, providers can change the layout of their pages frequently to circumvent reuse of customizations. This is costly, however, and reduces their ability to offer a consistent experience to their users. A possible compromise would be for content providers to tag documents with meta-data hints that flag portions of the content that should not be adapted by REUC. On the other hand, providers may benefit from REUC as this approach may increase traffic to their site from mobile users that may have otherwise stayed away, and reduces the cost of supporting multiple devices. Content providers could also improve REUC performance by tagging documents with meta-data that identifies the templates used to generate pages. This support would simplify the task of recognizing new Web pages, and would not require much effort from providers. For example, a provider may add unique IDs to each template they use, which could then be used by REUC.

3. PAGETAILOR

PageTailor is a prototype implementation of REUC for the Minimo Web browser, which is a Firefox-based browser that runs on Windows Mobile PDAs. PageTailor is implemented as a plugin based on the Mozilla API.

In this implementation, we target PDAs as they readily provide the processing power needed to accomplish reapplication tasks at reasonably fast speeds. Application of REUC to smart phones and other mobile platforms is left for future work.

Figure 3: The PageTailor toolbar on a PDA.

When browsing a Web page, users invoke the PageTailor customization interface through a button on their Web browser. The interface consists of a toolbar presented at the bottom of the browser's window (see Figure 3). The toolbar provides functionality that allows users to modify the layout of the Web page displayed. Specifically, it has six buttons: "Remove", "Increase", "Decrease", "Move, "Undo" and "Done". Users make modifications to the layout of the page by using the stylus to tap on elements, such as images or text fragments, and then select a button on the toolbar. The interface also allows for selection of multiple objects by dragging the stylus pen and tracing a line over the objects.

The PageTailor plugin executes within the environment of the Web browser and accesses the DOM interface exposed by the browser's layout engine. PageTailor also accesses persistent storage on the device in which it executes to store and retrieve user customizations.

To customize a Web page, PageTailor operates on the DOM model according to the user actions, and the layout engine updates the user's view of the page. When the user completes a customization session, her customization is saved to the device's storage. Each customization is stored as a file containing a sequence of operations and associated paths to target objects. PageTailor also maintains an index file which keeps track of associations between customizations and Web pages.

To support reuse of customizations when the user loads a page on the Web browser, the browser passes execution control to PageTailor after it has rendered the page (generating its DOM model) and has completed the execution of scripts. PageTailor then determines if there are customizations matching the page and applies them. When PageTailor matches a customization to a new Web page, it generates a regular expression which matches the new page and all other pages already associated with the customization and stores the association of this regular expression with the customization on the index file. This mechanism makes reuse of the customization on future visits to any of the associated pages fast, as PageTailor need not perform a structural analysis of the pages.

PageTailor avoids downloading images which have been removed by the user's customization by initially preventing the browser from loading all images, as images are not needed to generate the required DOM model. It is only after PageTailor has applied all customizations matching the page that it triggers the browser to load and display those images not removed by the user's customizations.

PageTailor also provides feedback to the user by showing on the browser's status bar the percentage of operations that were successfully applied to the page. It also gives the user the option of undoing the effects of customization operations, one at a time or all at once. In addition, PageTailor provides an interface that enables users to manually associate customizations to new pages for cases where the automatic process fails to do so, or in cases in which the user wishes to reuse existing customizations on domains that differ from those on which the customizations were initially made. With the same interface users can remove customizations, if they are no longer of benefit.

3.1 Discussion

The current PageTailor prototype is fully self-contained and executes stand-alone on the PDA. The main advantage of this approach is that it greatly simplifies the deployment of PageTailor. Once a user has installed Minimo on their PDA, by downloading a freely available executable from the Web, they only have to download and install the PageTailor plugin and they are ready to go. Because the plugin is written in JavaScript there is no need to compile any code.

The design of PageTailor, however, does not preclude the use of a server side proxy if one were to be available. The proxy could be used to offload the permanent storage of customizations, as well as the computation required to determine the suitability of customizations to Web pages. In addition, assuming the proper safeguards are put in place to protect the user's privacy, the proxy could operate as an aggregation point that lets users share customizations if so they choose.

A limitation of PageTailor is that allowing users to move objects arbitrarily on a page, without regard for the semantics of the underlying structure, makes it possible to break JavaScript code which addresses objects by path. This danger is limited, however, as well written JavaScript code address objects by IDs rather than paths, since paths can vary across browser implementations. A similar problem can occur when a user's customization moves a submission button outside the scope of its form. A possible solution to be explored could be maintaining a shadow copy of the page's DOM which maps original paths to new ones.

4. EXPERIMENTS

We conducted two controlled experiments to evaluate the suitability of PageTailor as a tool for adapting complex Web pages, as well as to determine PageTailor's performance in terms of its success rate for reapplying customizations to pages with content that changes over time, and across pages in the same Web site.

We conducted our user studies in a laboratory at the University of Toronto with participants from the general student body. Study participants were recruited through ads placed in high traffic areas of our building, emails sent to various departmental mailing lists and word of mouth. In both studies, participants were free to perform customizations which were of interest to them.

4.1 User Study 1: Longevity

The first goal of the study was to evaluate the feasibility of customizing Web pages using PageTailor. This is achieved by determining whether PageTailor provides the required functionality to accomplish customization objectives of users within a reasonable time. The second goal was to determine the *longevity* of user customizations, defined as the length of time (measured in days) over which a given user customization is accurately reapplied to a page. Longevity of customizations provides a measure of the length of time over which the effort a user invests in customizing a Web page can be amortized. Additional metrics we use to evaluate longevity are *desirability*, the length of time over which participants find it desirable to apply their initial customizations and *accuracy*, the participants' subjective view of how well PageTailor applies their customizations on a given day for which they deemed desirable. Desirability provides an upper bound on the longevity of customizations and the performance of PageTailor. Accuracy, on the other hand, is a measure of the closeness with which PageTailor approximates this upper bound.

4.1.1 Methodology

The study was organized in two phases, a customization and a validation phase. For the customization phase, the methodology was as follows. To evaluate PageTailor's ability to meet users' adaptation goals, before we introduced PageTailor to participants, we presented them with a printout of the page they were about to customize, and an outline illustrating the area of the page that could be viewed at once on the screen of a PDA. We then asked participants to sketch on paper their intended design. We told participants to take into consideration the dimensions of the outline, but we did not specify any other constrains on how the page could be transformed. Once they had completed their paper-based designs, participants were introduced to PageTailor and allowed time to familiarize themselves with its operation by customizing a sample Web page. Participants were then asked to attempt to replicate the customizations they had sketched earlier on paper, but this time using PageTailor. During this process, we collected traces of participants' customizations. Once they finished customizing, we asked for their feedback on whether they were able to accomplish their design goals.

In the validation phase, participants were asked to assess the desirability and accuracy of their customizations on future versions of the page they customized initially. For this purpose we asked participants to compare different versions of the page as they were being simultaneously presented on two desktop displays. The first display showed two views of the initial page the participant customized, one before and one after her customization was applied. The second display, showed two views of a future version of the same page, both before and after the participant's customization was applied.

For the future version of the page displayed, participants were asked first, to specify whether it was desirable to them to apply their initial customizations to the new page, and second, what percentage of their initial customizations was applied correctly by PageTailor. They were then asked to provide the same information for additional future versions.

We recruited sixteen participants with varying degrees of experience using PDAs. Four participants reported having no experience using a PDA, four others reported having low comfort with a PDA, and eight reported that they are comfortable using a PDA.

4.1.2 Platform

At the time we conducted our first user study, the Minimo Web browser was in its early stages of development, and did not provide support for extensions. Thus, we used a thin-client approach to enable participants to perform customizations on the PDA. We ran the Mozilla Firefox browser fitted with the PageTailor plugin on a dedicated desktop PC, and used VNC [28] to map the desktop's screen to the PDA. The PDA and desktop communicated over a dedicated low latency 802.11b link. In this configuration, to the participant, PageTailor appears to be executing directly on the PDA.

Our desktop had a 2.4GHz CPU, 512MB of RAM, two displays, and was running Fedora Core Linux 4. The desktop also ran the Mozilla Firefox 1.0.7 browser and a VNC server. Our PDA was a Dell Axim X51v equipped with a 634MHz XScale processor and 64MB of memory. The PDA ran Windows Mobile 5.0 and a VNC client. It's interface consisted of a 3.7-inch touch screen with support for VGA resolution at 480x640 pixels (portrait) with 16 bit color.

Web Page	Trace Length (Days)	DOM Objects
Amazon	270	2003
BBC	501	1359
MSN	431	1242
eBay	32	1231

Table 1: Pages used in Longevity Experiment

4.1.3 Pages Studied

The experiments were conducted on homepages of four popular Web sites: Amazon, BBC, MSN and eBay. We chose these four pages, first, due to their popularity and, second, because they are updated frequently making it important for customizations to persist. These pages include JavaScript code and are relatively sophisticated in terms of DOM objects, each having more than 1000 objects. For comparison, as of November 27, 2006, the Google homepage had only 110 objects.

For the eBay homepage, we cached copies of it every hour for 32 days. For the other pages, we obtained copies spanning longer periods from the Internet Archive's Wayback Machine [32]. The Internet Archive data gives us a larger range of dates (from 270 to 501 days), but the dataset is sparse, in some cases missing daily pages for periods of over one month. For Amazon, BBC and MSN, we chose the initial date to correspond to a day in which a new layout design is observed. We then included future versions of the page. In choosing future versions our goal was to provide both fine granularity and a long range of dates. To accomplish this, we approximate the sequence of dates to a power sequence of base 2, with days 0, 1, 2, 4, 8 and so on, as long as a version for the date is available. When a version of a page for a given day was not available in the archive, we chose the version for the next closest date. Table 1 gives a list of pages, the range of days studied and the number of DOM objects present in each page. Each Web page was customized and evaluated independently by four participants.

4.2 User Study 2: Coverage

The goals of this study were to determine the extent to which PageTailor can reapply customizations across pages of the same Web site and to quantify PageTailor's performance on a PDA. We define *coverage* as the proportion of pages in a site for which a given set of customizations can be applied successfully.

4.2.1 Methodology

We observe that while a Web site may have a large number of pages, these pages can be grouped into a much smaller number of page types that support specific functions within the site. For example, Figure 4 shows a somewhat simplified schematics of BBC News, a section the larger BBC Web site, which includes three types of pages: the section homepage, subsection homepages such as Europe and Entertainment, and article pages. Clearly, the number of page types varies between sites, and three is simply the example we use here.

The study was divided into a customization and a validation phase. In the customization phase, we asked participants to choose, out of a set of well known sites, one which they felt was most familiar to them. We then asked

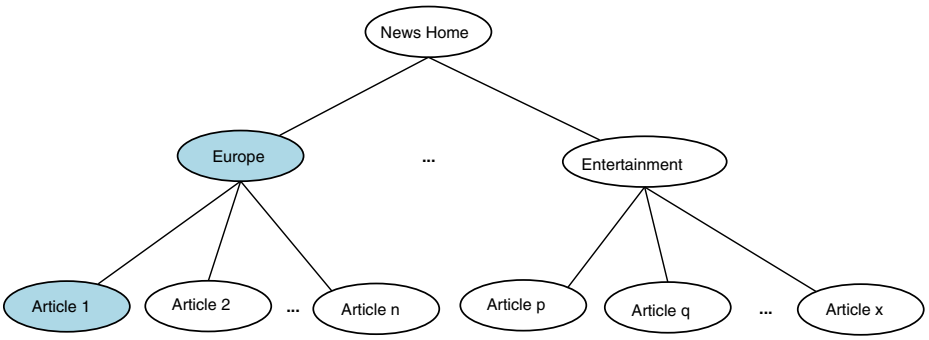


Figure 4: Schematics of the BBC News site.

the participants to customize two or three pages which serve different functions on a single section of their selected site. We collected traces of the operations employed in these customizations for later evaluation. In the validation phase we determined the coverage of the users' customizations by reapplying the customizations to other pages of the same site.

The study employed twenty participants, and their experience with PDAs also varied. Six reported having no experience using a PDA, two reported having low comfort with a PDA, and twelve reported that they are comfortable using a PDA. There is little overlap between participants of this study and those of the first study. Only one participant took part in both studies.

4.2.2 Platform

This study also used the Dell Axim X51v PDA. However, this time, the Minimo Web browser version 0.014 augmented with the PageTailor plugin ran natively on the PDA.

For the purpose of evaluating user customizations on a large number of pages, we transferred traces of the customizations to a desktop workstation where we applied them to the pages. The desktop was equipped with a 3GHz processor and 2GB memory, and ran version 1.5.0.3 of Mozilla Firefox browser.

We also measured the execution times of PageTailor when reapplying customizations to Web pages on the PDA device itself.

4.2.3 Sites Studied

In this study participants customized pages from four sites. The first three were taken from the set of sites evaluated in the longevity study, and they were Amazon, BBC and MSN. For the fourth site, we wanted to study reuse of customizations on a site with user generated content, so we included pages from Flickr.

For Amazon, we asked participants to customize pages from its Music section, specifically, a Jazz CD page and the General Jazz Listings page. For BBC, from the News section, participants customized a news article and the Europe News homepage. For MSN, participants customized a Health news article and the Health News homepage. Finally, for Flickr participants customized a page containing one photo belonging to a Flickr user, a tag page, and the homepage of the same user.

To evaluate the coverage of customizations, we chose pages which serve the same function as those that the participants customized. For Amazon we included CD pages and CD listings, for BBC, we selected news articles and news subsection homepages, for MSN we included health articles and health subsection homepages, and for Flickr we chose photo pages, user tags and sets, and user homepages. For each of these sets, we include all pages with similar function available, the exception being when there are more than twenty, e.g., Amazon Music has a very large number of CD pages. In such case we limit the choice to twenty randomly selected pages for each set. Table 2 lists the pages customized from each site and the test sets to which these customizations were applied. Each site was customized independently by five participants.

5. RESULTS

We begin by presenting results which evaluate the feasibility of customizing Web pages with PageTailor on a PDA. We then present results evaluating the longevity and coverage of user customizations. Finally, we discuss the execution time for reapplying customizations on same and other pages.

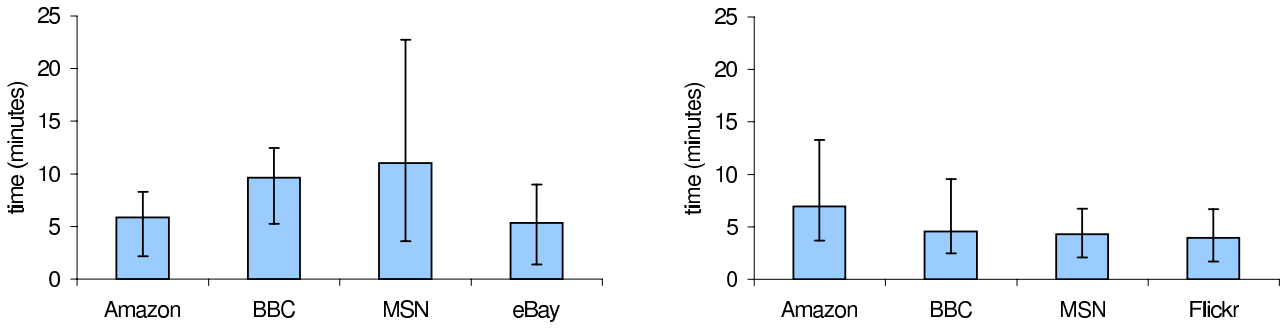
5.1 Feasibility

Of the sixteen participants in the longevity study, fourteen reported being able to implement their paper designs using PageTailor. The two participants unable to fully implement their designs, reported being able to make most of the customizations, however, they encountered challenges in selecting multiple elements and moving those with precision. It is encouraging to see, however, that during their first use, a large majority of users is able to implement designs which are as unrestricted as the paper and pencil allows.

Figure 5 shows, for each of the two studies, the minimum, average and maximum times users spent customizing pages in each site. In the longevity study, the average customization times varied between 5 and 11 minutes. In the coverage study, they varied between 4 and 7 minutes. On average, participants to the coverage study were faster because they were free to change their design objectives as they discovered layouts that were satisfactory to them. This result is somehow surprising as we expected the participants of the longevity study to do better than the other group as the time needed for the cognitive task of choosing the content to customize happened during the paper prototyping phase. Overall, the customization times for both experiments were reasonable, especially when considering the complexity of the pages involved.

Site	Page Customized	Test Set	
		Description	Number of Pages
Amazon	Jazz CD	Jazz CDs	20
	General Jazz CD Listing	Other Jazz CD Listings	17
BBC	News Article	News Articles	20
	Europe News Homepage	News Subsection Homepages	12
MSN	Health Article	Health Articles	15
	Health News Homepage	Health Subsection Homepages	9
Flickr	User Photo	User Photos	17
	User Tag	User Sets & Tags	20
		Other Sets & Tags	20
	User Homepage	Users Homepages	20

Table 2: Sites used in coverage experiment. From each site, participants customized two pages, and these customizations were tested on sets of other pages. For Flickr, participants were asked instead to customize three pages.



(a) Longevity Study

(b) Coverage Study

Figure 5: Minimum, average and maximum customization times for each study.

5.2 Longevity

In this section, we look at *longevity*, the length of time during which customizations originally made by a user remain applicable to a page. Longevity is important because it determines the length of time over which the user benefits from her customization effort. We found from participants' responses that they considered the effort spent customizing to be reasonable if these customizations would last for one month. In the remainder of this section, we show that for the majority of the users, PageTailor is able to preserve most of their customizations for more than 30 days.

Figure 6 shows the results of the longevity experiment for our four homepages, as time progresses. The histogram for each page shows the number of users that considered their customizations still desirable for the new version of the page on a given day. The histogram also shows the number of users who stated that PageTailor had properly applied all (100%) and at least 80% of their customizations to the new version of the page. For example, Figure 6(d) shows that after 32 days all four users still wanted to reapply their customizations to the page. Of these three considered that PageTailor reapplied all of their customization operations correctly and the remaining one considered that more than 80% of his customization operations were properly applied.

The results differ across pages. PageTailor perform best for the eBay homepage as this is the most static of the four pages studied, with layout components changing less frequently. Overall, by day 27, nine out of the sixteen participants still experience full accuracy. Moreover, almost all of our users reported that PageTailor reapplied at least 80% of their customizations for over a month, and in some cases as long as 16 months [2]. At 80% accuracy, we find that failures are either because the original content to which the user performed an operation is no longer present in the page or new content that has been added is left uncustomized. In general, we find that these failures have little impact on the overall layout of the page, and can be easily rectified by adding operations which customize only those objects that were left uncustomized.

5.3 Coverage

In this section we show that customizations can be reused across pages that serve the same function in a given Web site. The implication is that a user only needs to customize a small number of pages in order to browse a site which may contain a very large number of pages.

Figure 7 shows results from the coverage experiment evaluating reuse of customizations across pages which are similar. Each bar shows the breakdown of the success rate with which customizations are reapplied across pages that serve identical functions in each Web site. Success rate measures the fraction of operations from a customization which ap-

[2]The results for the Amazon Web page shows a discrepancy in days 27 and 74. We attribute this discrepancy to one participant who, confused with the instructions, reported his customization to be undesirable yet found it to apply properly.

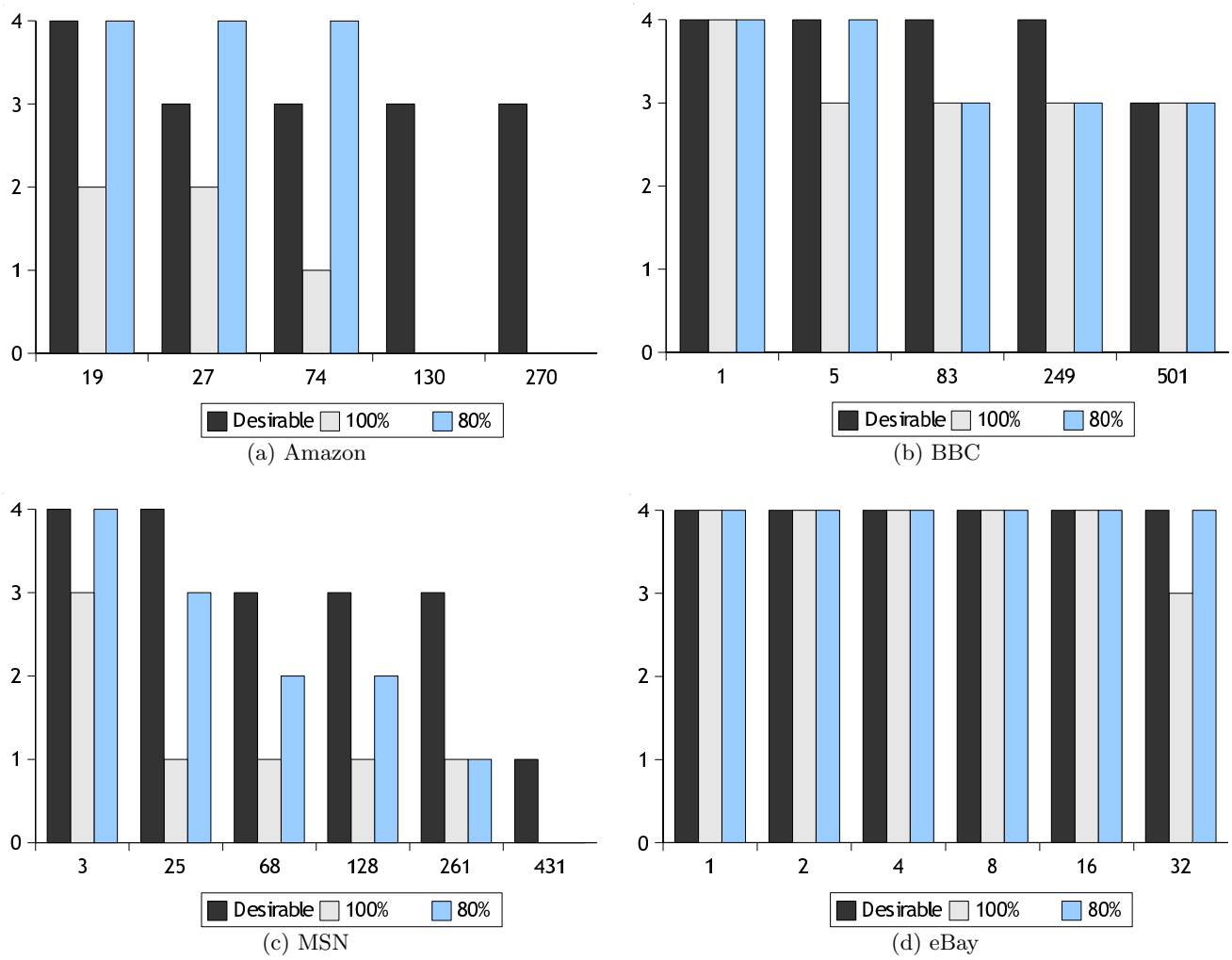


Figure 6: Longevity experiment. The figure shows that for the majority of users PageTailor is able to reapply their customizations accurately for over a month.

plied to a page. For example, Figure 7(b) shows that for 81% of news articles, PageTailor successfully reapplied at least 70% of the users' original customization operations. For all Web sites, PageTailor was able to apply all operations (100% success rate) to between 33% and 80% of pages that serve the same function.

To understand the practical implications of the differences in the success rates, we compared the accuracy perceived by users (as reported in Section 5.2) against the success rates reported by the system for Figure 6. We find that users reported 100% accuracy for pages where the system success rate was as low as 67%. We also analyzed in detail a random sample of pages from each of our sites and found that for success rates above 70%, the failures reported by the system were mainly due to missing content, and very often they were failures to remove those missing objects. For example, for the Amazon site, we observed failures to remove missing promotional banners. The effect of these failures to the overall layout of the page is minimal. Below the 70% success rate, failures become noticeable. There is a significant rate of false negatives, as well as some false positives. At these success rates, the structure of the pages begin to differ. Based on these observations, we consider success rates above the 70% to be accurate enough to provide a benefit.

Overall, for 75% of the pages we considered PageTailor succeeds in applying at least 70% of customization operations. Thus, customizations to one page provide coverage for a large number of pages serving the same function within a Web site. For example, participants that customized 1 CD page could reuse their customizations to browse close to 90% of the Amazon CD collection.

We also explored the application of customizations across pages that fulfill different functions (e.g., CDs and CD Listings), and across sections of large Web sites such as Amazon, where we evaluated how well customizations performed to a CD page apply to a DVD or a book page. As expected, for most pages customizations do not transfer well across functions or across sections, with PageTailor reporting accuracies well below 50%.

5.4 Execution Times

In this section we evaluate the time users will be required to wait for customizations to apply to a page they visit.

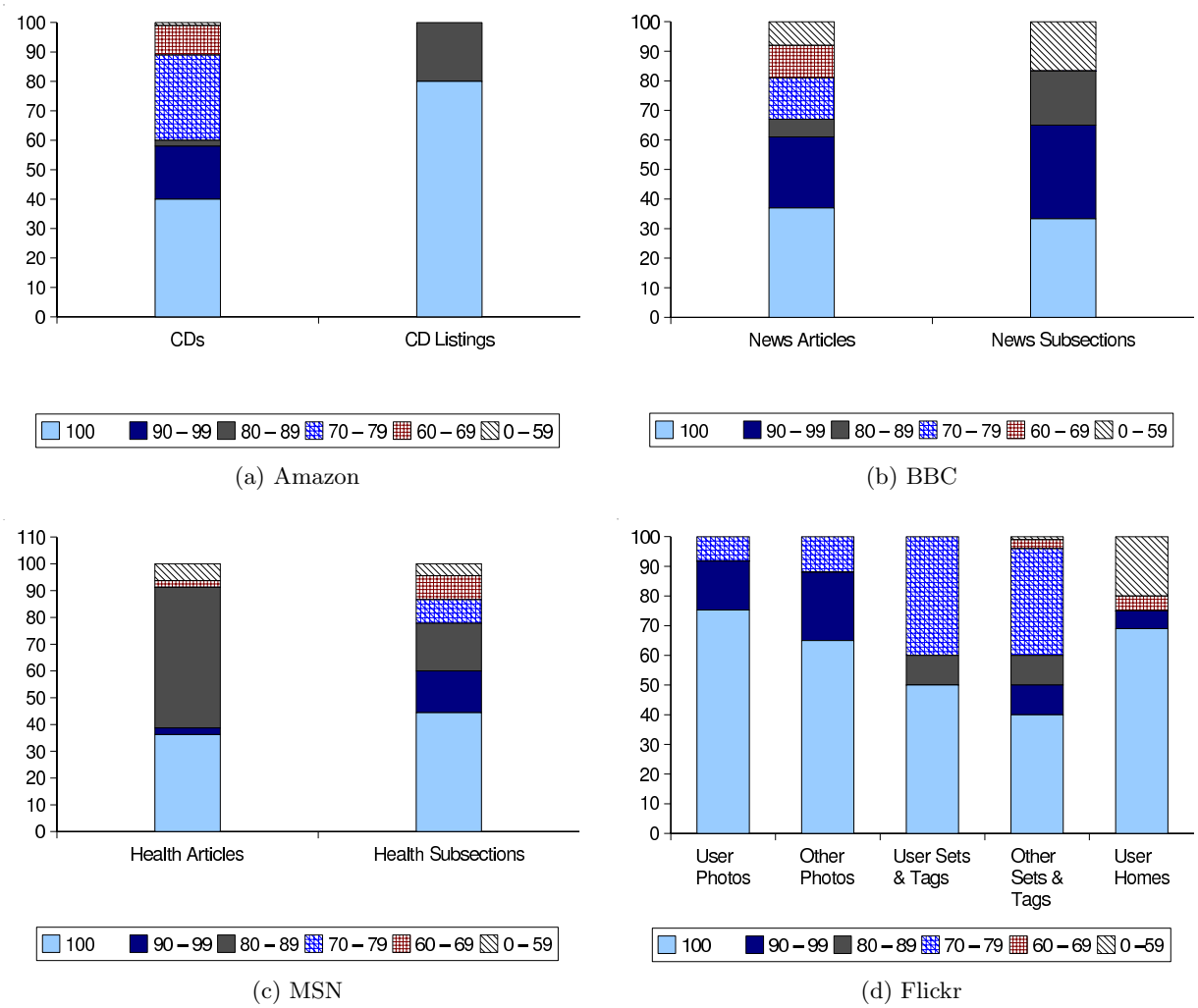


Figure 7: Coverage experiment. The figure indicates that customizations are reusable on pages which serve the same function within a section of a site.

For this evaluation, we select the following Web pages. Three BBC news articles with sizes (number of objects) which approximate the median size of all pages employed in the user study. To evaluate the effects of varying page size on execution time we also select a smaller Flickr page. We then select several customizations from various participants from our studies. One customization has 19 operations, the median size of customizations performed by all participants, and other customizations have smaller and larger sizes, specifically 7, 18 and 41 operations each. This allows us to observe the effects of varying customization sizes.

Table 3 presents execution times for reapplying customizations on a PDA, and a desktop. For comparison, we also present the expected download times for each page over a CDMA 1X cellular network, assuming an effective network bandwidth of 80 kbps. For the execution times on the PDA, we break down the results into two components, recognition time and reapplication time. Recognition refers to the time PageTailor takes to decide whether to apply a given customization to the page loaded or not. Reapplication indicates the time required to apply a customization to a page once they have been matched. We measured these times by loading each Web page and allowing PageTailor to execute. We report average results over five runs. In all cases, standard deviation was under 6%.

In Table 3, results for the first entry (BBC Article 1) show the times taken when loading a page the user customized initially. Since the URL for this page is already associated with the customization, the recognition phase does not involve any structural comparison of the Web page against the customizations on the device. Rather, it only includes the time taken to read the list of customizations from file and comparing the URLs of those customizations against the URL of the page on display. For all other results, customizations are applied to unseen pages, and therefore PageTailor incurs the recognition cost.

From Table 3, we expect the execution time for the average user to take between 12 and 29 seconds. The execution times are non-negligible, and this is in part because Minimo remains in its early stages of development as indicated by its version number (0.014), and as such remains to be optimized for PDAs. From Table 3, we can see that even without search (BBC Article 1) reapplication time is already 12 seconds. To show that these execution times could be better

Page	Page Size (objects)	Customization Size (operations)	Success Rate (%)	Recognition Time (s)	Reapplication Time (s)	Total (s)	Desktop (s)
BBC Article 1 (self)	1070	19	100	0.0	12.0	12.0	0.3
BBC Article 2 (good match)	1072	7	100	16.0	8.2	24.2	0.4
	1072	19	100	16.2	12.4	28.6	0.7
	1072	41	100	17.4	59.4	76.8	1.5
Flickr Photo (good match)	829	7	100	12.4	5.8	18.2	0.3
BBC Article 3 (poor match)	1146	7	71.43	17.8	27.8	45.6	0.9
	1146	19	36.84	19.2	55[a]	19.2	0.4

[a]At 36.84% success rate, the reapplication process does not normally execute as the recognition fails to match the page to the customization. However, we force its execution to evaluate the usefulness of a restricted search.

Table 3: Execution times. The table shows that execution time is dependent on the number of customization operations, page size and success rate.

on the PDA, we have included in the table results showing the total execution times of the same prototype on a desktop. This desktop is equipped with a 2.4GHz processor and a 512MB memory. The ratio between the processor speeds of the two devices is less than 4, however the ratio between execution times consistently exceeds 40. Moreover, as anecdotal evidence, we note that starting Minimo on the PDA requires approximately 23 seconds, whereas Pocket Internet Explorer requires only 3. Similarly the Opera browser also takes under 9 seconds on the same device.

The results in Table 3 also show that reapplication time is affected more significantly than recognition time by the size of the customizations, the complexity of the page and, the success rate with which a customization applies to a page. This is expected because the restricted search employed during recognition limits the search space to one level for each path.

In summary, the results above show that execution times when reusing customizations with the current prototype will, for the average user, vary between 12 and 29 seconds. These times are likely to drop, however, as the Minimo code matures. Moreover, because the recognition phase generates a regular expression on success, its costs need not be incurred every time a user visits a new page. For pages where URLs are similar, except perhaps with differing suffixes, a regular expression generated for one would also match the other. The results also show that execution times depend on the number customization operations, complexity of the Web pages and number of failed operations.

5.5 Discussion

As we indicated in Section 2.2.2, because the recognition phase employs a restricted version of the search algorithm, we expect a larger rate of false negatives to occur. For all pages and customizations used in the coverage study we compared the success rates of PageTailor under both the restricted and the unrestricted version of the search algorithm. For 14% of the pages to which the unrestricted algorithm reports success rates above 70%, the restricted algorithm reports success below that mark. The implication is that PageTailor will fail to automatically match customizations for 14% of appropriate pages, a sacrifice made to improve responsiveness. In such cases, however, users can explicitly associate existing customizations with the pages.

There is an inherent limitation to the size of our controlled experiment, both in the number of web sites and users involved. As a result, we cannot claim that the results presented in this paper are representative of unrestricted browsing behavior; however, based on our positive experience adapting, what by any measure constitute fairly complex pages, we are optimistic about the expected performance of PageTailor in real world conditions.

6. RELATED WORK

There has been significant research and industry effort related to adaptation of Web pages for consumption on devices with small screens. Some content providers have manually authored versions of Web pages for specific devices they wished to support [20]. Others employ a "lowest common denominator" approach, in which they design pages with only the features appropriate for the least capable device they wish to support. This is commonly achieved with technologies such as WML, cHTML or even low graphics or text-only versions of HTML pages. These solutions are inadequate as designing for each device incurs significant overhead and does not scale, and the lowest common denominator is inappropriate for users of devices with support for more features than the common denominator. For example, users of PDAs capable of displaying high resolution images and processing rich content are unhappy to be restricted to text-only versions of news articles. In addition, even users of similar devices may be interested in different content from a Web page. For example visitors to the BBC News homepage are often interested in different news topics. Thus, the layout of the page should facilitate each user's access to the content of their interest, a requirement which is clearly not met with these approaches.

There has also been research on systems which adapt Web pages automatically [7, 9, 8, 3, 11, 24, 29, 17]. Text summarization is a technique which allows automatic summarization of Web pages and progressive disclosure of details. In Power Browser [9] and Digestor [3], for example, users are presented with a summary page containing links to different units of the original page. However, this technique requires that users navigate back and forth between the summary page and the sectioned blocks and may result in loss of contextual information about the content.

A similar approach for automatically adapting Web pages is page fragmentation [11, 24, 5, 1, 17]. Based on Web page structure, Chen et al. [11] and Milic-Frayling et al. [24] partition Web pages into fragments or units that can fit in the small screen, and present a two-level hierarchy to navigate within the page. However, this technique still requires users to navigate back and forth within the hierarchy. Borodin et. al. [5] improve on page fragmentation by determining automatically the most relevant fragment when a user follows a

link, and providing that fragment first. However, this technique may be prone to error as it guesses what the user's interest is within a page. In contrast, in our approach users make an explicit choice of what is of interest to them. Moreover, with the above approach, users interested in multiple content from a Web page (e.g. sports and weather from a news portal) may have to manually navigate through many fragments.

Minimap [29] scales content of Web pages to automatically improve the display of Web pages on small screens. Minimap scales down non-textual elements such as images so they fit within the viewport of the device's browser while maintaining the dimensions of textual elements. Because this approach does not take the user's interest into consideration, the user may still need to scroll significantly. pTHINC [22] uses a thin-client approach to perform screen scaling on a server, and allows users to zoom into areas of the page. Screen scaling makes content of large pages too small for comfortable reading on a small screen, and an unscaled screen requires that users scroll considerably.

There are other systems that automatically adapt content based on a combination of rules and constraints [7, 23, 30, 31]. Unfortunately, content providers cannot be expected to provide constraints or rules for every data object, as this would not be very different from supplying customized content for every client type. As a result, small sets of rules apply to broad sets of content (e.g., all JPEG images are adapted the same way independent of their value to the user).

End-user customization has been used previously on the desktop environment to enable personalization of Web pages. Web pages such as Google Personalized Homepage, allow users to choose content units to be displayed in the page, and select the areas of the page in which these units should appear. However, support for such technique is limited to a few sites, the customization options offered are restricted, and users need to be familiar with the different interfaces provided by each site.

Closer in spirit to our approach are Chickenfoot [4], Greasemonkey [19] and Platypus [27]. Chickenfoot and Greasemonkey provide programming interfaces for the desktop through which users specify actions to be performed on page content, such as removing elements. Platypus adds a What You See Is What You Get (WYSIWYG) interface to Greasemonkey, allowing desktop users to customize graphically. However, none of these systems employ algorithms to ensure customizations are long-lived, nor are able to automatically reuse customizations on similar pages. Moreover, these systems have not been explored in the context of browsing on small screens.

In the mobile environment, Baudisch et al. [2] explore the use of a marquee menu to adapt the view of Web pages on small screens. In this approach, users can collapse and expand content units based on their interest. Because this technique uses the same mechanism as [11] to partition Web pages, user adaptations are limited to automatically identified fragments. Moreover, the focus of that work is on the interaction technique to allow users to specify customizations. In contrast, the focus of our paper is on reuse of customizations on dynamic pages as their content changes over time, and across similar pages of a Web site. We could support the use of a marquee menu, in addition to the current interface provided by PageTailor.

There has also been research effort in adapting Web content to address other limitations of mobile devices [14, 16, 25, 26, 15]. These systems address problems that are orthogonal to the screen constraints, such as limited computing power and bandwidth, and could be used conjunction with REUC.

7. CONCLUSIONS

We proposed Reusable End-User Customization (REUC), a technique for adapting Web pages to small devices. We presented PageTailor, a PDA-based REUC implementation and showed that it enables customizations made by users to a Web page to be reapplied in future visits to the page, even as content changes, and across similar pages within a site.

We showed that it takes users an average of 10 minutes to customize complex Web pages using PageTailor, and that because the structure of Web pages is stable enough, these customizations can be reapplied for over one month, and in some cases for up to 16 months. We also showed that because commercial web sites use a relatively small number of templates to generate their content, a user only needs to customize a limited number of pages to browse a large site. Overall, we found that customizations made to one page apply to at least 75% of other pages which serve the same function within the site.

8. ACKNOWLEDGMENTS

This research was supported by Bell University Labs under grant number 302814, and by the Canadian Foundation for Innovation and the Ontario Innovation Trust under grant number 7739.

9. REFERENCES

[1] S. Baluja. Browsing on small screens: Recasting Web-page segmentation into an efficient machine learning framework. In *Proceedings of the 15th International World Wide Web Conference (WWW2006)*, Edinburgh, Scotland, UK, May 2006.

[2] P. Baudisch, X. Xie, C. Wang, and W.-Y. Ma. Collapse-to-Zoom: Viewing Web pages on small screen devices by interactively removing irrelevant content. In *Proceedings of the 17th Symposium on User Interface Software and Technology (UIST '04)*, Santa Fe, NM, USA, Oct. 2004.

[3] T. W. Bickmore and B. N. Schilit. Digestor: Device-independent access to the World Wide Web. In *Proceedings of the 6th International World Wide Web Conference (WWW6)*, Santa Clara, CA, USA, Apr. 1997.

[4] M. Bolin, M. Webber, P. Rha, T. Wilson, and R. C. Miller. Automation and customization of rendered Web pages. In *Proceedings of the 18th Symposium on User Interface Software and Technology (UIST'05)*, Seattle, WA, USA, Oct. 2005.

[5] Y. Borodin, J. Mahmud, and I. Ramakrishnan. Context browsing with mobiles - when less is more. In *Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys 2007)*, San Juan, PR, USA, June 2007.

[6] T. M. Breuel. Information extraction from HTML documents by structural matching. In *Proceedings of*

the 2nd International Workshop on Web Document Analysis (WDA2003), Edinburgh, Scotland, UK, Aug. 2003.

[7] K. Britton, R.Case, A. Citron, R. Floyd, Y. Li, C. Seekamp, B. Topol, and K. Tracey. Transcoding: Extending e-business to new environments. *IBM Systems Journal*, 40(1):153–178, 2001.

[8] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke. Seeing the whole in parts: Text summarization for Web browsing on handheld devices. In *Proceedings of the 10th International World Wide Web Conference (WWW10)*, Hong Kong, China, May 2001.

[9] O. Buyukkokten, H. Garcia-Molina, A. Paepcke, and T. Winograd. Power Browser: Efficient Web browsing for PDAs. In *Proceedings of the Conference on Human Factors in Computing Systems 2000 (CHI'00)*, The Hague, The Netherlands, Apr. 2000.

[10] S. S. Chawathe. Comparing hierarchical data in external memory. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB'99)*, Edinburgh, Scotland, UK, Sept. 1999.

[11] Y. Chen, W.-Y. Ma, and H.-J. Zhang. Detecting Web page structure for adaptive viewing on small form factor devices. In *Proceedings of the 12th International World Wide Web Conference (WWW2003)*, Budapest, Hungary, May 2003.

[12] T. Dalamagas, T. Cheng, K.-J. Winkel, and T. Sellis. A methodology for clustering XML documents by structure. *Information Systems*, 31(3), May 2006.

[13] D. de Castro Reis, P. B. Golgher, and A. S. da Silva. Automatic Web news extraction using tree edit distance. In *Proceedings of the International World Wide Web Conference (WWW2004)*, New York, NY, USA, May 2004.

[14] E. de Lara, D. S. Wallach, and W. Zwaenepoel. Puppeteer: Component-based adaptation for mobile computing. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS '01)*, San Francisco, CA, USA, Mar. 2001.

[15] Y. Dotsenko, E. de Lara, D. S. Wallach, and W. Zwaenepoel. Extensible adaptation via constraint solving. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002)*, Callicoon, NY, USA, June 2002.

[16] A. Fox, S. D. Gribble, E. A. Brewer, and E. Amir. Adapting to network and client variability via on-demand dynamic distillation. *SIGPLAN Notices*, 31(9):160–170, Sept. 1996.

[17] Google Mobile Search. `http://www.google.com/xhtml`.

[18] K. Goto. Brand value and the user experience. *Digital Web Magazine*, July 2004.

[19] Greasemonkey. `http://greasemonkey.mozdev.org/`.

[20] M. Jones, G. Marsden, N. Mohd-Nasir, K. Boone, and G. Buchanan. Improving Web interaction on small displays. In *Proceedings of the 8th International World Wide Web Conference (WWW8)*, Toronto, Canada, May 1999.

[21] T. Kamba, S. A, Elson, T. Harpold, T. Stamper, and P. Sukaviriya. Using small screen space more efficiently. In *Proceedings of the Conference on Human Factors in Computing Systems 1996 (CHI '96)*, Vancouver, Canada, Apr. 1996.

[22] J. Kim, R. A. Baratto, and J. Nieh. pTHINC: A thin-client architecture for mobile wireless Web. In *Proceedings of the 15th International World Wide Web Conference (WWW2006)*, Edinburgh, Scotland, UK, May 2006.

[23] W. Y. Lum and F. C. Lau. A context-aware decision engine for content adaptation. *IEEE Pervasive Computing*, 1(3):41–49, July 2002.

[24] N. Milic-Frayling and R. Sommerer. SmartView: Flexible viewing of Web page contents. In *Proceedings of the 11th International World Wide Web Conference (WWW2002)*, Honolulu, HI, USA, May 2002.

[25] D. Narayanan, J. Flinn, and M. Satyanarayanan. Using history to improve mobile application adaptation. In *Proceedings of the 3rd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2000)*, Monterey, CA, USA, Dec. 2000.

[26] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker. Agile application-aware adaptation for mobility. *Operating Systems Review (ACM)*, 51(5):276–287, Dec. 1997.

[27] Platypus. `http://platypus.mozdev.org/`.

[28] T. Richardson, Q. Stafford-Fraser, K. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1), Jan/Feb. 1998.

[29] V. Roto, A. Popescu, A. Koivisto, and E. Vartiainen. Minimap - a Web page visualization method for mobile phones. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 2006)*, Montreal, Canada, Apr. 2006.

[30] B. N. Schilit, J. Trevor, D. M. Hilbert, and T. K. Koh. Web interaction using very small internet devices. *IEEE Computer*, 35(10):37–45, 2002.

[31] J. R. Smith, R. Mohan, and C.-S. Li. Transcoding internet content for heterogeneous client devices. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '98)*, Monterey, CA, USA, May 1998.

[32] The Wayback Machine. `http://web.archive.org`.

[33] G. Valiente. An efficient bottom-up distance between trees. In *Proceedings of the International Symposium on String Processing and Information Retrieval (SPIRE 2001)*, Laguna de San Rafael, Chile, Nov. 2001.

[34] WAP Forum. Wireless Application Protocol: Wireless Markup Language specification. `http://www.wapforum.org/what/technical/wml-30-apr-98.pdf`, Apr. 1998.

[35] World Wide Web Consortium. Cascading Style Sheets, Level 2: CSS2 specification. `http://www.w3.org/TR/REC-CSS2/`, May 1998.

[36] World Wide Web Consortium. Compact HTML for Small Information Appliances. `http://www.w3.org/TR/1998/NOTE-compactHTML-19980209/`, Feb. 1998.

[37] World Wide Web Consortium. Document Object Model (DOM) Level 2 Core Specification. `http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/`, Nov. 2000.

[38] J. T. Yao and M. Zhang. A fast tree pattern matching algorithm for XML query. In *Proceedings of the International Conference on Web Intelligence (WI'04)*, Beijing, China, Sept. 2004.