Computer Standards & Interfaces 31 (2009) 437-453

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

A personal Web page tailoring toolkit for mobile devices

Yung-Wei Kao [a,*], Tzu-Han Kao [a], Chi-Yang Tsai [a], Shyan-Ming Yuan [a,b]

[a] Department of Computer Science and Engineering, National Chiao Tung University, 1001 Ta Hsueh Rd., Hsinchu 300, Taiwan
[b] Department of Computer Science, Asia University, Lioufeng Rd., Wufeng, Taichung County, Taiwan

ARTICLE INFO

Article history:
Received 15 August 2007
Received in revised form 2 March 2008
Accepted 4 May 2008
Available online 18 May 2008

Keywords:
Mobile device
Web page
Personalization

ABSTRACT

It is common to browse web pages via mobile devices. However, most of the web pages were designed for desktop computers equipped with big screens. When browsing on mobile devices, a user has to scroll up and down to find the information they want because of the limited screen size. Some commercial products reformat web pages. However, the result pages still contain irrelevant information. We propose a system to personalize users' mobile web pages. A user can determine which blocks in a web page should be retained. The sequence of these blocks can also be altered according to individual preferences.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

1.1. Preface

Nowadays, we can build a mobile Web application easily. Take the ASP.NET mobile controls [1] (formerly known as MMIT, short for the Microsoft Mobile Internet Toolkit) for example, it reduces the work required for developers to develop applications that target different types of mobile devices, like mobile phones and PDAs. At runtime, MMIT will automatically detect the target device and return the proper presentation format. Thus, the developers can focus on the application logic without worrying about the presentation issues.

Similar academic researches on this topic include [2] and [3]. Both of them provide plug-ins for famous IDEs, such as JBuilder and Visual Studio.NET respectively, to assist developers to author a single generic application. Accompanied with the PUML transformation technology [4], this generic application can be further transformed into specific target formats for different mobile devices.

1.2. Motivation

Widespread of mobile devices makes it common to browse Web pages via them. However, most Web pages are mainly designed for desktop computers that are equipped with big screens. When browsing on mobile devices, a user might have to scroll up and down, left and right all the time to find the information they want. Because of the limited screen size, this kind of operation is really not user-friendly at all.

Fortunately, some famous websites have another simplified version of Web content specially provided for mobile devices, such as Google Mobile [5] and Yahoo Mobile [6]. On the other hand, it is a heavy burden on Web developers to craft and maintain multiple versions of the same website. Even with the help of the fascinating toolkits.

If we resize the original Web page to fit the width of mobile device, the vertical scroll bar will be too long to view, and the information is crowded. On the other hand, if we provide another version of the original Web page, there may be some important information lost in the mobile version, and the transformation of each page costs a lot for Web page developers. Hence, in this research, we propose a system that is designed to help users to personalize their mobile Web pages for handheld device browsing.

1.3. Research objectives

In this sub-section, the four major research objectives are listed and introduced briefly.

1.3.1. Easy-to-use

It does not make sense to launch another program other than the browser to personalize a Web page. When a user surfs on the Internet and finds a Web page that interests him/her, the configuration tool of this system should be able to pop up in the browser window somehow right away. Moreover, all the codes needed to accomplish this job (i.e. personalize Web pages) should be downloaded on the fly when

* Corresponding author.
E-mail addresses: ywkao@cs.nctu.edu.tw (Y.-W. Kao), gis89539@cis.nctu.edu.tw (T.-H. Kao), gis93532@cis.nctu.edu.tw (C.-Y. Tsai), smyuan@cis.nctu.edu.tw (S.-M. Yuan).

0920-5489/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2008.05.016

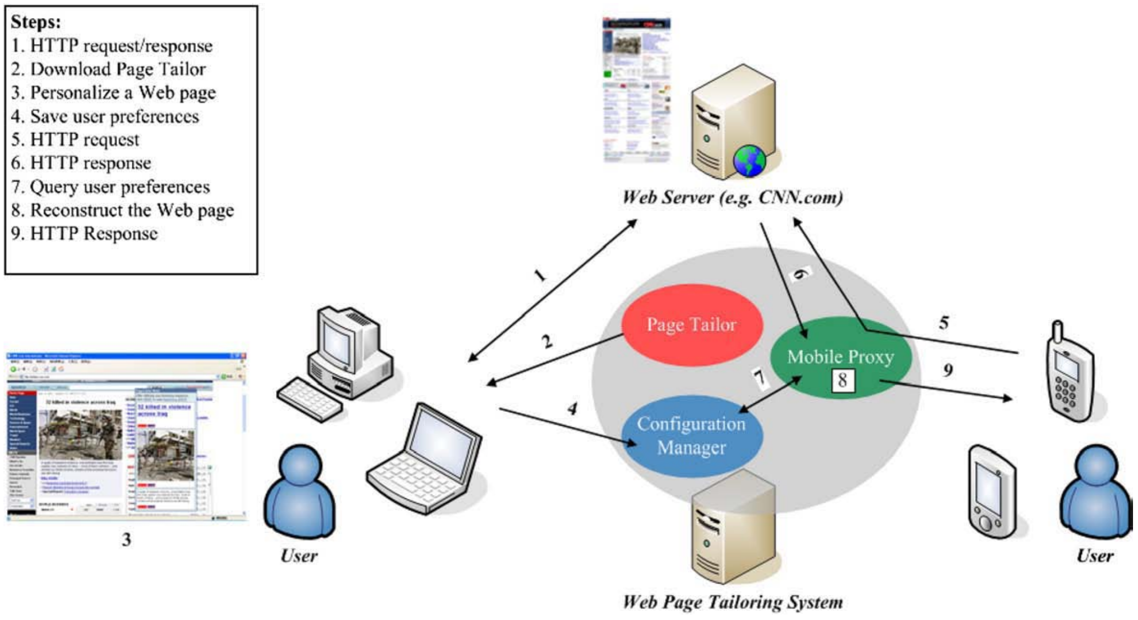


Fig. 1. Overview.

accessed, thus allowing a user to work on different computers at different places.

1.3.2. Personalizing Web pages visually

Web pages are usually composed of header, footer, sidebar, and content areas [7]. Parts of them are used to maintain a consistent style for the website, and other parts of them are used for navigation. Some renowned websites may even contain a lot of advertisements on it. In many Web pages, only a few of information is really needed to be shown on the mobile phone screen. This research also aims at allowing a user to determine which parts of a Web page should be retained while browsing this page with their mobile device.

A friendly user interface should thus be available for a user to perform this task. For example, with appropriate visual aids (such as highlight), a user can choose blocks in a Web page one by one with different granularity. Through the operation of drag-and-drop, a user can determine the relative position of the chosen blocks according to his/her personal preferences. In short, a user can reconstruct a mobile Web page simply with visual manipulations, and does not have to write any line of code.

According to the browser market share survey [8], Microsoft Internet Explorer is still by far the most dominant browser on the Web, with 83.88% usage market share, and Firefox has increased its share to 10.68%, with the other alternatives, such as Safari, Netscape, Opera, and Mozilla, occupying the remaining share.

The downloaded mobile code should work with at least the top two popular browsers, i.e. Microsoft Internet Explorer and Firefox Web browser.

1.3.3. Reducing wireless bandwidth consumption

More than screen size constraints, the limited memory and wireless network bandwidth also make it unsuitable for delivering the entire Web page untailored to mobile devices. Before returning a Web page to mobile devices, some adaptation must be taken to pre-process a Web page according to a user's preferences. So that the volume of data transmission to a mobile phone could be reduced, and thus reduce the consumption of wireless bandwidth as well.

1.3.4. Automatic mobile Web page content extraction

The content adapting algorithm we proposed can automatically adapt Web pages to mobile devices. Users always only care about a part of the web content. Content adapting applications should provide a function to extract these parts from a Web page.

1.4. Research contribution

This paper discusses the problems encountered and our corresponding solutions. The major contributions of this research are listed below.

1. A cross-browser configuration tool is designed.
2. The web-based nature of our configuration tool allows a user to configure the settings from different computers, and requires no pre-installation of any software.
3. Blocks in a Web page can be chosen correctly under the premise that the layout of a Web page does not change frequently.
4. A web-based management interface is provided.
5. An automatic algorithm for mobile Web page generation is proposed.

1.5. Outline of this paper

This paper is divided into seven sections. The following is a brief description of the content of each section. In Section 2, an overview of the proposed system and its three major components are given. In Section 3, the System design details, problems encountered, and our

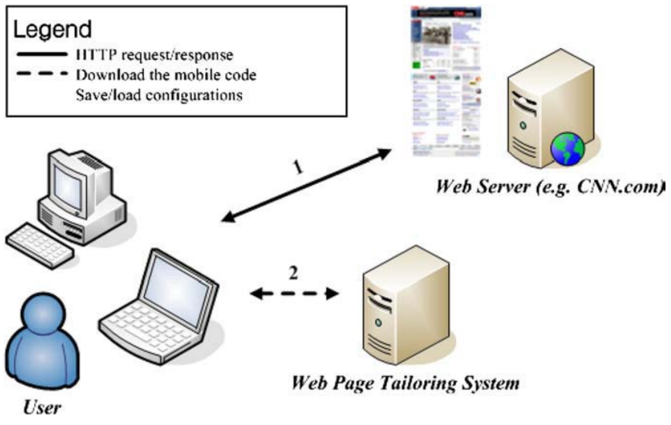


Fig. 2. Personalize Web pages using PC or laptop.

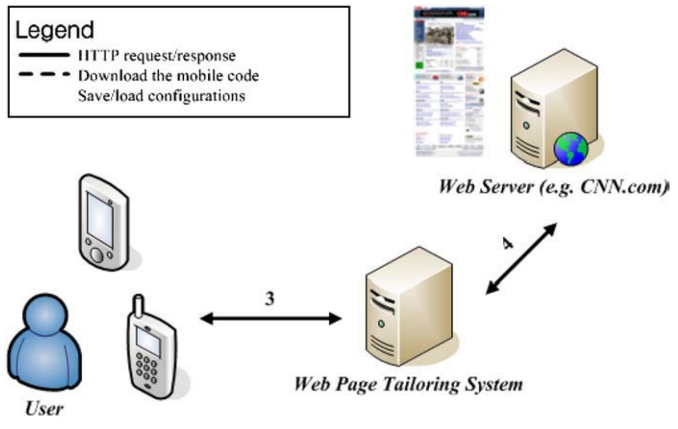


Fig. 3. Browse Web pages via mobile devices.

corresponding solutions are illustrated. Furthermore, we discuss our mobile Web page generation algorithm in Section 4. In Section 5, some tests are conducted to evaluate our system. A practical example showing how to make use of the proposed tool to eliminate unnecessary scrolling is also presented. Several related works are discussed in Section 6. Finally, in Section 7, we make the conclusion and discuss about the future work.

2. System design

2.1. Overview

Basically, the personalizing process in our Web page tailoring system comprises two steps (Fig. 1). First, user must specify his/her preferences of a Web page using a PC or laptop. Second, he/she has to configure the browser on his/her mobile device to go through a specially made proxy, which is responsible for adjusting the content of Web pages according to the preferences set in the first step. Two pictures are given below to illustrate separately the relationship between a user, our Web page tailoring system, and a remote Web server (such as CNN.com) in each step.

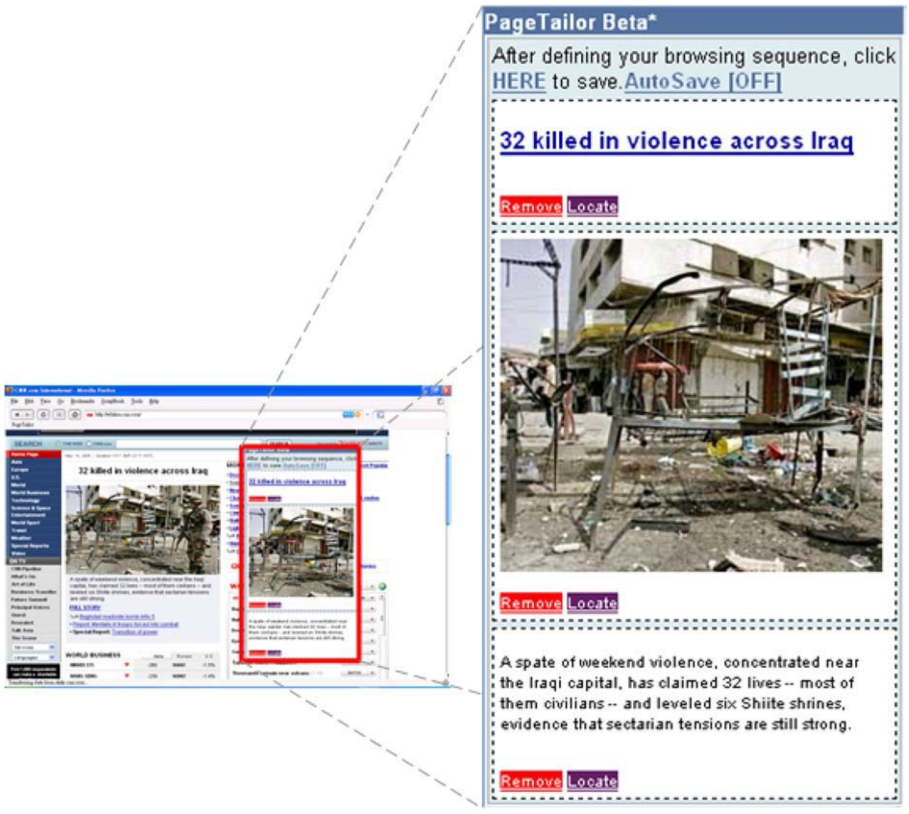
Fig. 2 describes the interaction in the first step. When a user enters a URL in his/her Web browser, a HTTP request is sent to (Line 1) the corresponding Web server specified in the URL. After processing the request by the server, a HTTP response is sent back (Line 1). If the user wants to personalize that page, a program hosted on a tiny Web server included in our system would be downloaded (Line 2) and executed in his/her browser. With the help of that program, the user can specify his/her preferences simply by visual manipulations. After finishing the job, preferences about this page will be sent back and stored in a database for later use (Line 2).

Fig. 3 pictures the interaction in the second step. Since the user would configure the browser on his/her mobile device to use a proxy included in our system, we would snoop each HTTP request and modify its corresponding response (Lines 3 and 4) in between. For example, if the user visits a Web page that has been personalized before, some actions would be taken to tailor the Web page to meet the user preferences.

In order to achieve the above tasks, three components are designed in our system: Page Tailor, Configuration Manager, and Mobile Proxy. The purpose and functions of each component will be introduced separately. System design details are presented in Section 3.

2.2. Page Tailor

Page Tailor in the form of mobile code can be downloaded and executed in a user's browser when he/she is about to personalize a Web page. It provides some visual manipulations for users to help them specify their preferences about a Web page. The preferences here include: blocks of a Web page that should be retained and their final


PageTailor Beta*

After defining your browsing sequence, click HERE to save. AutoSave [OFF]

32 killed in violence across Iraq

A spate of weekend violence, concentrated near the Iraqi capital, has claimed 32 lives -- most of them civilians -- and leveled six Shiite shrines, evidence that sectarian tensions are still strong.

Fig. 4. Page Tailor in Firefox Web browser.

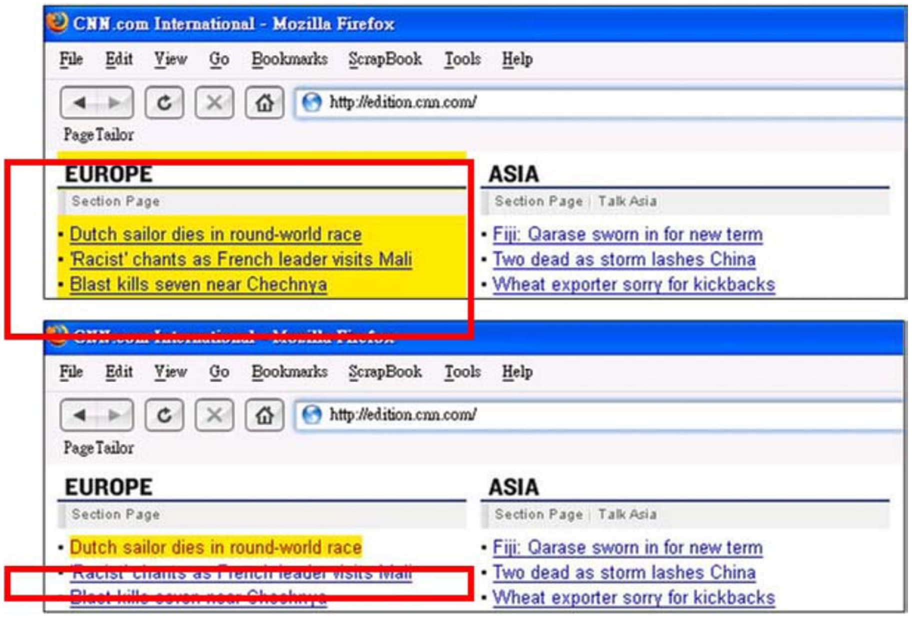

CNN.com International - Mozilla Firefox
File Edit View Go Bookmarks ScrapBook Tools Help
http://edition.cnn.com/
PageTailor
EUROPE
Section Page
Dutch sailor dies in round-world race
'Racist' chants as French leader visits Mali
Blast kills seven near Chechnya
ASIA
Section Page | Talk Asia
Fiji: Qarase sworn in for new term
Two dead as storm lashes China
Wheat exporter sorry for kickbacks

Fig. 5. Select blocks at different granularity.

arrangement. All the preferences about this page would be saved in a remote database that is managed by Configuration Manager. Fig. 4 is a snapshot when executing Page Tailor in a Firefox Web browser.

2.2.1. Execution and initialization

When browsing Web pages, a user can click on the installed bookmarklet to download and execute Page Tailor. From the other perspective of users, it seems that the Web page itself provides the personalizing functions.

After the Page Tailor window is launched in the user's browser, some actions are performed in the background automatically. First, Page Tailor will connect with Configuration Manager to retrieve the user preferences about this current visited page. If the user has personalized this page before, Page Tailor would retrieve the old preferences, and then use the data retrieved to reconstruct the past, such as blocks that had been selected and their order.

On the contrary, if there are no preferences about this page, nothing will happen, of course. The purpose of this action is to help users accelerate the setting time; particularly when he/she only wants to perform a slight modification.

2.2.2. Visual manipulations

In order to help a user specify his/her preferences about a Web page, Page Tailor provides some visual manipulations. Fig. 5 demonstrates

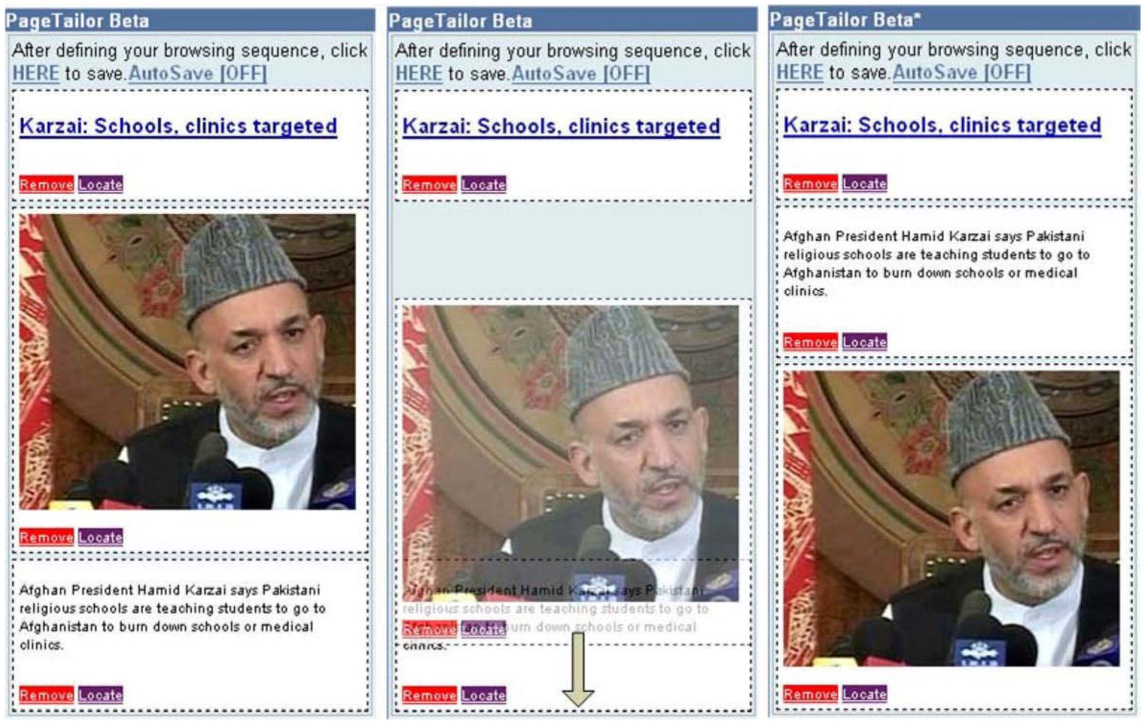
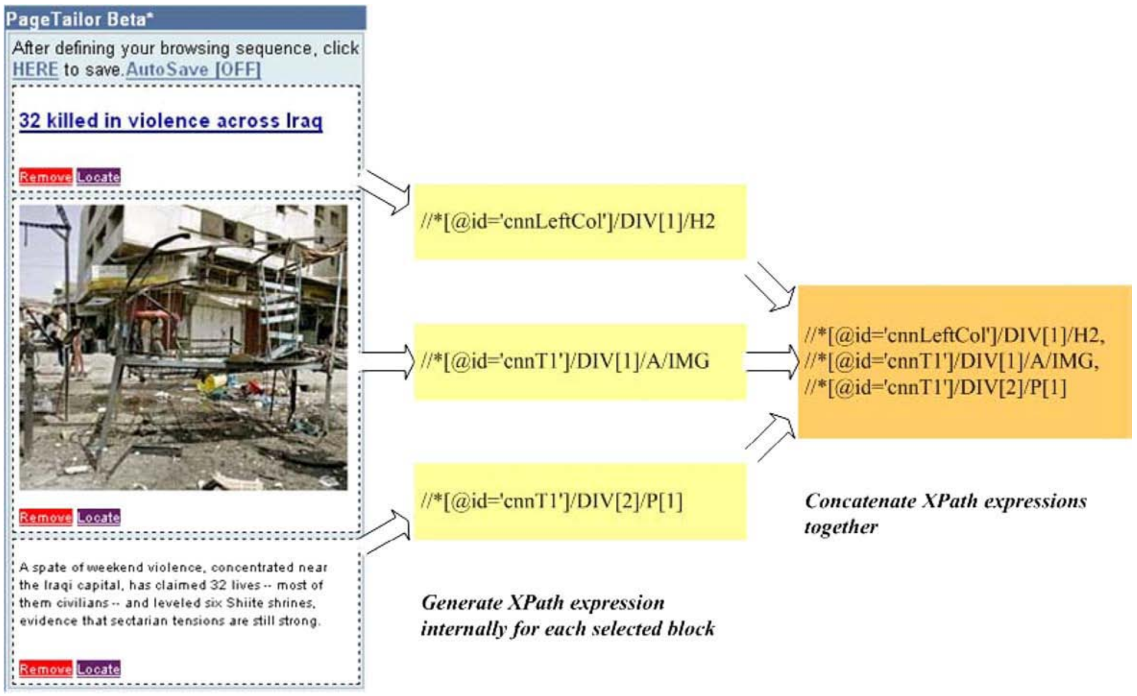

PageTailor Beta
After defining your browsing sequence, click HERE to save. AutoSave [OFF]
Karzai: Schools, clinics targeted
Remove Locate
Afghan President Hamid Karzai says Pakistani religious schools are teaching students to go to Afghanistan to burn down schools or medical clinics.

Fig. 6. Rearrange the selected blocks.


PageTailor Beta*
After defining your browsing sequence, click HERE to save. AutoSave [OFF]
32 killed in violence across Iraq
Remove
Locate
A spate of weekend violence, concentrated near the Iraqi capital, has claimed 32 lives -- most of them civilians -- and leveled six Shiite shrines, evidence that sectarian tensions are still strong.
Generate XPath expression internally for each selected block
Concatenate XPath expressions together

```
//*[@id='cnnLeftCol']/DIV[1]/H2
//*[@id='cnnT1']/DIV[1]/A/IMG
//*[@id='cnnT1']/DIV[2]/P[1]

//*[@id='cnnLeftCol']/DIV[1]/H2,
//*[@id='cnnT1']/DIV[1]/A/IMG,
//*[@id='cnnT1']/DIV[2]/P[1]
```

Fig. 7. Internal expression of user preferences about a Web page.

the feature that a user can select a block in a Web page at different granularity. For example, in the top half of this picture, a block containing more information than that in the bottom is selected. A selected block is highlighted in yellow.

Fig. 6 illustrates another feature — drag and drop. In this picture, three views are shown from left to right. In the beginning, three blocks have already been selected (left). Next, we switch the last two blocks (middle), and then the final result comes out (right). The sequence of blocks in the Page Tailor window would be the same as that in the browsers on mobile devices.

2.2.3. User preferences

XPath [9] is a language that describes how to locate specific elements in a document. This is the standard we adopted to store the user preference. In our Web Page Tailoring system, the user preferences stored are composed of XPath expressions. In other words, when a user adds a block to the Page Tailor window, Page Tailor would internally generate an XPath expression for that block. By using XPath expressions, we can uniquely identify this block in the future provided that if the layout of this page does not change too frequently.

As for the sequence of selected blocks, the XPath expressions of selected blocks are concatenated together according to their order in the Page Tailor window (separated by commas) to form the user preferences about this page. Fig. 7 is a practical example.

2.3. Configuration Manager

The preferences specified by a user will be stored in a database. Configuration Manager serves as a gatekeeper to control the access to the backend database. It provides a Web-based interface) for a user to manage his/her preferences base on the HTTP 1.0 Web protocol. Two Web services are also exported to allow other components in this system to access the preferences programmatically. One is used for querying the database and the other for updating.

The reason why we adopted the Web service approach ultimately is because of its language- and platform-specific nature. Hence, other components utilizing the services in our system could be implemented in different programming languages that are more appropriate for specific tasks.

2.4. Mobile Proxy

Mobile Proxy is a specially made proxy that is responsible for the final step in completing the personalizing process. It would monitor every HTTP request and makes Web service calls (query service) with the request URL as the parameter to Configuration Manager. The returned user preferences, if any, could then be employed to filter out the unwanted Web page content and rearrange the remaining blocks.

Fig. 8 summarizes the communication between these three major components. Page Tailor is first downloaded and executed in a user's browser. Then the Web services exported by Configuration Manager are used by it and Mobile Proxy to access the backend database where user preferences are stored.

3. Detailed design, standards, and interfaces

In this section, we will describe the system design, standards, and interfaces details of the three major components in our system and the problems encountered. Also, our solutions to these problems are depicted. The system components and their interfaces are shown in Fig. 9.

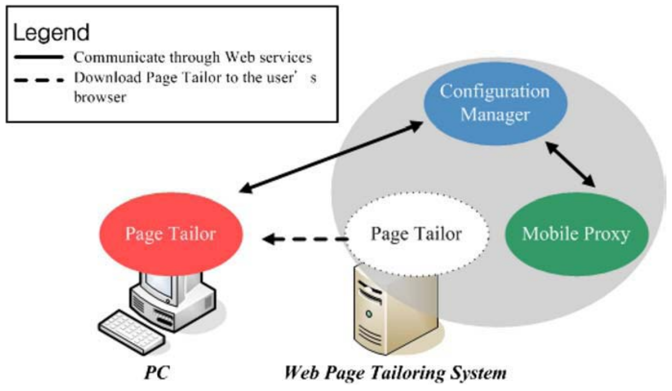

Legend
Communicate through Web services
Download Page Tailor to the user' s browser
Configuration Manager
Page Tailor
Page Tailor
Mobile Proxy
PC
Web Page Tailoring System

Fig. 8. The communication between these three components.

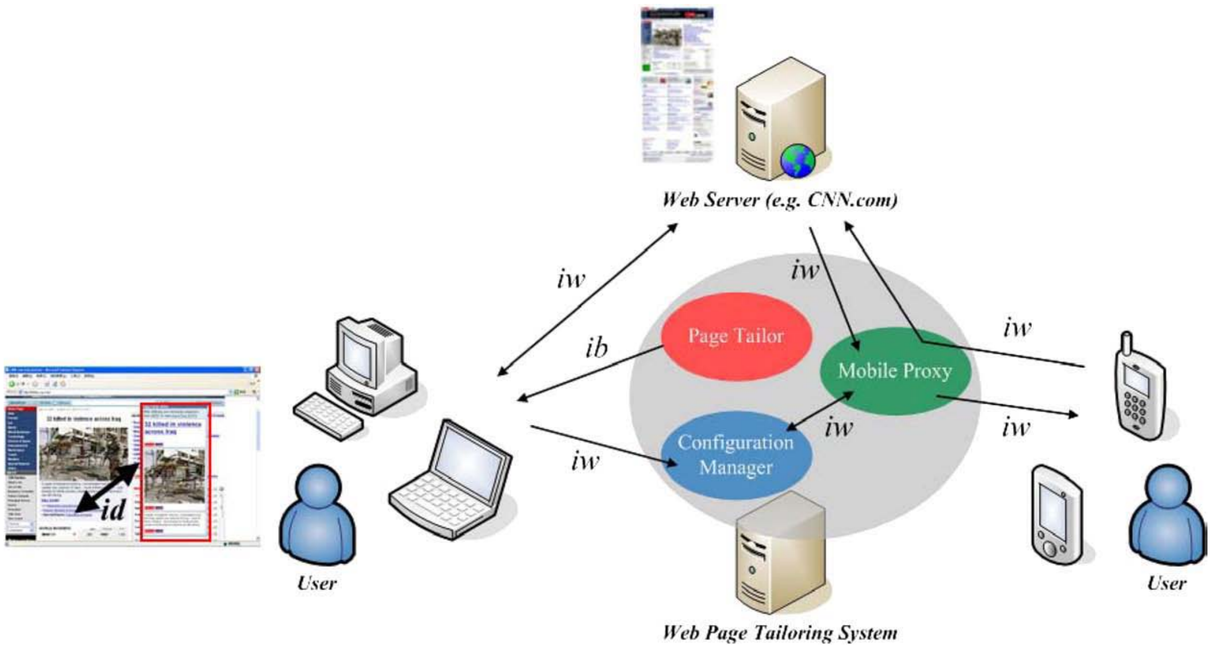
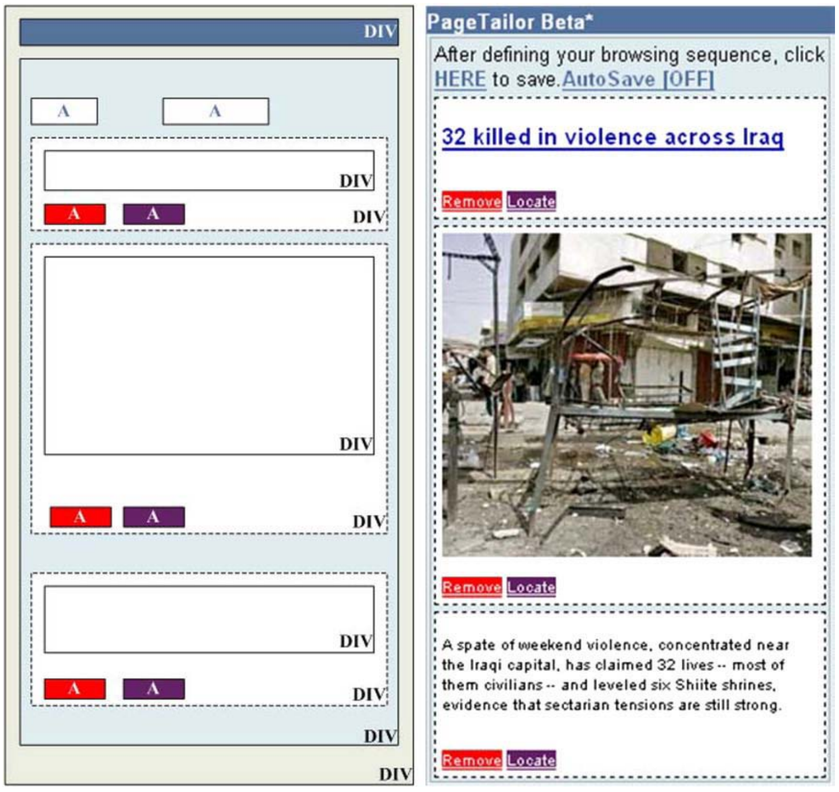


Fig. 9. System components and their interfaces.

```
javascript:
void(s=document.createElement('script'));
void(s.type='text/javascript');
void(s.id='tk_bookmarklet');
void(s.src='http://localhost:3000/javascripts/loader.js');
void(b=document.getElementsByTagName('body')[0]);
void(b.appendChild(s));
```

Fig. 10. This is the bookmarklet used to launch Page Tailor.

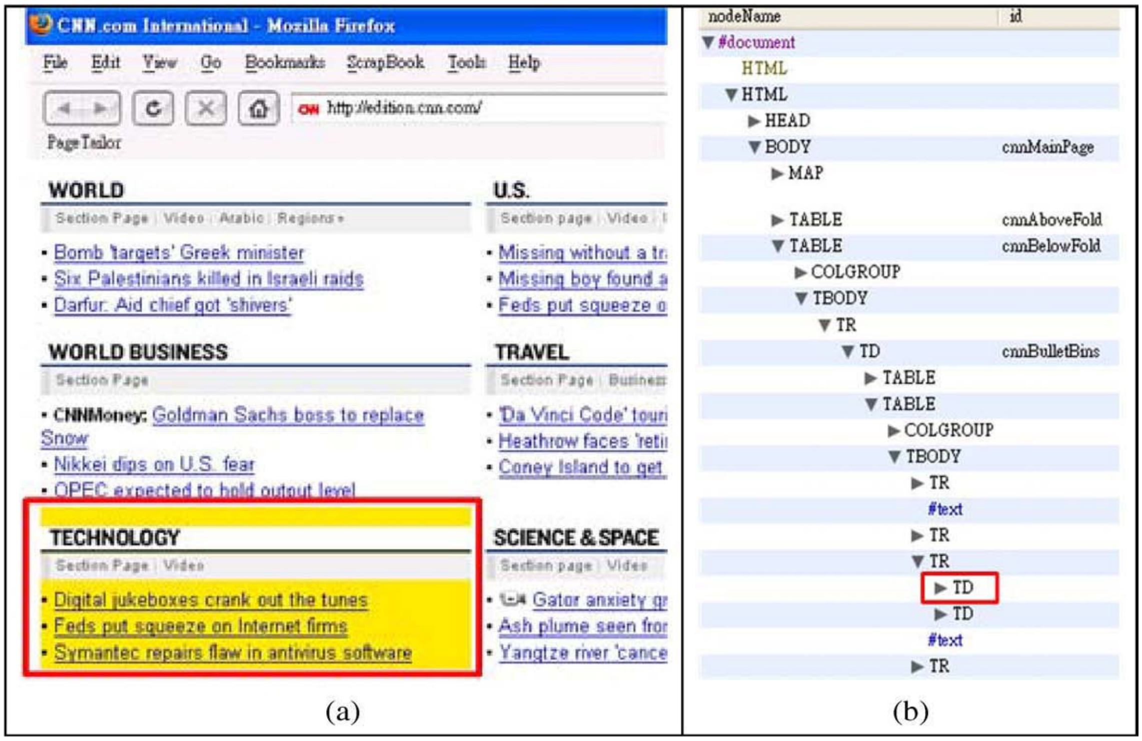


After defining your browsing sequence, click HERE to save. AutoSave [OFF]

32 killed in violence across Iraq

A spate of weekend violence, concentrated near the Iraqi capital, has claimed 32 lives -- most of them civilians -- and leveled six Shiite shrines, evidence that sectarian tensions are still strong.

Fig. 11. The containment hierarchy of elements.



CNN.com International - Mozilla Firefox

File Edit View Go Bookmarks ScrapBook Tools Help

http://edition.cnn.com/

Page Tailor

WORLD
Section Page | Video | Arabic | Regions
- Bomb targets' Greek minister
- Six Palestinians killed in Israeli raids
- Darfur: Aid chief got 'shivers'

U.S.
Section page | Video
- Missing without a tr
- Missing boy found a
- Feds put squeeze o

WORLD BUSINESS
Section Page
- CNNMoney: Goldman Sachs boss to replace Snow
- Nikkei dips on U.S. fear
- OPEC expected to hold output level

TRAVEL
Section Page | Business
- 'Da Vinci Code' touri
- Heathrow faces 'reti
- Coney Island to get

TECHNOLOGY
Section Page | Video
- Digital jukeboxes crank out the tunes
- Feds put squeeze on Internet firms
- Symantec repairs flaw in antivirus software

SCIENCE & SPACE
Section page | Video
- Gator anxiety gr
- Ash plume seen fro
- Yangtze river 'cance

(a)

```
nodeName                      id
▼#document
    HTML
  ▼HTML
     ▶HEAD
     ▼BODY                    cnnMainPage
        ▶MAP
        ▶TABLE                cnnAboveFold
        ▼TABLE                cnnBelowFold
           ▶COLGROUP
           ▼TBODY
              ▼TR
                 ▼TD          cnnBulletBins
                    ▶TABLE
                    ▼TABLE
                       ▶COLGROUP
                       ▼TBODY
                          ▶TR
                           #text
                          ▶TR
                          ▼TR
                             ▶TD
                             ▶TD
                           #text
                          ▶TR
```

(b)

Fig. 12. The selected block (a) and its corresponding subtree (b).

```
query(current url, callback);

function query(current_url, callback_function_name){
  var req  = document.createElement("script");
  req.type = "text/javascript";
  req.id   = random number
  req.src  = "http://.../.../query_service?url=" + current_url
                      +"&callback=" + callback_function_name
                      +"&id=" +req.id;

  Add the element to the web page
}

function callback(service_response, element_id){
  Process the response(i.e. service_response)
 Find and remove the element with the specified value (i.e. element_id) of the ID attribute
}
```

Fig. 13. How to load the user preferences (pseudo-code).

In Fig. 9, we can see that, almost all of the interfaces are iw, i.e. HTTP1.0 Web interface. Because our system is designed for Web application, the *iw* interface enables us to develop different system components by different programming languages or platforms. Hence, it provides cross-platform communication for system development.

3.1. Page Tailor

Page Tailor is designed to be implemented in JavaScript because of the following considerations.

1. JavaScript was designed to add interactivity to HTML pages. This point conforms to the first objective of this paper, i.e. easy-to-use, since we can embed the configuration tool in a Web page, so that users can configure their settings directly in the browser.

Table 1
Examples of origin comparisons

URL	Result	Description
http://store.company.com/dir2/other.html	Success	
http://store.company.com/dir/inner/another.html	Success	
https://store.company.com/secure.html	Failure	Different protocols
http://store.company.com:81/dir/erc.html	Failure	Different ports
http://news.company.com/dir/other.html	Failure	Different hosts

```
callback(user preferences, id specified in the query string);
```

Fig. 14. The response of query service (pseudo-code).

2. By using JavaScript, we can manipulate a Web page through the DOM interface. DOM provides the interface between the JavaScript and original Web pages. In other words, we can change the appearance of a Web page to reflect the user's choice. This one conforms to the second objective, i.e. personalizing Web page visually.
3. JavaScript is one of the most popular scripting languages on the Internet, and works in all major browsers, such as Internet Explorer and Firefox. This one agrees with the third objective, i.e. support for mainstream browsers.

3.1.1. Page Tailor bookmarklet

The interface between user and Page Tailor is bookmarklet. That is, the *ib* interface between user and Page Tailor in Fig. 9. The source code of this bookmarklet is shown in Fig. 10. When a user clicks that, an element with tag name “script” is created (Line 2). After being created, the new element contains no properties. The “src” property (Line 5) is employed to indicate the location of a remote JavaScript file. A created element exists only in the browser's memory, and not as part of this Web page. In order to let the “script” element take effect, we have to add it to the Web page. For that reason, we find the first occurrence of “body” element (Line 6) in the Web page, and then append the “script” element as a child of it (Line 7).

By this method, we can dynamically load an external JavaScript file on demand to do something with a Web page. It can also work for other types of files such as CSS.

3.1.2. Loading external JavaScript Libraries

JavaScript has been used in many applications. Some open source projects have already incorporated the commonly used functions such as drag-and-drop and visual effects. In Page Tailor, we do not reinvent the wheel but take advantage of the Script.aculo.us JavaScript framework [10].

Although we can dynamically load these JavaScript files, the exact time spent on loading, however, depends on the browsers. In other words, even if we add the “script” elements to a Web page in some order, which is not equivalent to load the first JavaScript file and then the second one. The results are still unexpected.

To address the above problem, we must postpone the loading of scriptaculous.js until after prototype.js has been loaded. By selecting a unique identifier from the file (prototype.js or scriptaculous.js) and checking periodically whether the identifier has been defined or not, we can indirectly deduce the current state of that file. In other words, an identifier has been defined, which means that the file containing it must have been loaded, too. For example, we choose the identifier named Prototype in prototype.js and Draggable in scriptaculous.js.

3.1.3. Containment hierarchy of Page Tailor

After pagetailor.js is loaded, it would create the containment hierarchy of elements that together compose the Page Tailor window, and the entire hierarchy would be appended to the Web page. Fig. 11 shows the correspondence between each element and their appearance displayed on the browser. The tag name of each element contained in this hierarchy is labeled on the left-hand side of the figure.

3.1.4. Generating XPath expressions

The DOM tree and XPath standards are used to store the user preferences. Page Tailor extracts the XPathes of blocks which users interest in on Web pages, and store these XPathes to the Configuration Manager. The degree of support for XPath over HTML varies in different types and brands of browsers, so we must deal with the conversion between a selected block and its corresponding XPath expression by ourselves in Page Tailor. The XPath expressions contained in the user preferences are restricted to internal use only. Then, we will make a statement on how to generate the XPath expressions of a selected block.

As seen in Fig. 12, every selected block corresponds to a subtree of the document. The way to find its XPath expression can be divided into two steps. First, start from the selected block (or node) and walk recursively up the tree to find all of its ancestors. Then, for each node in the path count the number of its siblings that have the same tag name as that of the node. With these two data, we can finally generate the XPath expressions of the selected block.

```
update(current url, user preferences, callback);

function update(current_url, user_preferences, callback_function_name){
  var req  = document.createElement("script");
  req.type = "text/javascript";
  req.id   = random number
  req.src  = "http://.../.../update_service?url=" + current_url
                      +"&prefer=" + user_preferences
                      +"&callback=" + callback_function_name
                      +"&id="+req.id;

  Add the element to the web page
}

function callback(service_response, element_id){
 Find and remove the element with the specified value (i.e. element_id) of the ID attribute
}
```

Fig. 15. How to update the user preferences (pseudo-code).

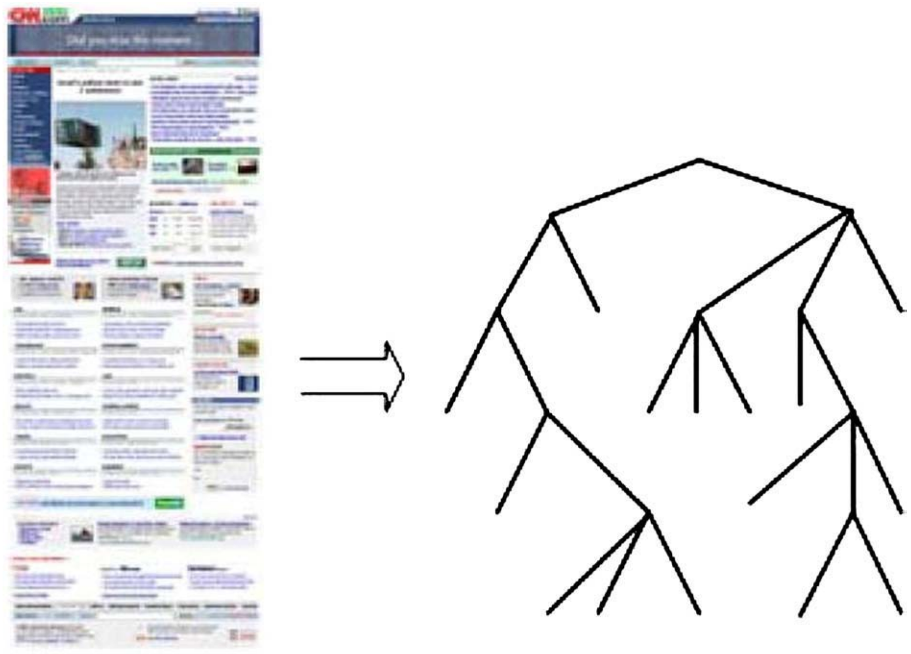


Fig. 16. Produce a corresponding DOM tree.

3.1.5. Same origin policy

It is the XMLHttpRequest object that can be used by JavaScript to transfer data to and from a Web server using HTTP. It is supported by almost all popular browsers. For security reason, however, browsers often impose some restrictions (also referred to as the "same origin policy") on this object.

The "same origin policy" dates from Netscape Navigator 2.0. This policy prevents a document from one origin from getting or setting properties of another document from a different origin. In general, two documents are considered to have the same origin if they are identical in three aspects: protocol, host, and optionally port. Table 1 lists some examples of origin comparisons with the URL "http://store.company.com/dir2/other.html".

From the comparisons in the above table, it is not difficult to imagine that in Page Tailor, it is impossible to retrieve data from Configuration Manager using the XMLHttpRequest object, because a Web page surely comes from the Web server that is situated in a different origin from where Configuration Manager might live. Hence, we do not use the XMLHttpRequest of AJAX standard in Page Tailor.

3.1.6. Accessing user preferences

Using the method using in 3.1.1, we can overcome the restriction imposed by the "same origin policy". In the Page Tailor bookmarklet, we dynamically create a "script" element, assign some attributes to it, and add this element to the Web page. After the remote file has been loaded, the JavaScript code written in that file would be executed accordingly. Hence, the limitation of "same origin policy" of communication between the Page Tailor and Configuration Manager can be overcome by the dynamical JavaScript loading mechanism in HTTP 1.0 standard.

However, the value assigned to its "src" attribute does not point to a remote file. It points to the query service provided by Configuration Manager. Therefore, adding this element to a Web page is equivalent to using the query service. Moreover, since the type attribute of the "script" element has a value of "text/javascript", the response of the query service should be in the form of JavaScript, or some runtime errors might occur. As for how to make use of the response of query service, we can see Fig. 13.

The pseudo-code is taken from pagetailor.js. When we need the user preferences of a Web page, the function named query would be invoked (Line 01). What the function does is as described in the last paragraph. There are three parameters in the query string: url, callback, and id. When Configuration Manager receives a request, it would extract the values of each parameter. Among them, the value of url is employed to query the database to find the corresponding user preferences about this Web page. The value of callback contains the name ("callback" in this case) of a function to process the response of query service. And id, the last one, is utilized to identify the "script" element that makes this request, since more than one "script" element could be added to a Web page at the same time.

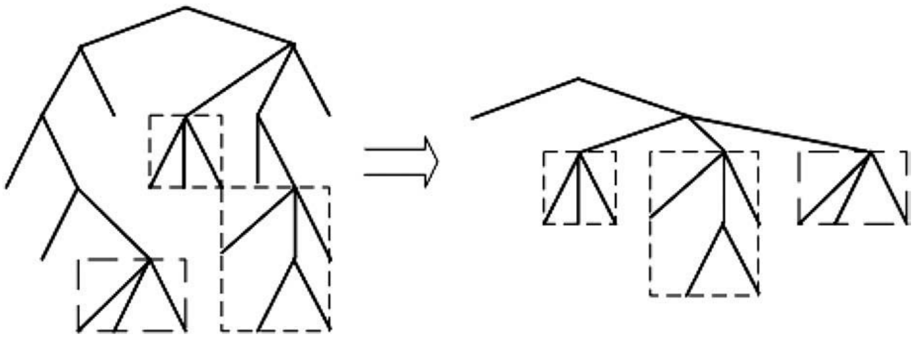


Fig. 17. A new DOM tree (right) would be created to hold the replicas of selected elements in original DOM tree (left).

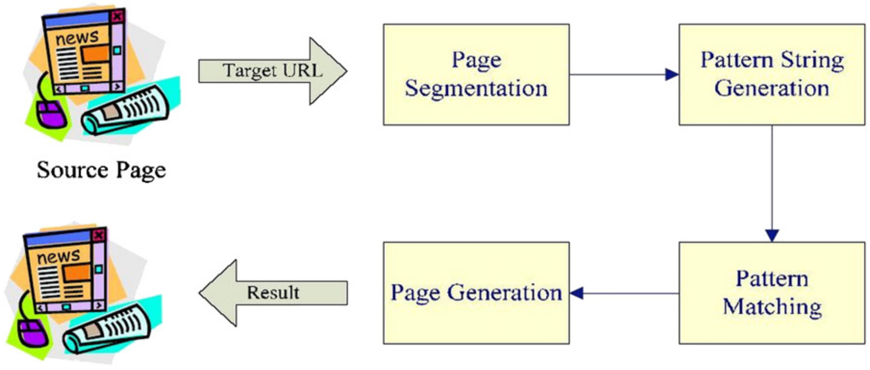


Fig. 18. An overview of pattern-oriented scheme for mobile Web page generation.

```
VipsTreeConstruct(Node treeNode, VipsNode nips Node){
  if(treeNode.AocValue < Pr-define AOC) return;
  else{
      for EACH child of treeNode{
        Construct a VipsNode and Put this child Node in it;
        VipsTreeConstruct(this ChildNode, current VipsNode);
      }
  }
}
```

Fig. 19. The VIPS tree construction algorithm.

As mentioned before, the response will be interpreted as JavaScript. What the Configuration Manager returns is actually a function invocation like the one in Fig. 14. The function name, "callback", is identical to the one involved in the query string. Two parameters are passed to this function. The first one contains the user preferences about the URL specified in the query string, the second one is the id specified in the query string as well, which is left intact.

When the response is sent back to the client side, the function named "callback" would be invoked with the user preferences filled by Configuration Manager. Since Page Tailor has obtained the user preferences, the reconstruction of the past could be performed in the body of this function. Finally, it comes to the id's turn. We can use the id to find the "script" element that was added to the Web page to trigger this series of events and remove it. Here the callback invocation is sent back from the Configuration Manager to the Page Tailor through the *iw* interface again.

The above descriptions illustrate how to load the user preferences without being restricted by the "same origin policy". Fig. 15, on the contrary, illustrates how to update the user preferences. It is not surprising that these two operations are very similar. They differ only in two respects. First, one more parameter is involved in the query string, which is the data required to update. Second, the callback function only serves as a scavenger to remove the "script" element.

3.2. Configuration Manager

We adopt Ruby on Rails [11] to implement the Configuration Manager. Controllers are the subprograms in a Rails application that performs tasks. Controller actions are sequences of Ruby code that correspond directly to the tasks this application can be asked to do. When an action is executed, it not only has access to the data from a submitted form but also to the models. In our system, two Web services exported by Configuration Manager are implemented as separate actions. That is, query_service and update_service. These two interfaces enable the Page Tailor to manipulate the data stored in the backend database.

3.3. Mobile Proxy

Mobile Proxy in our system is developed from Muffin [12]. We can modify the content of the Web page before it is sent back to the client according the user preferences specified by the user. The interfaces in the filter logic class depend on what kind of filtering we intend to perform. There are two interfaces designed: RequestFilter and ReplyFilter. The RequestFilter interface is to filter requests before they are sent to a server. On the contrary, the ReplyFilter is to modify

```
var getAOCValue(Node){
  var value=0;
  if(Node is Text Node){
      return Length*FontSize;
  }else if(Node is Image Node){
      return ImageSize;
  }
  else if(Node is Element Node){
      for EACH child of Node{
        value=value+getAOCValue(this ChildNode);
      }
  }
  return value;
}
```

Fig. 20. The algorithm of getting the AOC value.

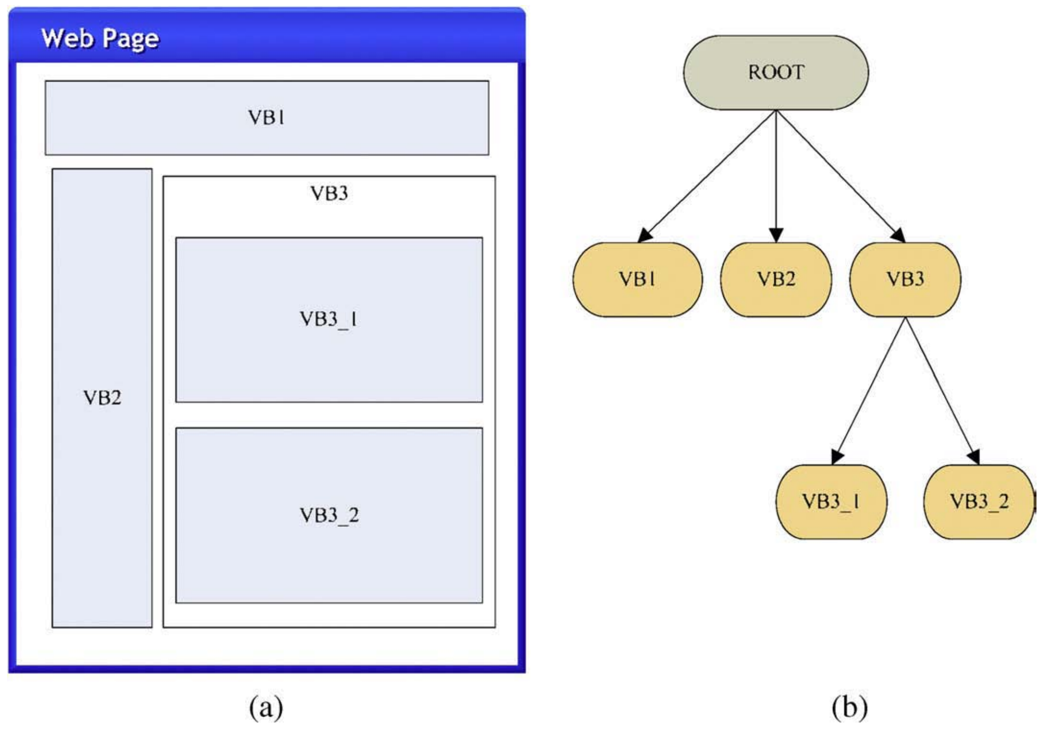


Fig. 21. An example of VIPS tree construction.

replies after a server responds. These two interfaces follow the HTTP 1.0 standard of *iw* interface.

3.3.1. Pre-processing of Web Content

If a request issued by the client is destined for a URL that has corresponding user preferences stored in our database, Mobile Proxy would start a chain of processing steps. First of all, it would use a HTML parser, NekoHTML [13] to parse that Web page and produce a corresponding DOM tree (Fig. 16). After that, an XPath engine, Jaxen [14] is employed to process XPath queries over the DOM tree. Some specific elements would be selected at this time, and a new DOM tree would be created alongside to hold the replicas of these elements (Fig. 17). The sequence of the replicas in the new DOM tree would refer to the sequence of the XPath expressions in the user preferences.

3.3.2. Transforming to XHTML

The result page (i.e. the new DOM tree) has already been constructed now. However, it is not necessarily self-validating because what we have done already is to find, clone and collect the blocks of content directly. To further polish that, an open source utility, JTidy [15], would be used. This tool was originally designed to fix mark-up errors and also offers a means to convert existing HTML content into

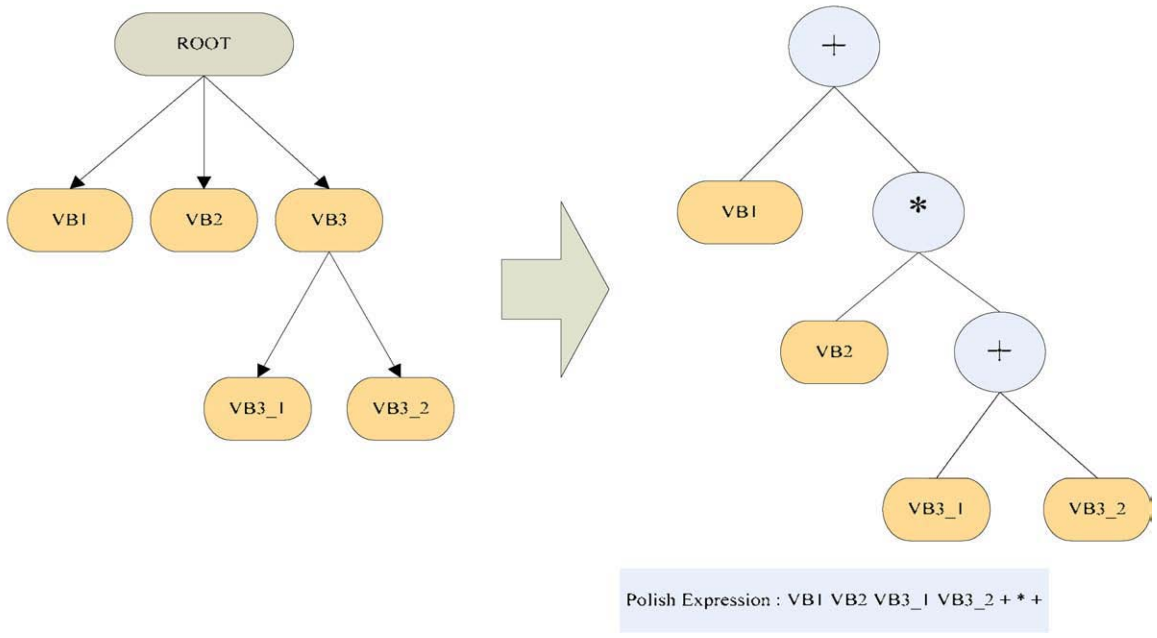


Fig. 22. An example of slicing tree construction.

```
var getNodeType(Node){
  if(Node is Text Node){
      if(ParentNode of Node is Anchor Node){
        return Index Type;
      }else{
        return Length*FontSize;
      }
  }else if(Node is Image Node or Object Node){
      return Content Type;
  }
  else if(Node is Element Node){
      var index, content;
      for EACH child of Node{
        var type=getNodeType(childNode);
        if(type is index){
            index=index+getAOCValue(childNode);
        }else{
            content=content+getAOCValue(childNode);
        }
      }
      if(index >= content){
        return Index Type;
      }else{
        return Content Type;
      }
  }
}
```

Fig. 23. The algorithm of getting the node type.

well-formed XML, such as XHTML. After the process of JTidy, a well-formed and validated result page is finally turned up.

4. Automatic mobile Web page generation

Our mobile Web page generation flow consists of four major steps: page segmentation, pattern string generation, pattern matching and page generation (Fig. 18). In the first step – page segmentation, we apply the VIPS algorithm into segment the target page into small blocks and construct a VIPS tree. In the second step – pattern string generation, we transform the VIPS tree into a slicing tree and generate a pattern string – the Polish Expression [16] of this binary slicing tree. In the third step – pattern matching, we compare this pattern string with all existing pattern strings. If there is any existing pattern string matched, we take the annotations of the existing pattern as result. In the fourth step – page generation, we generate the result page according to the source page and the annotations.

4.1. Page segmentation

In this step, we use parts of VIPS algorithm and change several method of it for our purpose (Fig. 19). We segment the Web page into blocks and retain the hierarchical structure. In order to extract the content blocks, we find explicit separators by analyzing the presence of tags such as <TABLE>, <TR>, <TD>, and <DIV>. In general, every node in the DOM tree can represent a visual block. Different from VIPS algorithm, we used the AoC (Area of Content) value instead of DoC (Degree of Coherence) value. Each node will be assigned an AoC value to indicate how many areas are consumed by the content within the node (Fig. 20). We extract the content blocks recursively while the AoC value of the node is great than pre-defined AoC. After all the blocks are processed, we obtain a vision-based content structure tree (Fig. 21).

4.2. Pattern string generation

In order to represent the page structure in a string, we transform the VIPS tree to a binary slicing tree (Fig. 22). After the binary slicing tree is constructed, we perform a postorder traversal of it and obtain a string — Polish Expression.

For transforming the VIPS tree to a slicing tree, we perform a level-order traversal of the VIPS tree. In the traversal, we construct corresponding slicing tree node according to different states of VIPS tree node. If the VIPS node has next sibling node, we construct an operator node and put these two VIPS tree node as its child nodes. If the VIPS node has child nodes, we construct an operator node to the slicing tree. Furthermore, if the VIPS node is a left node, we construct a content node. According to the tag name of the node, we assign the operator type to each operator node. The node with tag name "TD" is assigned to a "*" symbol, and the node with other tag is assigned to a "+" symbol (Fig. 23).

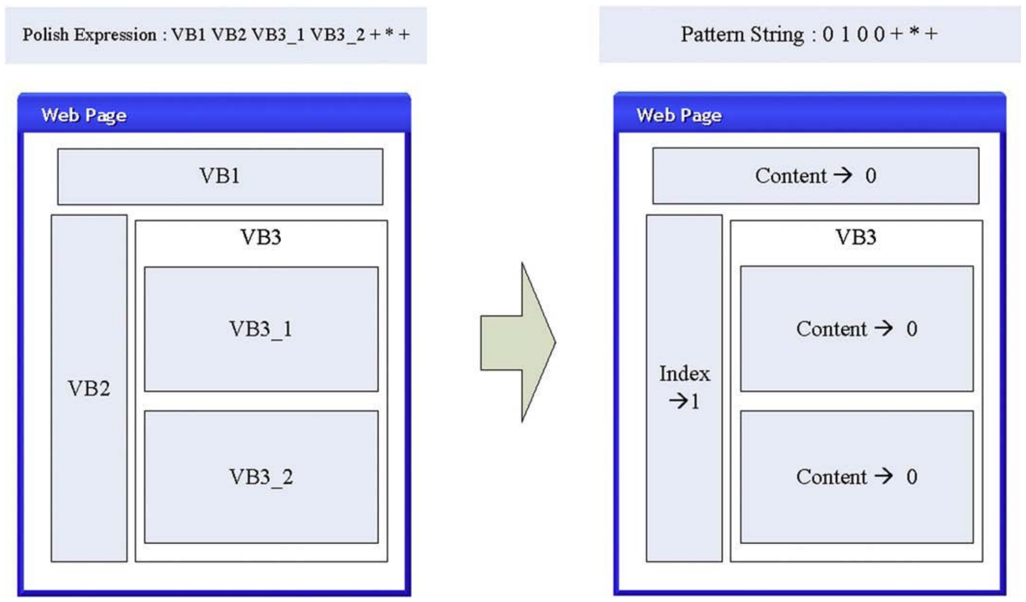


Fig. 24. An example of pattern string generation.

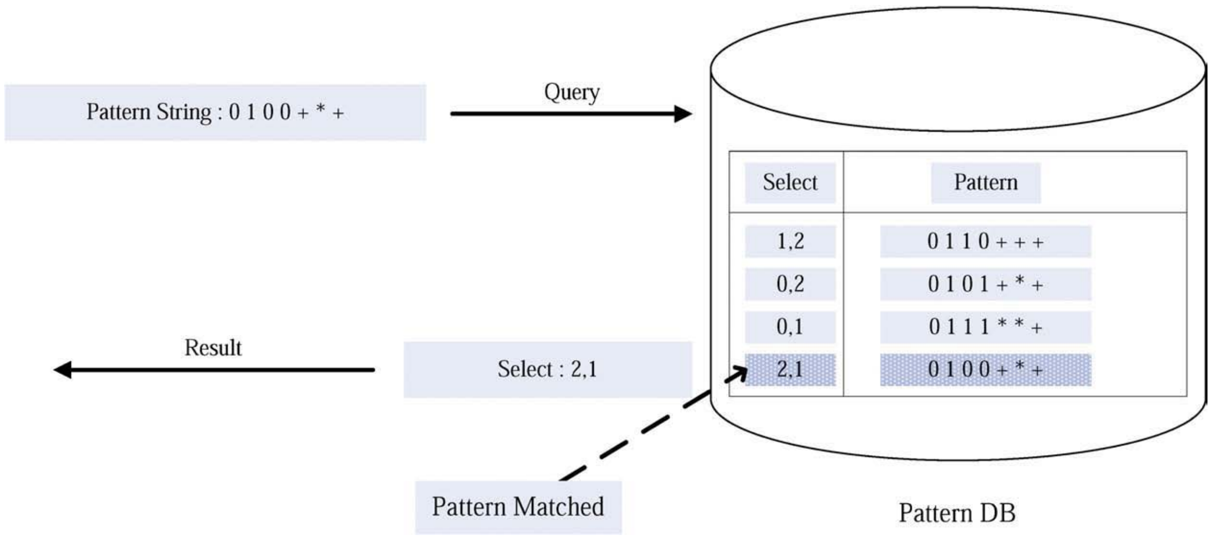


Fig. 25. An example of pattern matching.

4.3. Pattern matching

The Polish Expression represents a slicing tree structure in a string (Fig. 24). We can compare the tree structures by comparing their Polish Expression strings. If two trees have the same Polish Expression strings, they have the same tree structure. However, the tree structure of Web pages is usually complex. It is difficult to find two pages that have the same pattern string. For Web page similar comparisons, we compare two pattern strings from beginning, and find out the number of the same characters. We divide this number by the string length and obtain a similar value. If this similar value is larger than the pre-defined value, these two patterns are matched.

4.4. Page generation

After pattern matching process, we obtain a page annotation for the Web page. As in Fig. 25, the annotation contains a sequence of block numbers. There are commas to separate the sequence. A block number is the position of the block in the pattern string. According to this sequence, we represent these blocks as the output page (Fig. 26).

When a mobile device accesses the Web page through the Mobile Proxy for the first time, the mobile Web page will be automatically generated following these four steps. At the same time, the result page annotation will be stored into the Configuration Manager. The customization of user's mobile Web page can be done by the Page Tailor mentioned in the previous section.

5. Evaluation

In this section, we evaluate our Web page tailoring system in different aspects. A practical example is also given to show how this system can eliminate the unnecessary scrolling by filtering the unwanted Web page content.

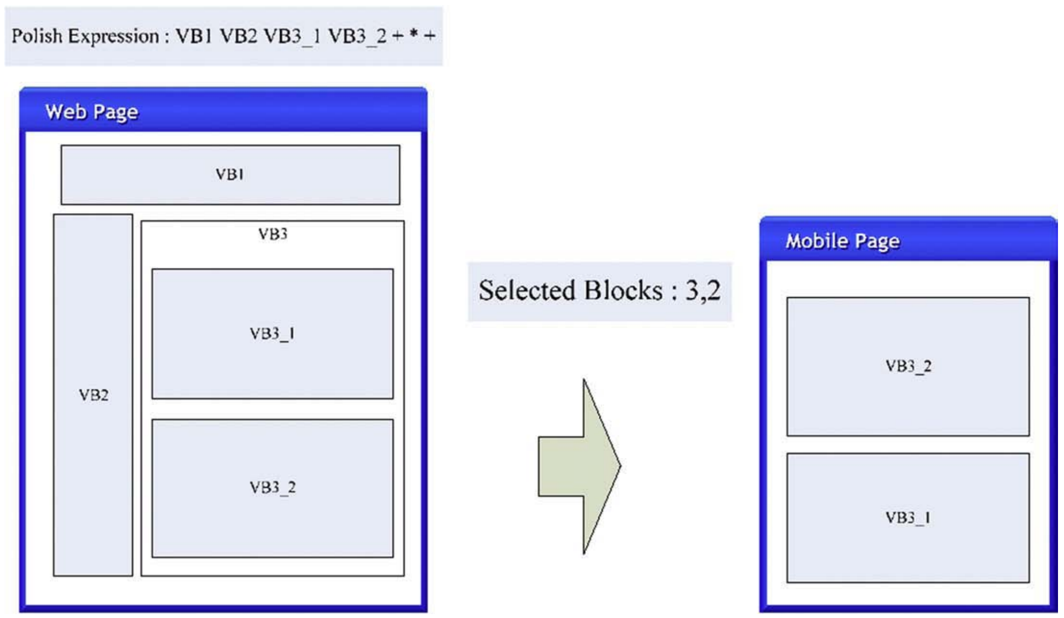


Fig. 26. An example of page generation.

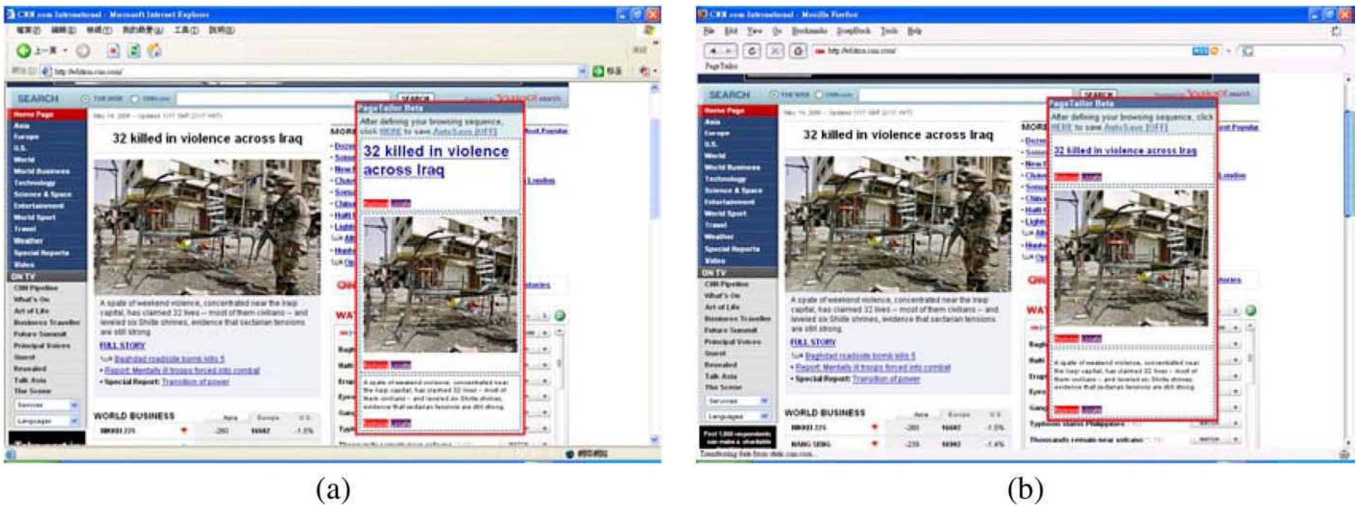


Fig. 27. Usability test (a) Page Tailor in Internet Explorer (b) Page Tailor in Firefox Web browser.

5.1. Usability test

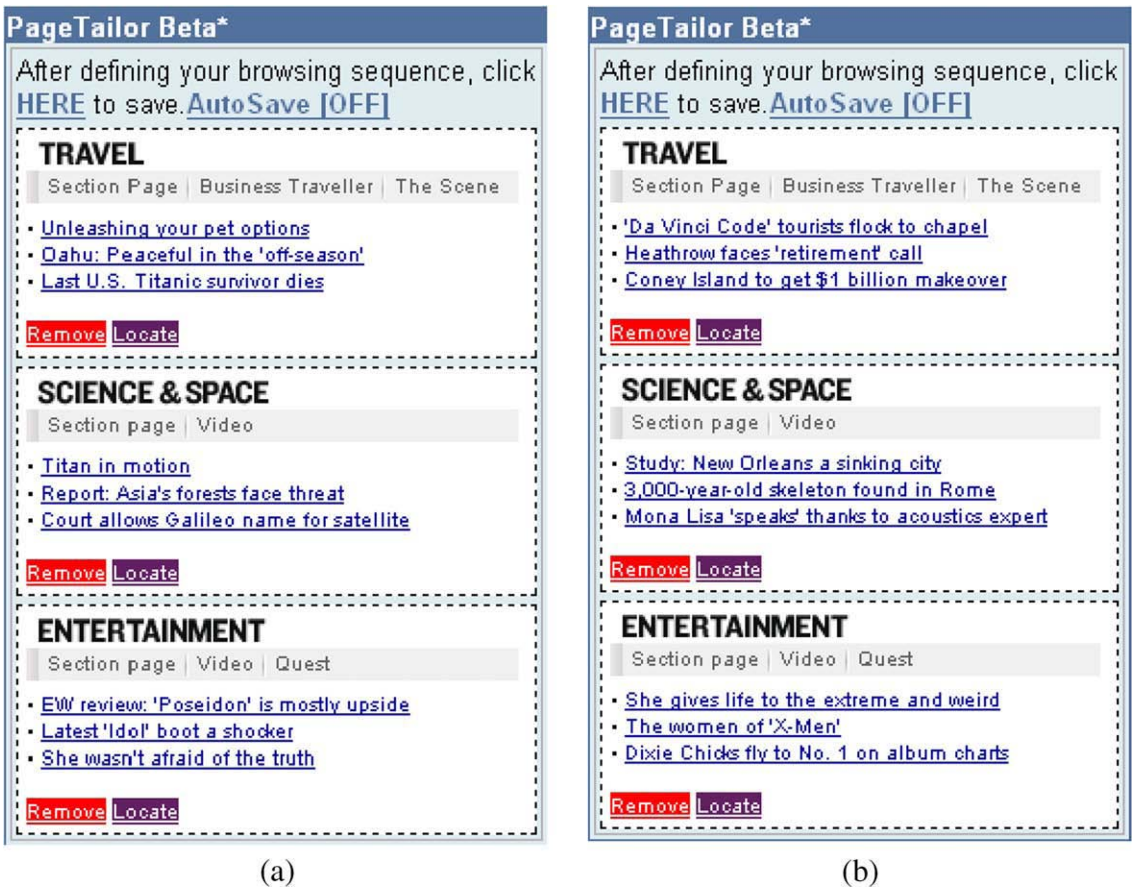
To test the usability of Page Tailor on different browsers, we personalize a Web page in Internet Explorer and launch Page Tailor (on the same Web page) in Firefox Web browser to check if the same result can be obtained, and vise versa. Fig. 27 shows the result of the test.

5.2. Stability test

To test whether the user preferences of a Web page can really be employed to extract accurately the blocks of content, we make a continuous check on the result. Fig. 28 contains two snapshots that are taken on different days without modifying the user preferences.

5.3. Example

In this sub-section, we show the unnecessary information eliminated result using our system. Also, we compare our system to Google Mobile Proxy. As seen in Fig. 29 (a), only parts of the Web page can be presented on the limited screen at one time. There is plenty of room for scrolling in both horizontal and vertical directions. In Fig. 29 (b), we personalize this Web page by selecting three blocks of



(a)

PageTailor Beta*

After defining your browsing sequence, click HERE to save. AutoSave [OFF]

TRAVEL
Section Page | Business Traveller | The Scene
- Unleashing your pet options
- Oahu: Peaceful in the 'off-season'
- Last U.S. Titanic survivor dies

Remove Locate

SCIENCE & SPACE
Section page | Video
- Titan in motion
- Report: Asia's forests face threat
- Court allows Galileo name for satellite

Remove Locate

ENTERTAINMENT
Section page | Video | Quest
- EW review: 'Poseidon' is mostly upside
- Latest 'Idol' boot a shocker
- She wasn't afraid of the truth

Remove Locate

(b)

PageTailor Beta*

After defining your browsing sequence, click HERE to save. AutoSave [OFF]

TRAVEL
Section Page | Business Traveller | The Scene
- 'Da Vinci Code' tourists flock to chapel
- Heathrow faces 'retirement' call
- Coney Island to get $1 billion makeover

Remove Locate

SCIENCE & SPACE
Section page | Video
- Study: New Orleans a sinking city
- 3,000-year-old skeleton found in Rome
- Mona Lisa 'speaks' thanks to acoustics expert

Remove Locate

ENTERTAINMENT
Section page | Video | Quest
- She gives life to the extreme and weird
- The women of 'X-Men'
- Dixie Chicks fly to No. 1 on album charts

Remove Locate

Fig. 28. Stability test.

content that are located in the middle of this page. After the personalizing process (shown in Fig. 29 (c)), the unwanted Web page content will be filtered. The result Web page of our system is precise and users can read this page with a glance.

Compared to Google Mobile Proxy, which is shown in Fig. 30. We can see that the result Web page of the Google Mobile Proxy is still too long in length for scrolling up and down, as in Fig. 30 (c). There is still some unimportant information that maybe the users are not interested in, as in Fig. 30 (d). Compared to our result Web page, as in Fig. 29 (c), our result page can be more personalized and precise for page presentation.

6. Related works

In this section, content adaptation approaches were introduced. Commercial products and related researches that fall into these three categories were described subsequently. Unfortunately, most of the adaptation results are not perfect. For example, it is hard to locate the wanted information in the reformatted page. A user might have to scroll through tons of irrelevant stuff before he/she can exactly find what he/she wants. On the contrary, some results contain too few information to view for users, users cannot find out what they want in the shrunk mobile Web page.

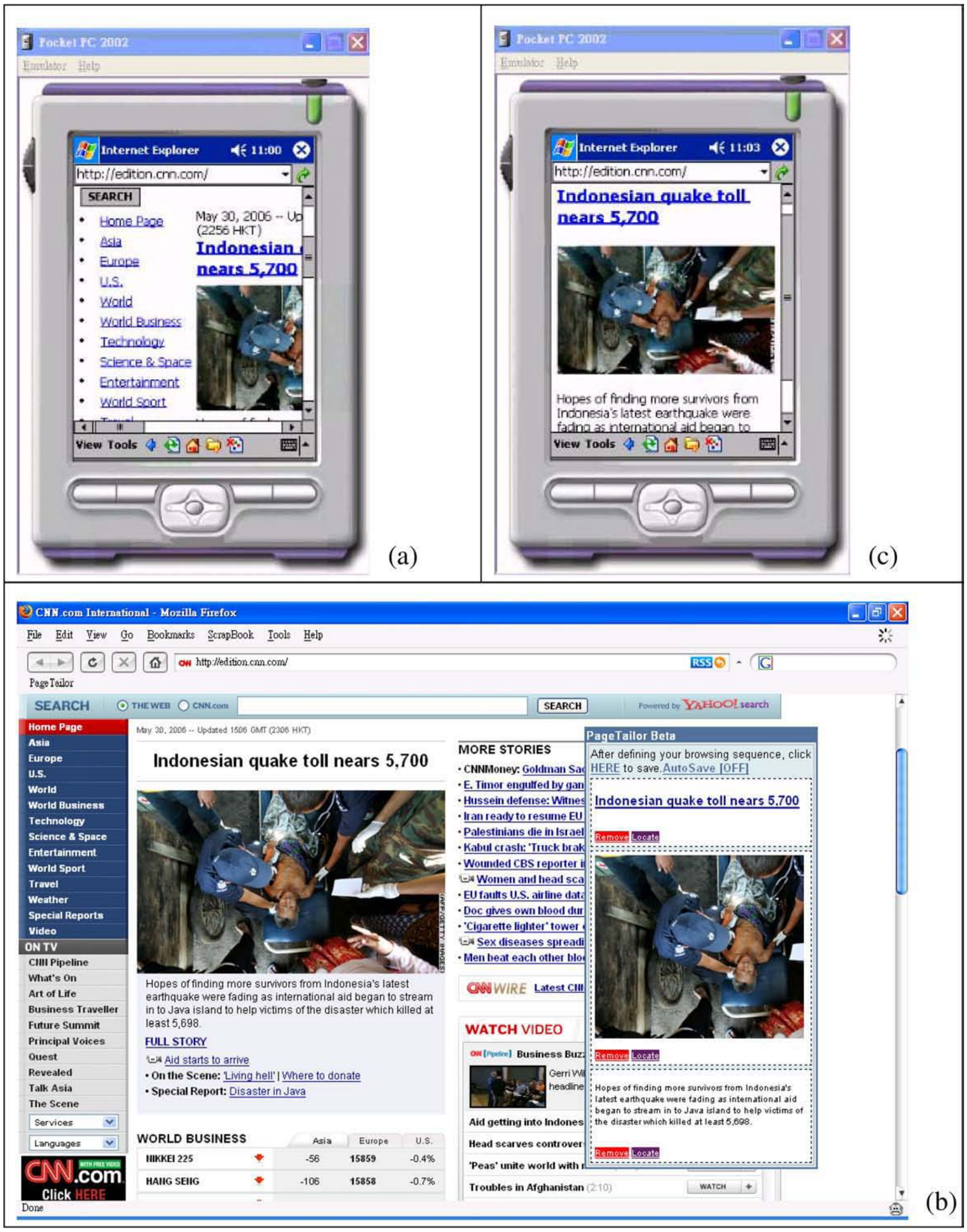
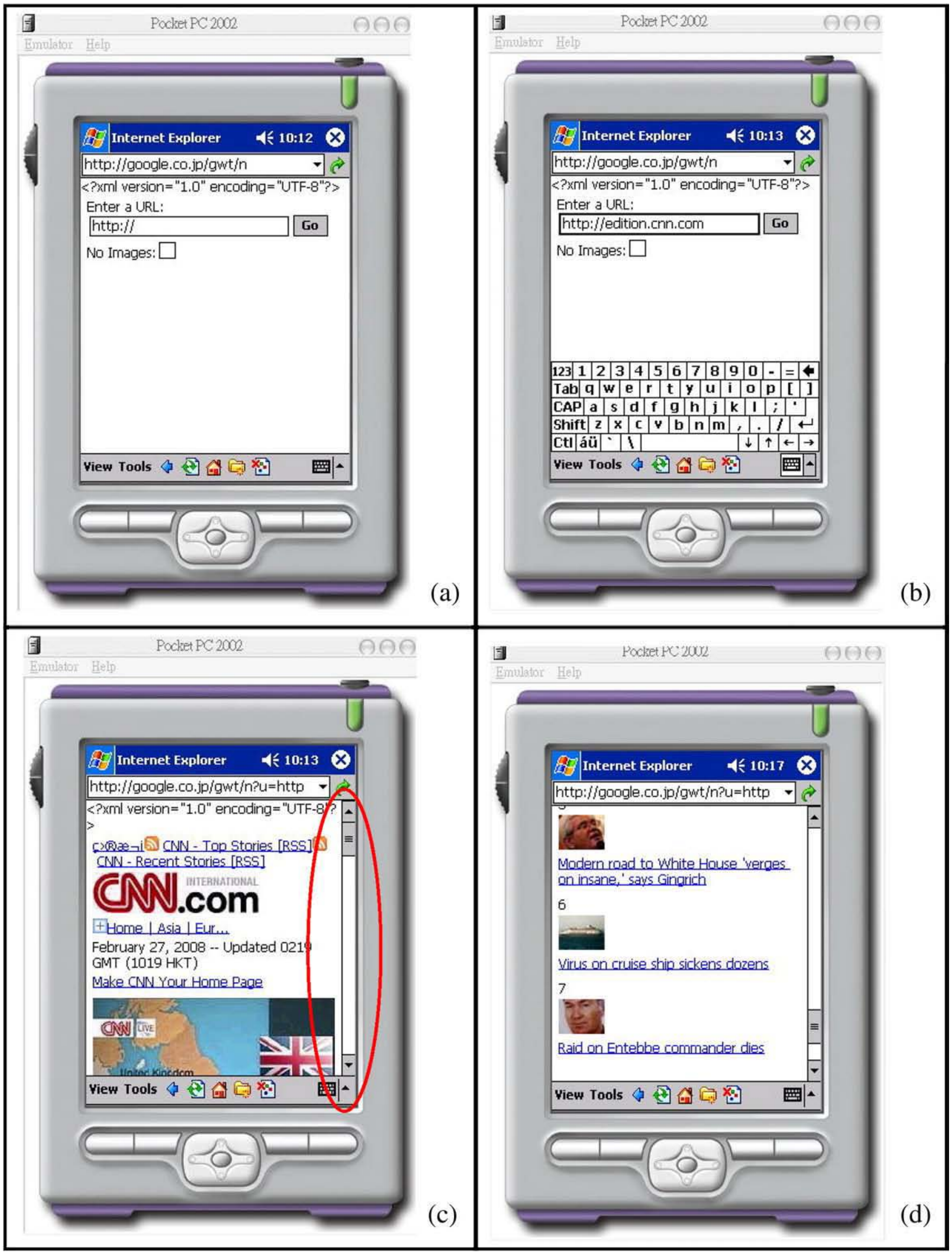

Pocket PC 2002
Emulator Help
Internet Explorer 11:00
http://edition.cnn.com/
SEARCH
Home Page
Asia
Europe
U.S.
World
World Business
Technology
Science & Space
Entertainment
World Sport
May 30, 2006 -- Up (2256 HKT)
Indonesian
nears 5,700
View Tools
(a)
Pocket PC 2002
Emulator Help
Internet Explorer 11:03
http://edition.cnn.com/
Indonesian quake toll nears 5,700
Hopes of finding more survivors from Indonesia's latest earthquake were fading as international aid began to
View Tools
(c)
CNN.com International - Mozilla Firefox
File Edit View Go Bookmarks ScrapBook Tools Help
http://edition.cnn.com/
PageTailor
SEARCH THE WEB CNN.com SEARCH Powered by YAHOO! search
Home Page
Asia
Europe
U.S.
World
World Business
Technology
Science & Space
Entertainment
World Sport
Travel
Weather
Special Reports
Video
ON TV
CNN Pipeline
What's On
Art of Life
Business Traveller
Future Summit
Principal Voices
Quest
Revealed
Talk Asia
The Scene
Services
Languages
CNN.com Click HERE
Done
May 30, 2006 -- Updated 1506 GMT (2306 HKT)
Indonesian quake toll nears 5,700
Hopes of finding more survivors from Indonesia's latest earthquake were fading as international aid began to stream in to Java island to help victims of the disaster which killed at least 5,698.
FULL STORY
Aid starts to arrive
• On the Scene: 'Living hell' | Where to donate
• Special Report: Disaster in Java
WORLD BUSINESS Asia Europe U.S.
NIKKEI 225 -56 15859 -0.4%
HANG SENG -106 15858 -0.7%
MORE STORIES
• CNNMoney: Goldman Sa
• E. Timor engulfed by gan
• Hussein defense: Witnes
• Iran ready to resume EU
• Palestinians die in Israel
• Kabul crash: 'Truck brak
• Wounded CBS reporter i
• Women and head sca
• EU faults U.S. airline data
• Doc gives own blood dur
• 'Cigarette lighter' tower
• Sex diseases spreadi
• Men beat each other blo
CNN WIRE Latest CNN
WATCH VIDEO
Business Buzz
Aid getting into Indones
Head scarves controver
'Peas' unite world with
Troubles in Afghanistan (2:10) WATCH
PageTailor Beta
After defining your browsing sequence, click HERE to save. AutoSave [OFF]
Indonesian quake toll nears 5,700
Remove Locate
Remove Locate
Hopes of finding more survivors from Indonesia's latest earthquake were fading as international aid began to stream in to Java island to help victims of the disaster which killed at least 5,698.
Remove Locate
(b)

Fig. 29. A practical example of how to use our tool.



(a)
http://google.co.jp/gwt/n
<?xml version="1.0" encoding="UTF-8"?>
Enter a URL: http://
No Images: ☐

(b)
http://google.co.jp/gwt/n
<?xml version="1.0" encoding="UTF-8"?>
Enter a URL: http://edition.cnn.com
No Images: ☐

(c)
http://google.co.jp/gwt/n?u=http
<?xml version="1.0" encoding="UTF-8"?>
ç®æ¬¡ CNN - Top Stories [RSS]
CNN - Recent Stories [RSS]
CNN.com INTERNATIONAL
Home | Asia | Eur...
February 27, 2008 -- Updated 0219 GMT (1019 HKT)
Make CNN Your Home Page

(d)
http://google.co.jp/gwt/n?u=http
Modern road to White House 'verges on insane,' says Gingrich
6
Virus on cruise ship sickens dozens
7
Raid on Entebbe commander dies

Fig. 30. The result page of Google Mobile Proxy.

There are three categories of mobile content adaptation: client-based application adaptation, client-server application adaptation, and proxy-based application adaptation [17]. Content adaptation can occur in the client device, on the content server or in an intermediate proxy server. In the client-based application adaptation, the required transcoding is performed by the client device. In the service-based application approach, one common way of providing content to different devices is to store the content as XML, and then use XSLT to convert the content to appropriate markup languages. In the proxy-based application approach, a proxy server analyzes and transcodes the content on-the-fly, before sending the result to the client.

6.1. Client-based application adaptation

Opera's mobile browser includes Small-Screen Rendering technology [18]. This technology intelligently reformats today's Web pages to fit the screen width of mobile devices, thereby eliminating the need for horizontal scrolling. It is only the layout of the page that is changed. All the content and functionality remain. A full-featured Opera Mini (Opera's Web browser product) simulator is available at Ref. [19]. Another company named ACCESS also has their proprietary browser product NetFront. The rendering technology adopted by this browser is what they call Smart-Fit Rendering [20]. Smart-Fit Rendering, just like Opera's Small-Screen Rendering, also renders Web pages to fit the narrow screen width of mobile devices.

6.2. Client-server application adaptation

Some of the optimal context adaptation model has been proposed, such as Ref. [21], which is a server-based adaptive Web system. They claim that the result of their adaptation is optimal, and both the response time and network traffic can be reduced. Also, Lum and Lau have proposed an adaptation system based on decision policy [25]. They designed the decision policy from contextual information to transcoding strategies. This system computes scores of each possible version of content. Moreover, Lee, Chandranmenon and Miller developed a middleware-based content adaptation server, named GAMMAR [26]. GAMMAR use a table-driven architecture to manage the transcoding service, which located among a cluster of network computers.

6.3. Proxy-based application adaptation

Google had released a service to the public, similar to Refs. [22] and [23]. Since this service still lacks official statements, here we temporarily call it "Google Mobile Proxy" [24]. This proxy provides some distinctive features, such as pagination and links collapsing. Pagination means the process of dividing and numbering documents into pages.

7. Conclusion and future work

7.1. Conclusion

Motivated by the increasing needs of browsing Web pages on mobile devices, this paper focuses on the inconvenience of limited screen size of mobile device. This kind of inconvenience is the common feature of most of the mobile devices. We propose a Web page tailoring tool to help end users to personalize their mobile Web pages. The contributions of this paper including:

1. A cross-browser configuration tool is designed.
2. The web-based nature of our configuration tool allows a user to configure the settings from different computers, and requires no pre-installation of any software.
3. Blocks in a Web page can be chosen correctly under the premise that the layout of a Web page does not change frequently.
4. A web-based management interface is provided.
5. An automatic algorithm for mobile Web page generation is proposed.

7.2. Future work

There are several aspects could be improved of our Web page tailoring system. For example, the functionality of Mobile Proxy can be further extended. At this point, it is only capable of generating XHTML, and the mechanism for detecting the capabilities of mobile devices has not been integrated yet. On the other hand, our system has not been completely evaluated yet, we will accomplish the evaluation in the future.

References

[1] Mobile ASP.NET Web Applications, http://www.asp.net/default.aspx?tabIndex=6&tabId44.
[2] Jen-Kai Wu, Shyan-Ming Yuan, "A Visualized Kit for Developing Applications on Multiple Mobile Devices", Department of Computer Science and Engineering National Chiao Tung University Master Thesis, 2005 June.
[3] Chi-Han Kao, Shyan-Ming Yuan, "A Multi User-interface Generation Plug-in for Visual Studio.NET", Department of Computer Science and Engineering National Chiao Tung University Master Thesis, 2005 June.
[4] Sheng-Po Shen, Shyan-Ming Yuan, "XML-based Mobile Application Development Framework", Department of Computer Science and Engineering National Chiao Tung University Master Thesis, 2004 June.
[5] Google Mobile, http://www.google.com/mobile/index.html.
[6] Yahoo Mobile, http://mobile.yahoo.com/.
[7] Jinlin Chen, Baoyao Zhou, Jin Shi, Hongjiang Zhang, Qiu Fengwu, Function-based object model towards website adaptation, Proceedings of the 10th international conference on World Wide Web, May 01–05 2001, pp. 587–596, Hong Kong, Hong Kong.
[8] Browser Market Share Survey by NetApplications.com, http://netapplications.com.
[9] W3C. XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath.
[10] Script.aculo.us JavaScript framework, http://script.aculo.us/.
[11] Ruby on Rails, http://www.rubyonrails.org/.
[12] Muffin World Wide Web Filtering System, http://muffin.doit.org/.
[13] NekoHTML, http://people.apache.org/~andyc/neko/doc/html/index.html.
[14] Jaxen, http://jaxen.org/.
[15] JTidy, http://jtidy.sourceforge.net/.
[16] M. Lai and D. Wong. Slicing tree is a complete floorplan representation. In DATE '01: Proceedings of the Conference on Design, Automation and Test in Europe, pages 228–232.
[17] Jin Jing, Abdelsalam Helal, Ahmed Elmagarmid, Client-server computing in mobile environments, ACM Computing Surveys 31 (2) (1999) 117–157.
[18] Opera's Small-Screen Rendering ™, http://www.opera.com/products/mobile/smallscreen/.
[19] Opera Mini ™ simulator, http://www.opera.com/products/mobile/operamini/demo.dml.
[20] ACCESS Smart-Fit Rendering ™ technology, http://www.access-us-inc.com/Products/client-side/Prod_NetFront.html.
[21] Rong-Hong Jan, Ching-Peng Lin, Maw-Sheng Chern, An optimization model for Web content adaptation, Computer Networks 50 (7) (2006) 953–965 (SCI, EI) (NSC 93-2219-E-009-002 and NSC 93-2752-E-009-005-PAE).
[22] Skweezer, http://www.skweezer.net/.
[23] IYHI, http://www.iyhy.com/.
[24] Google Mobile Content Proxy, http://www.google.com/gwt/n.
[25] Wai Yip Lum, Francis C.M. Lau, A context-aware decision engine for content adaptation, IEEE Pervasive computing 1 (3) (July–September 2002).
[26] Yui-Wah Lee, Girish Chandranmenon and Scott C. "GAMMAR: A Content-adaptation Server for Wireless Multimedia Applications", Bell Laboratories.

Yung-Wei Kao was born on March 12, 1982 in Taipei, Taiwan, Republic of China. He received his MBA degree in Department of Information Management of National Central University in 2006. His interests are in Web 2.0 and network security.

Tzu-han Kao was born on December 20, 1976 in Taichung, Taiwan, Republic of China. He received his BS degree in Computer Science and Information Engineering from Chung Hua University Taiwan, in 2000. He now received PHD degree from Institute of Computer Science & Engineering, in Department of Computer Science of National Chiao Tung University in 2006. His interests are in Web 2.0, Context-aware, Ubiquitous, and Pervasive Computing.

Shyan-Ming Yuan was born on July 11, 1959 in Mauli, Taiwan, Republic of China. He received his BSEE degree from National Taiwan University in 1981, his MS degree in Computer Science from University of Maryland, Baltimore County in 1985, and his PhD degree in Computer Science from the University of Maryland College Park in 1989. Dr. Yuan joined the Electronics Research and Service Organization, Industrial Technology Research Institute as a Research Member in October 1989. Since September 1990, he has been an Associate Professor at the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan. He became a Professor in June 1995. His current research interests include Distributed Objects, Internet Technologies, and Software System Integration. Dr. Yuan is a member of ACM and IEEE.