The current issue and full text archive of this journal is available at
www.emeraldinsight.com/1744-0084.htm

Received 4 February 2013
Revised 22 May 2013
Accepted 24 May 2013

Achieving "One-Web" through customization and prioritization

Nassiriah Shaari
University Utara Malaysia, Sintok, Malaysia, and
Stuart Charters and Clare Churcher
Lincoln University, Lincoln, New Zealand

Abstract

Purpose – Accessing web sites from mobile devices has been gaining popularity but may often do not give the same results and experiences as accessing them from a personal computer. The paper aims to discuss these issues.

Design/methodology/approach – To address these issues, the paper presents a server-side adaptation approach to prioritising adaptive pages to different devices through prioritisation system. The prioritisation approach allows users to prioritise page items for different devices. The prioritisation engine reorders, shows, and removes items based on its priority set by users or developers.

Findings – With this approach, the overall web page's structure is preserved and the same terminology, content, and similar location of content are delivered to all devices. A user trial and a performance test were conducted. Results show that adaptive page and prioritisation provides a consistent and efficient web experience across different devices.

Originality/value – The approach provides advantages over both client-side and proxy and has conducted significant experimentation to determine the applicability and effectiveness of the approach.

Keywords Advanced web applications, Applications and standards, Mobile computing for the internet, Performance of web applications, Web semantics architectures, Applications and standards

Paper type Research paper

1. Introduction

Producing and maintaining multiple versions of a web site for different platforms is expensive, time consuming and error prone (Byron, 2011). Visitors can become frustrated navigating different versions of a web site if the site layout and structure is not consistent (Compuware, 2011). The "One-Web" W3C (2008) approach to web development is a single web site that provides a consistent experience across different platforms.

To achieve a "One-Web" web site, we present an approach which allows the customised delivery of a single site to different platforms based on user preferences. The primary driver of our work is the display of web sites on mobile devices as they continue to be the largest growing platform for accessing the internet. This is particularly the case in economically less developed countries where mobile communication infrastructure is being deployed more rapidly, with higher data rates and greater reliability than traditional telecommunications infrastructure (ITU, 2011; Ohri, 2011).

We have trialled our adaptation engine against a social networking site, Facebook, as it has a large user base and currently maintains multiple versions of the web site for different platforms.

International Journal of Web Information Systems
Vol. 9 No. 4, 2013
pp. 264-278

1744-0084
DOI 10.1108/IJWIS-02-2013-0002

In this paper, we outline previous approaches to prioritisation and customization of web site content before describing the implementation of our adaptation engine. We also describe the user trials of a site designed in a "One-Web" manner and delivered using our adaptation engine against the traditional multi-version implementation.

2. Background

There are a number of prior approaches to the issue of web content adaptation. These approaches can be characterised by where they take place (server-side, through a proxy, client-side) as well as by how the adaptation is carried out. Customization occurs when users alter aspects of the delivered page such as the position, appearance, or the type or amount of content. Personalization is similar but it is the system which determines the changes based on a user's historical preferences (Nielsen, 2009).

Some common examples of customization include: moving, adding, or removing blocks of content on a page (e.g. Yahoo! and iGoogle); modifying preferences such as menu options, displaying images or including widgets; or changes to colours and fonts.

Several studies have investigated approaches to customizing web pages, which allows users to determine content or items of interest for mobile devices.

PROTEUS (Anderson *et al.*, 2001) makes use of syntactic translation using the information seeking routes of the current and previous visitors to shorten navigation paths through web sites and remove content from pages. This approach reduces the amount of data required to be downloaded by a mobile web visitor and also the number of clicks required to find information. Proteus runs on a web proxy and acts as an intermediary between the site visitor and the site itself. Proteus builds a set of tuples as a state search space of the web site and uses a utility function to evaluate each state. A search can then be conducted to find the best state for a given visitor given their prior history and the history of other users. The results of the search return a modified page which has shortcuts to information the visitor is likely to request. An evaluation of Proteus showed that whilst the prototype implementation was not suitable for real time personalization (taking minutes to return a customised page) off-line pre-personalized pages improved the user experience for mobile web visitors by reducing the time to find information they required.

Highlight (Nichols and Lau, 2008) allows users to create a mobile version of existing web sites on a desktop. Highlight splits web site pages into multiple pages or "pagelets", based on tasks users demonstrate on a desktop version. This helps to reduce individual page size and download time on the mobile devices. Highlight allows users to test their customised applications. If there are missing tasks, users can add them to the application. It is suitable for sites with tasks that are frequently performed by users. These were all done through Highlight Designer, a proxy-side adaptation system that adapts the page content and layout. Highlight (through its Highlight Designer) is a proxy-side adaptation system that adapts both the page's content and layout. An informal user study conducted with three users showed that Highlight Designer is easy to use even by novice users. An empirical evaluation comparing performing the same set of tasks using Highlight to browse with desktop browser features showed that Highlight application reduces the number of interactions.

PageTailor (Bila *et al.*, 2007) is a reusable end-user customization tool for the mobile web, which allows customization to be made on a user's browser via a plug-in. It is targeted to PDA uses and implemented on the Minimo web browser. It allows users to adapt the layout of web pages in which users can move, remove, or resize page items. The customization is stored on the device's persistent storage, thus allowing the same customization to be applied on each visit until the cookies expire. PageTailor first loads the whole page (except images, which are loaded after the customization rules/preferences are applied) on the browser and applies the customization on the DOM tree. The customization is done once or for a minimal number of pages and is long lasting. User studies conducted in lab experiments showed that the customization lasts at least a month and is applicable to more than 75 per cent of pages with similar structure. Other pages with similar page structure will also be customized based on the earlier customization. However, users may want to have different things available on different sites regardless of them having the same structures. Low-end and mid-range devices may not support large storage. In addition, users need to install a specific browser and plug-in to use the adaptation engine.

A customizable mobile device oriented web data extraction scheme has been proposed by Xiao *et al.* (2008, 2009). The customization engine resides in a proxy server, as they found that most users did not agree to install a plug-in on the client device. Based on the DOM tree, the system splits web page into two layers – a link block and the corresponding content block. The engine customizes and personalizes content based on a user's browsing history. Similar to PageTailor, it will deliver content that users want and hide the rest; but the adaptation and customization is done using AJAX instead of a browser plug-in. The customization and personalization only works if the mobile device supports AJAX as part of the system requires AJAX code to run on the client-side. Similar to PageTailor and Highlight, the customization process involves content and layout adaptation. The system was evaluated for its efficiency (speed) and effectiveness (accuracy) of the splitting mechanism, which showed the system's efficiency and effectiveness are acceptable.

A toolkit to personalize web pages for mobile devices has been proposed by Kao *et al.* (2009). The concept of the adaptation engine is also similar to PageTailor by Bila *et al.* (2007) but it requires users to set their preferences through a desktop computer instead of on the client device. The system allows users to determine blocks of content on a web page to be displayed or retained after customization. The mobile code for the personalisation, called PageTailor and is implemented in JavaScript (N.B. this PageTailor is not the same as the PageTailor developed by Bila *et al.* (2007)). User preferences for the page are identified by the XPath expressions of the object or content. Users can specify the preference by selecting blocks of items to retain and altering their order on the customized page. This is done using the visual manipulation provided by PageTailor code, which manipulates the page through the DOM interface. To access the customized page, users need to configure their browser to go through a proxy first. Filtering the unwanted content reduces the page length, which in turn reduces scrolling. No detailed evaluations were reported. However, simple tests conducted showed that the system worked consistently across the two browsers tested (Internet Explorer and Firefox) and was stable as customization based on users' preferences produces the same blocks of content on different days.

Proteus (Caetano *et al.*, 2007) (note this Proteus is different from that of Anderson *et al.* (2001)) is a proxy side dynamic adaptation system adapting web pages on small screen devices based on user preferences stored in a profile. The HTML code is validated and unimportant information such as comments are removed. The DOM structure is used to represent the page in memory for further actions. For each block of summarised text, a more link is appended at the end of the summary that links to the original text. Proteus allows users to filter images or figures, so only textual information is displayed. It allows users to determine the compression rate for images if they decide to display the image. It also adapts a page into a thumbnail or conventional HTML text. All these are done based on users' preferences specified on a form on the web. Preliminary results showed that the system worked as intended. No formal user studies or empirical evaluation was reported.

These existing approaches are limited in their applicability and generalizability as they often rely on specific client software or technology, e.g. a specific browser or Javascript capability. This limits the extent of their potential uptake.

3. User requirements

In order to develop our approach we conducted a number of user surveys and a user trial. Our survey (Shaari, 2013) showed that the use of mobile devices for accessing the web was increasing but was not yet widespread. Issues that users reported with accessing web sites on mobile devices included:

- infrastructure issues;
- slow downloads;
- unstable connections;
- design issues;
- jumbled page content;
- long pages requiring lots of scrolling; and
- difficulties viewing images.

In addition to the issues with accessing web sites on mobile devices we also surveyed the types of web site most commonly accessed both on desktop and mobile devices. The most frequently accessed type of web site on both classes of devices were social networking sites, of these, Facebook was the most common. For this reason the Facebook web site was selected as an exemplar for our study.

Facebook maintains two sites: one for desktop machines and one for mobile devices. A user trial comparing these two sites (Shaari, 2013) found that users were confused and frustrated with inconsistent layout, functionality and navigation between the sites. Based on the results of those trials we set out to develop an approach which would allow a single web site to:

- ensure a similar page structure is delivered on all devices;
- ensure consistency of terms used and the location of items on all devices;
- minimise items displayed and reduce navigation within a page; and
- support users visiting the site on multiple different devices.

4. Our approach

Our initial user trial (Shaari, 2013) showed that users often had different use-cases for a site depending on which device they were using to access the site. The users wanted to access different items on the site using their desktops and mobile devices. We therefore developed an approach where all items on the master page can be associated with a level of prioritisation. In this section we discuss how we achieved this.

Our approach is to allow the priority of *div* elements in a HTML page to be set by the developer or the user. These priorities can be customised for each device or class of device. We also propose the use of a cut-off value. The cut-off value is designed to remove low priority content and therefore reduce page size, and download times. We anticipate that the cut-off value will be of particular use for small screen devices and those with limited network connectivity. If the cut-off value is used, we recognise that it may be important to access the removed content occasionally and therefore propose the use of a < more. . . > link to allow the user to access the page with no content removed.

To achieve this we augment the *div* HTML tags of a page with a rank attribute. This allows developers and users to specify priorities for different devices. We recommend that developers provide default priorities for different classes of device and users can alter these individually if they choose. We developed a server-side database to store the default and user defined prioritisations. A significant component of realising our approach is the prioritisation engine which modifies the page to reflect user priorities before delivering it to the remote device.

5. Specifying priorities for page elements

In (X)HTML the *div* tag is the structural element used to separate the page into different sections and to control page layout. This is common practice for many web sites and is employed in sites such as Yahoo![1], Google[2], and Stuff[3].

In our approach we categorise *div*s into two categories, prioritisable and non-prioritisable. Prioritisable *div*s are those which have an attribute *id*, and are thus uniquely identifiable. Non-proritisable *div*s are those which are not uniquely identified. Each *div* could also have *div*s or other elements nested within it. The attribute *id* acts as a unique identifier for the item within a page. Non-prioritisable *div*s will remain in the prioritised page and will assume higher priority and be displayed first among its prioritisable siblings. The distinction between prioritisable and non-prioritisable items allows the developer control over which items can be re ordered.

In order to manipulate and prioritise the *div*s, we introduce a new attribute *rank* which is added to the attribute list of a *div*. The *rank* attribute is processed by the prioritisation engine but does not impact the display or rendering of HTML by browsers allowing developers to retain their existing development tools and workflow in developing web sites.

The values for the pair of *id* and *rank* for each page element are stored in a database on the server for each user and for each device. A cut-off value can also be stored for each user and device. This is the maximum number of page elements that will initially displayed on the prioritised page. The remaining elements will be available via a < more. . . > link.

Developers can determine default values of cut-off and element ranks for each class of device. If this is done carefully the need for individuals to select and store their own preferences will be greatly reduced. Users who wish to modify their preferences will

need to be identifiable; typically this would be through a registration and authentication process. Other users will get the default options.

6. Prioritisation engine

The prioritisation engine is a core component of our approach. The engine reads the rank augmented HTML page, the preferences stored for the page and device and then performs the prioritisation and outputs the final result for delivery to the user's browser. The full process is shown in Figure 1 and described in more detail below.

When a page is requested via the prioritisation engine the requesting device is detected using the user agent string. If a user is identifiable then the user's preferences are retrieved from the database, if not the default priorities are retrieved. The HTML page is loaded from disk and preferences are associated with page elements via the attribute *id* in the *div* element. The prioritisation algorithm is then run. If a cut-off has been set then items with a priority at or below the cut-off are removed and the < more... > link added to the page to allow them to be displayed if required. An XSLT processor is then used to re-order and transform the remaining elements on the page. Items are sorted based on the value of the attribute *rank* in ascending order. For each level of the page tree *div*s are sorted relative to other *div*s at that level. Thus, a

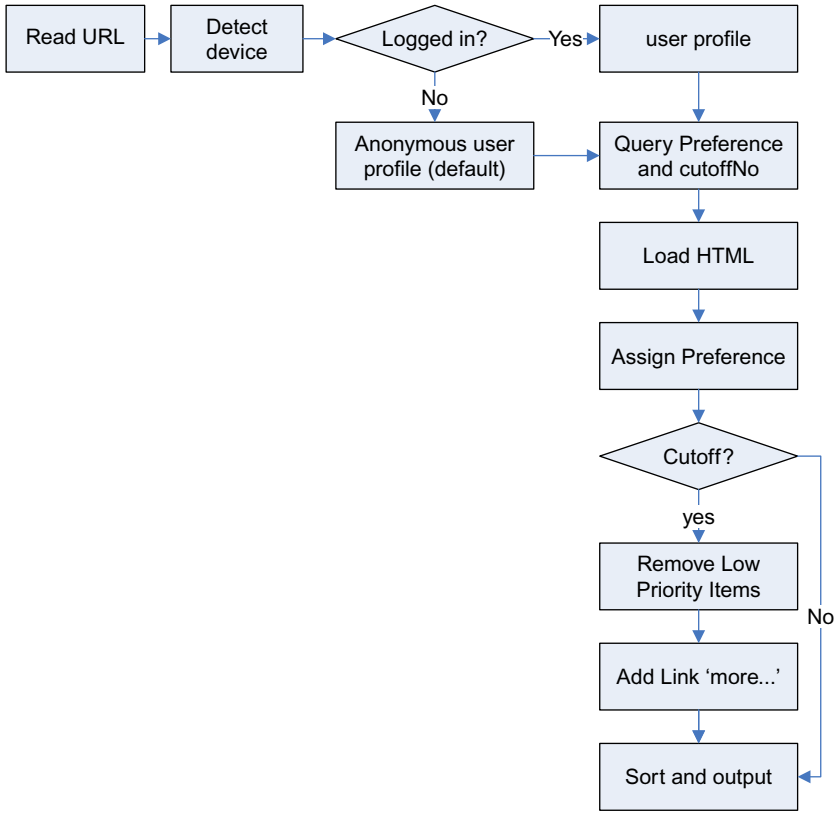
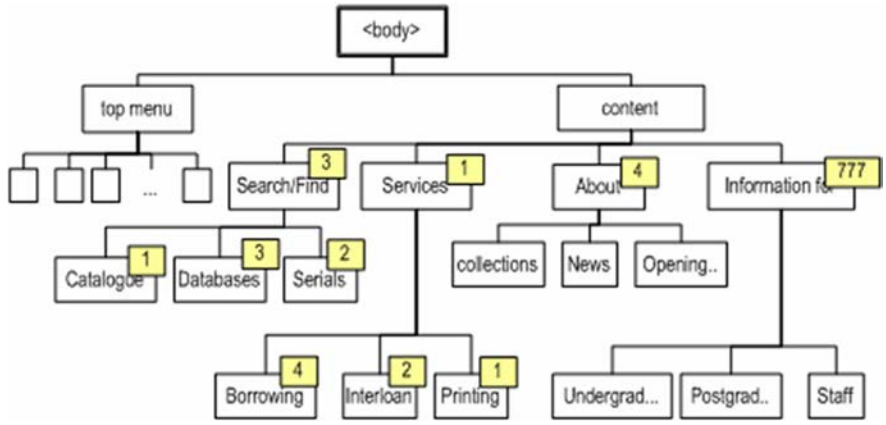


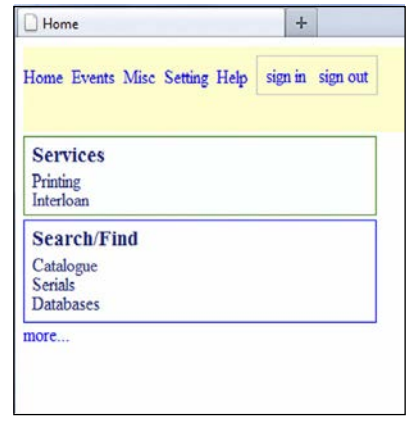
Figure 1.
Tasks performed by the prioritisation engine

child node with priority 1 in a parent with priority 4 will appear below a child with priority 2 in a parent with priority 3. Items with highest priority (value of 1) are displayed first at that level. If multiple *div*s at the same hierarchical level share the same preference, they are displayed in the order they initially appear in the HTML tree. All other elements are copied unchanged. An example of how this works can be seen in Figure 2(a)-(c), Figure 2(a) shows the structure of the original page after priorities have been assigned but before the page has been re-ordered. Figure 2(b) shows how the page



(a)

(b)



(c)

Figure 2.
Original page structure and prioritised results

Notes: Page structure with assigned priorities shown in yellow boxes; original page; prioritised page with cut-off of 3 set. Lower priority elements can be accessed via the <more…> link-

renders in a browser without being prioritised and Figure 2(c) shows the result after the page has been processed by the prioritisation engine.

The prioritisation engine was implemented using PHP5 and employing the XSLT processor libraries for manipulating page structure. User preferences and device identification were stored in a MySQL database. The system was deployed on an instance of Microsoft Internet Information Server 7 running on Windows 2008 R2 inside a virtual environment.

7. Evaluation

Having developed the prioritisation approach and implemented the prioritisation engine we designed a two-fold evaluation. The first aspect of the evaluation is a user evaluation to compare the utility and usability of the prioritised version of a web site with a more traditional approach of developing multiple sites to satisfy the requirement of each class of device. The second component of the evaluation was a performance evaluation to examine the scalability of the solution and the impact of page delivery times.

Our earlier surveys had shown that the differences between the two versions of Facebook (desktop and mobile) had caused users problems. A mock Facebook page was prepared using the *div* structure described in Section 4. It was designed to closely resemble the desktop Facebook page. Our earlier surveys (Shaari, 2013) showed a clear preference for items thought most useful for mobile users and we used these to determine ranks for page elements. Figure 3 shows the actual mobile Facebook page along with the result from putting our mocked up page through the prioritisation engine.

8. Method

A trial was carried out to compare the actual and Facebook mobile page and our prioritised mobile page as shown in Figure 3. We selected tasks from a set of frequent tasks identified in Shaari (2013) which met one of the following criteria:

- *Task 1*. Located at the top of both pages (update status).
- *Task 2*. Difficult to find in Facebook mobile (find information about a friend).
- *Task 3*. Hidden behind "more . . . " link in the prioritised mobile (display photo album).

Full details of the trials can be found in Shaari (2013). Figure 4 shows an overview of the trial process.

The background questionnaire asked about the participant's use of Facebook and experience with mobile devices and touch screens. An initial pilot revealed that participants were able to use real phones better than emulator software. A touch screen mobile phone was used for the actual trial and a familiarisation session was provide to ensure the participants were familiar with the processes (e.g. scrolling, selecting) they would require to carry out the tasks.

A within-subjects design was used for the trials (Nielsen, 1993). 14 regular Facebook users were recruited for this trial from the Lincoln University community. All participants performed the tasks on both the actual Facebook mobile site and the prioritised mobile version. A second trial involved a different five participants carrying out the same tasks using the actual and mocked up desktop versions.

To lessen the learning transfer effect, a counterbalancing approach (Sharp *et al.*, 2006) was used. The orders of versions of pages participants trialled were alternated.

Figure 3.
Actual Facebook mobile page and our mocked up page after prioritisation

After performing the three tasks on each version, participants were asked what they liked and disliked about the site, what tasks proved difficult and what other comments they might have. After participants completed the set of tasks on both versions they were asked which version they preferred and about the factors that influenced their choice.

Then, a semi-structured interview was conducted. Participants were asked questions related to their opinion about the overall concept of prioritisation.

9. Results

Times were recorded for participants completing the three tasks on the prioritised version and on the actual Facebook mobile site. The means and one standard deviation

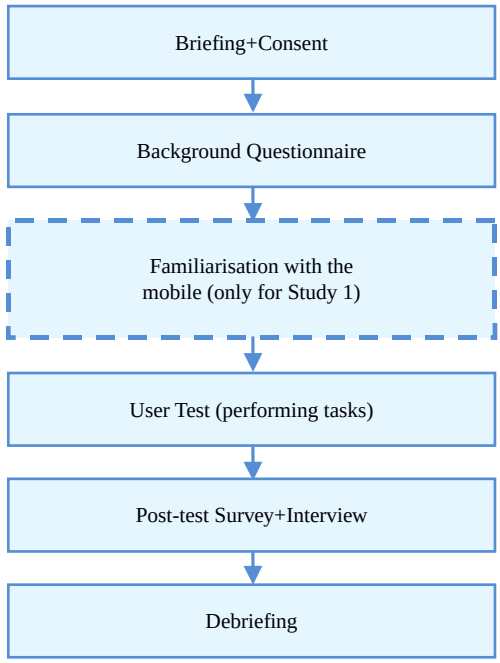


Figure 4.
Protocol for user trials

of these times are shown in Figure 5 (the dotted box shows the mean time just for users who completed Task 3). In all cases the tasks were carried out more quickly on the prioritised version even among those users who had experience with using the Facebook site.

A paired-samples *t*-test was conducted to examine whether there were significant differences of mean completion time between the tasks performed on the prioritised version and the tasks performed on the Facebook mobile version. The *t*-test results

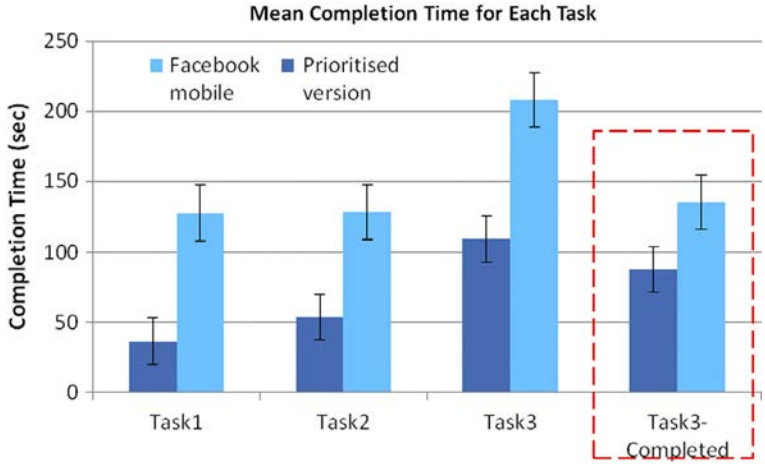


Figure 5.
Mean completion times for tasks performed on the actual mobile site versus the prioritised site

revealed completion time for each task was significantly shorter ($p < 0.05$) on the prioritised version than on the Facebook mobile version

For Task 1 (updating status) and Task 2 (find a friend) users had difficulty on the Facebook mobile version as the status box and search boxes were in a different positions than on the full version. While the prioritised version looks quite different from the full version the consistency in the relative positions of items made it easier for the users to find their way around.

Task 3 (display photo album) was under the < more... > link on the prioritised version. The majority of the users after scrolling down the first page confidently investigated the < more... > link and completed the task. Two users took more time but eventually found what they were looking for. On the actual Facebook page the unfamiliar terminology for links (photos, photo stories) caused confusion to such an extent that three users could not complete the task. Figure 5 shows the mean time for all users (including those who abandoned the task) and also the times just for those users who completed the task (dotted box).

Throughout the trials, participants were observed and were seen to use their experience and knowledge from the Facebook full site to carry out the tasks on both the prioritised and actual mobile versions. Participants seemed to easily find, with less clicking and scrolling, the items they were looking for if the order or locations of items had some similarities to the Facebook full site version. Conversely, most participants were inclined to overlook the items that they were looking for if the items appeared in a different location to that on the Facebook full site version.

It was observed that participants, using their desktop experience, have already determined where things should be on a page. Thus, not initially finding the items where they expected them to be, a few participants were reluctant to scroll further or carefully inspect the page. This reluctance to scroll caused the participants to try and click any link available on the top of the page even though the links were incorrect. This observation is similar to those of Marsden *et al.* (2002) and Shrestha (2007) who found that participants are reluctant and not prepared to scroll.

Participants were asked about being able to give their own priorities to elements on the pages and were receptive of this idea. There was good agreement among the participants about what items they would include the top five items being mentioned by more than a third. This suggests that it would be possible to provide default prioritisations that would be acceptable to the majority of users (although a larger trial would be necessary to determine this more accurately). Users would be able to provide their own modifications if they desired.

10. Performance trial

The performance of the prioritisation engine was tested to ensure that the time required to perform the prioritisation would not devalue the users experience. The tests were undertaken with: databases with different numbers of rows for user, device, item tuples; pages with different numbers of prioritisable items and different numbers of items being chosen to be displayed. Each part of the process was investigated: querying the database, finding the prioritised items, ordering the items and displaying them. The overall times for preparing and displaying a prioritised page was compared with displaying a similar static page.

The greatest contribution to the overall time was querying as the database size grew. Figure 6 shows the time for the prioritisation with different sized databases. The tests were undertaken with a standard relational database without any query optimisation. For much larger databases, different data structures or query optimisation could be investigated if query time became an issue.

Increasing the size of the database from one thousand to one million records increased the time taken by the query by approximately 0.3 milliseconds. If this linear trend continues for larger databases then the additional time for increased rows should be negligible.

The size of the page to be prioritised also has an effect on the performance of the approach. Tests were done with up to 100 items on a page representing a large page. The performance of each stage of the prioritisation process is shown in Figure 7.

Increasing the number of items in a page from ten to 100 items increased the time taken by the UpdateRank phase by about 3 milliseconds; increased the time taken for query by about 2 milliseconds; and increased the time for sort by only about 0.3 millisecond. If the linear trend continues for pages with more items then it suggests that time could be an issue for UpdateRank and Query. This test was undertaken on a

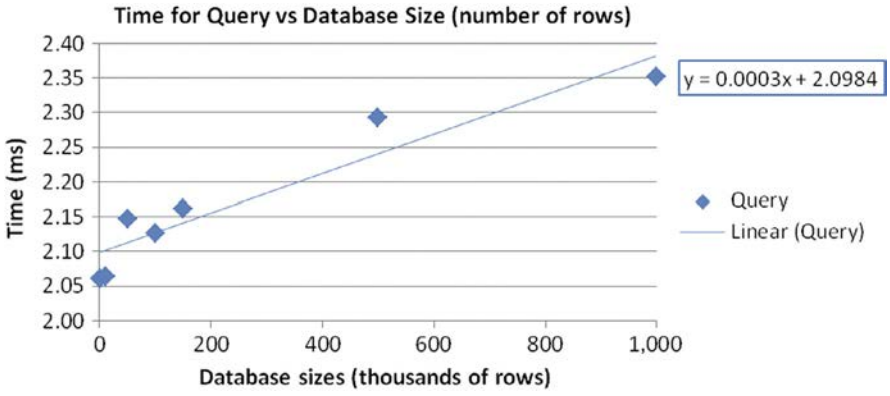


Figure 6.
Time for queries over database size (thousands of rows)

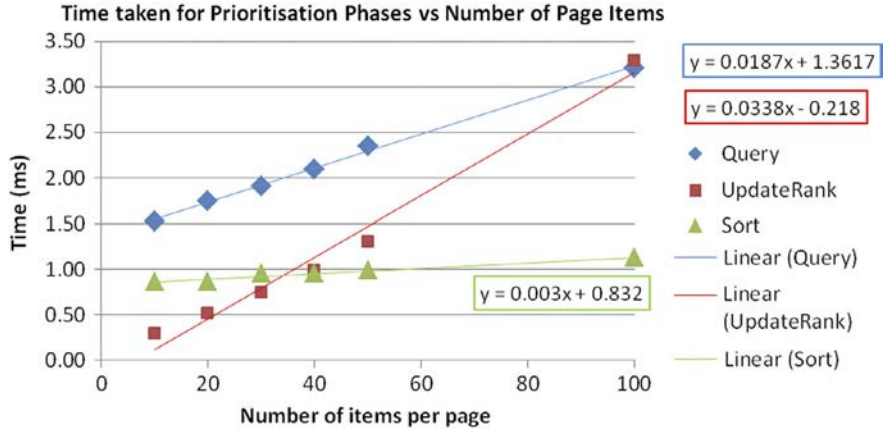


Figure 7.
Time for different prioritisation phases over number of page items

prototype without any code optimization. Although it is unlikely that a page will have more than 100 items, code optimization mechanisms could be investigated.

Results showed that the time taken to prioritise a page was higher than the time taken to load the static page. To illustrate, on a database with 1,000 records, prioritising a ten item page with five items assigned a preference (five items with a preference) added around 19 milliseconds compared with loading the static page. Similarly, on a database with 1,000 K records, prioritising a 50 item page with 50 items of preferences added around 24 milliseconds than only loading the equivalent static page.

The overall total time spent to deliver (prioritise and display) a prioritised page including the network transmission delay and rendering time should be below the general benchmark web site load times of 2-3 s (Gomez, 2010b) and should be within the response time that most users are willing to wait for web sites to load on their mobile devices (Compuware, 2011; Gomez, 2010a).

11. Discussion

Our results show the practicability and feasibility of the prioritisation of adaptive pages to produce similar web pages on different devices. This approach allows only one version of a web site be developed which reduces the costs associated with developing and maintaining multiple versions of sites, and ensures consistency of pages on all devices. It is easy for the developers. Developers can set defaults items to be prioritised and displayed on different devices without changing the original desktop version. The prioritisation also allows users to have only items of interest displayed; and with carefully thought out defaults, only a minor customization is required for the majority of users. Further work would determine if developers are able to determine the appropriate defaults for a web site. Our results with users suggest that there is a consensus among users as to which items are high priorities but that users will have different relative priorities within that grouping. This would suggest that users would need to make only small adjustments to priorities if any.

The prioritisation preserves the pages' parent-child relationships, ensuring the overall layout and structure of the original page is maintained. Our trials' results showed that this similarity and familiarity helped users to have a good browsing experience, in which they were able to relate and recall their experience with the familiar full site version, thus performing tasks easily. In comparison, users had difficulty with navigating the differently structured pages for Facebook desktop and mobile version. Participants performed tasks more quickly on the prioritised version. The prioritisation also imposes only small overhead.

12. Conclusion

Our implementation was based on the assumption that web developers (or other stakeholders involve in decision making in the web site development) would easily detect and determine the areas of content to be prioritised (the *div*s that could be prioritised) and would follow the general recommendation that each *id* within a page should be unique.

The prioritisation engine was designed to operate on the server; this removes any requirement for an intermediate proxy or client-side software. This approach enables the delivery of a consistent experience to all users.

The requirements for our approach were:

- ensure a similar page structure is delivered on all devices;
- ensure consistency of terms used and the location of items on all devices;
- minimise items displayed and reduce navigation within a page; and
- support users visiting the site on multiple different devices.

The prioritisation engine approach meets these requirements by augmenting the *div* element so the same source page generates pages for all devices. The use of a single source ensures the consistency of structure and terms as content is preserved. The items displayed can be customised on a per-device, per-user basis which supports users visiting on an arbitrary number of devices and allows an experience that is appropriate for the device being used.

We have undertaken both qualitative and quantitative evaluations of the approach which have demonstrated that it requires very little overhead and provides a browsing experience at least as good as, and in many cases better than, providing independent sites.

Future work would allow the development of an appropriate user interface for selecting priorities and expansion of the concept to manipulate media appropriately. Our work has focused on sites with a structure that is generated server-side and has not focused on sites that make extensive use of client-side scripts to dynamically create or modify pages. Additional work would be required to accommodate this type of site. We envisage a local prioritisation engine that works with server-side preferences and priorities to just download the appropriate page items.

Notes

1. http://nz.yahoo.com/
2. www.google.co.nz
3. www.stuff.co.nz/

References

Anderson, C., Domingos, P. and Weld, D. (2001), "Personalizing web sites for mobile users", *Proceedings of the 10th World Wide Web Conference* (*WWW10*), *Hong Kong*.

Bila, N., Ronda, T., Mohomed, I., Truong, K.N. and Lara, E.D. (2007), "PageTailor: reusable end-user customization for the mobile web", *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services, San Juan, Puerto Rico*.

Byron, L. (2011), *One Mobile Site to Serve Thousands of Phones*, available at: www.facebook.com/notes/facebook-engineering/one-mobile-site-to-serve-thousands-of-phones/10150122073713920 (accessed 2 April 2011).

Caetano, M.F., Fialho, A.L.F., Bordim, J.L., Castanho, C.D., Jacobi, R.P. and Nakano, K. (2007), "Proteus: an architecture for adapting web page on small-screen devices", *Network and Parallel Computing*, Lecture Notes in Computer Science, Springer, Berlin, pp. 161-170.

Compuware (2011), "What users want from mobile", White Paper, available at: www.gomez.com/wp-content/downloads/19986_WhatMobileUsersWant_Wp.pdf (accessed 22 July 2011).

Gomez (2010a), "Great expectations – what users want from the mobile web experience", White Paper, available at: www.gomez.com/resources/whitepapers/mobile-great-expectations4/ (accessed 19 November 2010).

Gomez (2010b), *"When Seconds Count" National Consumer Survey on Website and Mobile Performance Expectations*, available at: www.gomez.com/wp-content/downloads/GomezWebSpeedSurvey.pdf (accessed 1 October 2010).

ITU (2011), *The World in 2011 ICT Facts and Figures*, available at: www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf (accessed 4 December 2011).

Kao, Y.-W., Kao, T.-H., Tsai, C.-Y. and Yuan, S.-M. (2009), "A personal web page tailoring toolkit for mobile devices", *Computer Standards & Interfaces*, Vol. 31 No. 2, pp. 437-453.

Marsden, G., Cherry, R. and Haefele, A. (2002), "Small screen access to digital libraries", paper presented at the CHI'02 Extended Abstracts on Human Factors in Computing Systems.

Nichols, J. and Lau, T. (2008), "Mobilization by demonstration: using traces to re-author existing web sites", *Proceedings of the 13th International Conference on Intelligent User Interfaces Gran Canaria, Spain, 13-16 January*.

Nielsen, J. (1993), *Usability Engineering*, Academic Press, Boston, MA.

Nielsen, J. (2009), *Customization of UIs and Products*, available at: www.useit.com/alertbox/customization.html (accessed 18 August 2009).

Ohri, K. (2011), "The adoption of mobile devices for surfing internet will happen faster in India: Mahesh Narayanan", available at: www.afaqs.com/media/story.html?sid=30940 (accessed 27 June 2011).

Shaari, N. (2013), "Customisation of web content for desktop and mobile devices", PhD thesis, Lincoln University, Lincoln.

Sharp, H., Rogers, Y. and Preece, J. (2006), *Interaction Design: Beyond Human-Computer Interaction*, 2nd ed., Wiley, Hoboken, NJ.

Shrestha, S. (2007), "Mobile web browsing: usability study", *Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology*, ACM, Singapore, pp. 187-194.

W3C (2008), *Mobile Web Best Practices 1.0*, available at: www.w3.org/TR/mobile-bp/ (accessed 2008).

Xiao, X., Luo, Q., Hong, D., Fu, H., Xie, X. and Ma, W.-Y. (2009), "Browsing on small displays by transforming web pages into hierarchically structured sub-pages", *ACM Trans. Web*, Vol. 3 No. 1, pp. 1-36.

Xiao, Y., Tao, Y., Li, Q. and Li, W. (2008), "A customizable mobile device oriented web data extraction base on user behavior feedback", paper presented at the 2008 International Conference on Computer Science and Software Engineering, Wuhan, China, 12-14 December.

Corresponding author
Stuart Charters can be contacted at: stuart.charters@lincoln.ac.nz