COMPUTER
ENGINEERING
PROGRAM

Middle East Technical
University
Northern Cyprus
Campus

eMINE Technical Report Deliverable 8 (D8),

October 2013

Implementation and Evaluation of Web Page Transcoding Based on Eye Tracking

M. Elgin Akpınar
elgin.akpinar@metu.edu.tr
Middle East Technical University,
Ankara, Turkey

Yeliz Yesilada
yyeliz@metu.edu.tr
Middle East Technical University
Northern Cyprus Campus,
Kalkanlı, Güzelyurt, TRNC,
Mersin 10, Turkey

As web technologies evolve and enable designers to create more visually interactive and consequently complex web pages, unfortunately, accessing these pages in alternative forms such as by using screen readers or mobile devices, is still a major problem. In order to solve this problem, we proposed a transcoding method which combines the aspects of eye tracking studies and heuristic roles of visual elements. In this technical report, we first implement the algorithm in two different modes and then evaluate the proposed transcoding method. Our technical evaluation showed that, block based transcoding improves the accessibility by reducing the access time to certain visual elements in a web page, loading time of the pages, CPU usage, total number of requests and downloaded file size for web pages.

METU
NCC

eMINE

The World Wide Web (web) has moved from the Desktop and now is ubiquitous. It can be accessed by a small device while the user is mobile or it can be accessed in audio if the user cannot see the content, for instance visually disabled users who use screen readers. However, since web pages are mainly designed for visual interaction; it is almost impossible to access them in alternative forms. Our overarching goal is to improve the user experience in such constrained environments by using a novel application of eye tracking technology. In brief, by relating scanpaths to the underlying source code of web pages, we aim to transcode web pages such that they are easier to access in constrained environments.

Acknowledgements

The project is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) with the grant number 109E251. As such the authors would like to thank to (TÜBİTAK) for their continued support.

Contents

1 Introduction **1**

2 Role Path Construction **1**
2.1 Common Scanpaths 2
2.2 Common Role Path 3

3 Implementation **4**
3.1 Block Based Transcoding 5
3.2 Role Based Transcoding 6

4 Evaluation **10**
4.1 Access Time Comparison Results 11
4.2 Visual Comparison Results 12
4.3 Mobile Simulation Results 12

5 Discussion **15**

6 Summary **15**

A Pages **17**
A.1 www.apple.com 17
A.2 www.babylon.com 18
A.3 uk.yahoo.com 19
A.4 www.godaddy.com 20
A.5 www.bbc.co.uk 21

B Charts **22**
B.1 www.apple.com 22
B.2 www.babylon.com 23
B.3 www.avg.com 24
B.4 uk.yahoo.com 25
B.5 www.godaddy.com 26
B.6 www.bbc.co.uk 27

Middle East Technical University,
Northern Cyprus Campus,
Kalkanlı, Güzelyurt, TRNC,
Mersin 10, TURKEY

Tel: +90 (392) 661 2000
http://www.ncc.metu.edu.tr/

Corresponding author:
M. Elgin Akpınar
elgin.akpinar@metu.edu.tr

1 Introduction

As the web technologies improve, more attractive and visually interactive pages have been created by the designers. However, this interaction and attraction come with some accessibility problems. Accessibility of disabled people and mobile accessibility are two major areas in which classical web design patterns decrease the accessibility of the page. In brief, assistive technologies, such as screen readers, follows the node order in DOM structure of the web page. This makes browsing a time-consuming and strenuous for disabled users (Borodin et al, 2007; Mahmud et al, 2007a,b). Moreover, mobile devices are also not capable of providing the same user experience in PCs or laptops, due to some limited capabilities and resources of these devices (Hattori et al, 2007; Yin and Lee, 2004; Zakas, 2013). The limitations of mobile devices are small screens (Ahmadi and Kong, 2012; Milic-Frayling and Sommerer, 2002), limited memory and bandwidth (Xie et al, 2005; Zakas, 2013), and absence of a keyboard or a mouse (Chen et al, 2009; Xiao et al, 2005; Xie et al, 2005).

To address these problems, we proposed a web page transcoding algorithm which uses the aspects of eye tracking studies (Akpınar and Yeşilada, 2013c). Based on a predefined path, user navigates through the web page by following the order in the path. This path will be constructed with respect to the common scanpaths of pages, which are produced in eMINE Scanpath Analysis Algorithm (Eraslan et al, 2013b). The proposed system architecture is a proxy server, so that, accessibility problems of both disabled people and mobile device users are aimed to be solved. The transcoding application provides two different modes. In the first mode, the web page is transcoded with respect to the common scanpath of the web page which is loaded by the user as a JSON file. This file consists of the block names in the common scanpath. The other mode is based on the roles of visual elements. Using a general role path, the visual elements are grouped under their roles and transcoding application follows the order of roles in the common role path. In the proxy application, the page to be transcoded is segmented. Additionally, in role based transcoding mode, heuristic roles of visual elements are also detected. After certain types of user interaction, current index in the path changes and user views the corresponding visual element.

The aim of this technical report is to first describe the common scanpath to role path conversion process. Moreover, this report gives information about the implementation of transcoding algorithm in two different transcoding modes and basic functions of the demo application. Finally, it presents the findings of preliminary evaluation of the system and discusses the success of the approach.

The rest of this technical report has been organized in the following way: Section 2 explains our role path construction process and its results. Then, Section 3 gives detailed information about the implementation of the transcoding method. Section 4 describes our preliminary evaluation process and its results. Section 5 discusses our findings and future work. Finally, Section 6 concludes the technical report.

2 Role Path Construction

eMINE Scanpath Analysis Algorithm aims to generate a common scanpath by using eye tracking data and the visual element tree of web page after it segmented. It projects the fixations of eye tracking data over the visual elements and generates a path for each user.

Page	Common Scanpaths	
	Searching	**Browsing**
Apple	1	$2 \rightarrow 3$
Babylon	$1 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$	1
AVG	$1 \rightarrow 2 \rightarrow 1 \rightarrow 2$	$1 \rightarrow 1$
Yahoo	1	$1 \rightarrow 1 \rightarrow 2$
Godaddy	$1 \rightarrow 2 \rightarrow 2$	$1 \rightarrow 1 \rightarrow 3 \rightarrow 1$
BBC	1	$2 \rightarrow 2$

Table 1: Common scanpaths

Then, the algorithm calculates the most common scanpath by using these individual paths. First of all, it finds two most similar scanpaths by using Levenshtein Distance, and then finds the common scanpath by applying Longest Common Subsequence method over these two scanpaths. It iterates until only one scanpath is left in the list (Eraslan et al, 2013b; Yesilada et al, 2013). Moreover, further studies on eMINE Scanpath Analysis Algorithm gives detailed information about its implementation and evaluation as well as the correlation between scanpaths and visual element structure of a web page (Eraslan et al, 2013a). In this section, we first give brief information about the evaluation process of this algorithm and then using the results of this study, we convert visual element scanpaths to a single role path.

2.1 Common Scanpaths

In order to evaluate eMINE Scanpath Analysis Algorithm, Eraslan et al (2013a) conducted a a user study with six pages from different complexity groups. Users were told to complete two different types of tasks: searching and browsing. The results of this experiment are then used to calculate a common scanpath for each page by using eMINE Scanpath Analysis Algorithm. The results were analyzed for different sizes of participant sets, including 10, 20, 30 and over 30 participants and used to construct a single common scanpath for each page.

Six pages were used in this eye tracking experiment which include Apple.com, Babylon.com, Avg.com, Yahoo.com, Godaddy.com and Bbc.co.uk. These pages were selected with respect to their complexities. Apple and Babylon are low complexity pages, AVG and Yahoo are medium complexity pages and Godaddy and BBC are high complexity pages. In order to give an example, Figure 1 represents AVG Home Page. Border lines around content visualize the visual element structure of the web page. Screenshots for other pages are listed in Appendix A.

When eMINE Scanpath Analysis Algorithm was applied on eye tracking data, it returned a common scanpath for each page. The scanpaths for different pages categorized by tasks are listed in Table 1.

eMINE Scanpath Analysis Algorithm gives very detailed information about a common scanpath of a single page. However, we cannot compare two common scanpaths, because of the fact that, even if two visual elements in two different scanpaths are represented with the same numeric value; indeed, these two nodes are different visual elements. In order to solve this problem, we use the heuristic role detection algorithm which aims to label a visual element with respect to the role of it in the web page (Akpınar and Yeşilada, 2013a,b).

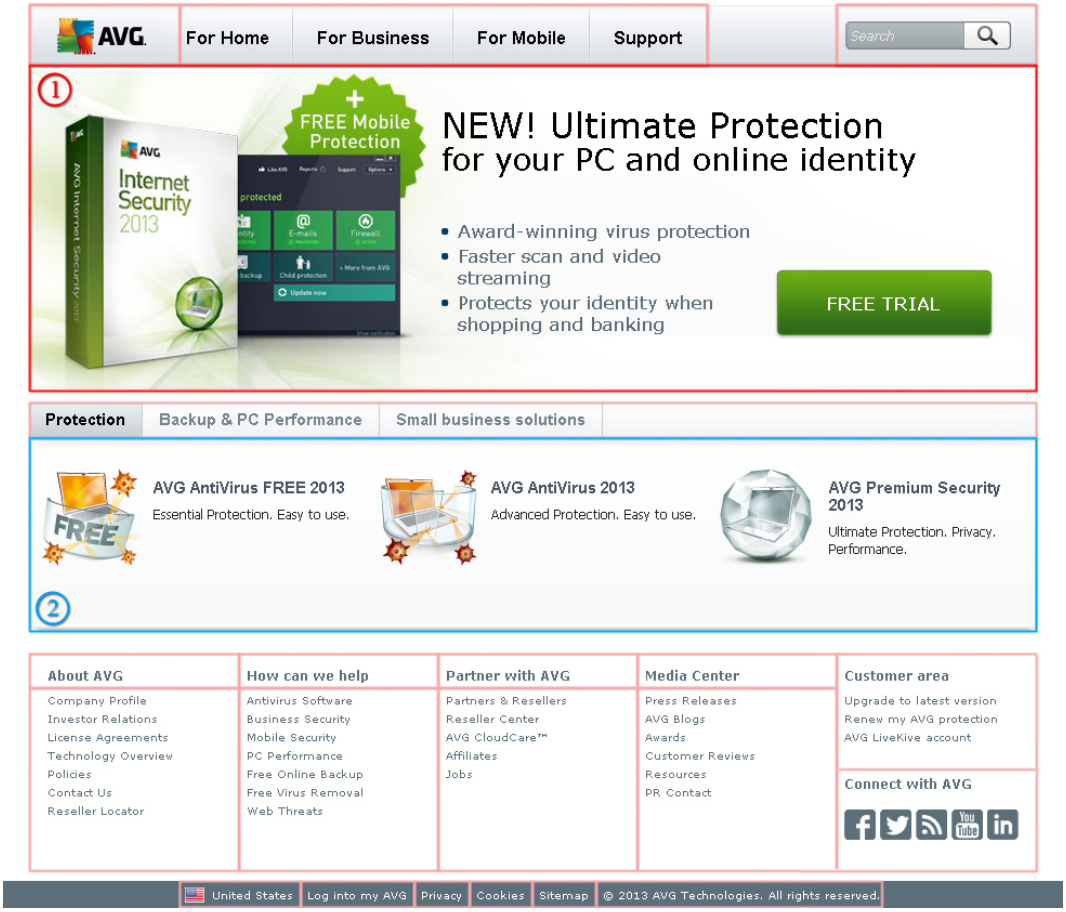


Figure 1: AVG Home Page

Using this labeling method, we convert the numeric pointers to role labels for each visual element.

2.2 Common Role Path

In order to construct a common role path for all pages to use in transcoding algorithm, we simply detect the roles of each visual element in Table 1, by using the heuristic role detection algorithm. After this conversion, the common scanpaths in Table 1 are converted to common role paths in Table 2, in which numeric instance names are replaced with the corresponding role of each visual element.

All of the roles in Table 2 are defined in the knowledge base of the role detection algorithm and assigned to the visual elements since the visual elements satisfied their criteria more than other roles. These blocks basically have a specific role. *Article* contains a set of paragraphs such as the main content of the page. *Title* is an identification name for a section of the page content. *SpecialGraphic* is an image which has a special meaning in the content. *Link*s are used to navigate user to internal or external sources. *List* is an array of items, for example, an array of links. *Container* is a special kind of role in the knowledge base. Visual

Task	Page	Role Path
Searching	Apple	Container(*SpecialGraphic*)
	Babylon	*SpecialGraphic* → *SpecialGraphic* → *SpecialGraphic* → *SpecialGraphic* → *SpecialGraphic* → *SpecialGraphic*
	AVG	*Article* → Container(*SpecialGraphic*, *Title*, *Link*) → *Article* → Container(*SpecialGraphic*, *Title*, *Link*)
	Yahoo	Container
	Godaddy	Container(*Title*, *Article*, *Link*) → Container(*Title*, *Link*) → Container(*Title*, *Link*)
	BBC	Container(*Title*, *Link*, *List*)
Browsing	Apple	*SpecialGraphic* → *SpecialGraphic*
	Babylon	*SpecialGraphic*
	AVG	*Article* → *Article*
	Yahoo	Container → Container → Container
	Godaddy	Container(*Title*, *Article*, *Link*) → Container(*Title*, *Article*, *Link*) → Container(*Icon*) → Container(*Title*, *Article*, *Link*)
	BBC	Container(*Link*, *List*) → Container(*Link*, *List*)

Table 2: Corresponding roles of visual elements in Table 1

elements which have *Container* role do not have a specific role in the page. Rather, they contain smaller visual elements, and these elements either have a specific purpose in the web page, or they are also container of smaller visual elements. For example, in Table 2, Container(*Title*, *Article*, *Link*) describes a visual element which contains a *Title*, an *Article* and a *Link*. There are many other roles in the knowledge base, however, these roles do not have any instances in Table 1.

When we analyze the role assignments and their relation with each other, we see that, the role diversity is very narrow, which only contains *SpecialGraphic*, *Article*, *Title*, *Link*, *List* and *Container*. Then, scanpaths of Apple, Yahoo and BCC include only one role. On the other hand, the scanpaths of Babylon, AVG and Godady, which are consisted of multiple nodes, contains duplicates of the same roles. For example, in the scanpath of Babylon, although original scanpath contains 5 different visual elements, all of them belong to the same role: *SpecialGraphic*. Another problem is with the high frequency of the *Container*. Although there are many instances of *Container*, the visual element roles, which they contain, differ than each other. Therefore, each *Container* which contains different types of visual elements should be separated than others. For example, Container(*Title*, *Article*, *Link*) and Container(*SpecialGraphic*, *Title*, *Link*) should be distinguished. In order to sum up, although the results of eye tracking experiments are very promising, it requires further user experiments to construct a general role path.

3 Implementation

In order to evaluate our transcoding approach, we implemented a demonstration of proposed system on Accessibility Tools Framework (ACTF) on Java platform[1]. ACTF is a framework and extensible infrastructure, which enables developers to build a variety of utilities for improving the accessibility of applications and content for people with disabilities. ACTF provides a IE based browser and interfaces to control this browser, such as changing textual

[1] `http://www.eclipse.org/actf/`

content of an element. Using these interfaces, DOM structure of a web page and its visual rendering properties are retrieved.

In the transcoding application, there are two modes: block based transcoding and role based transcoding. In block based transcoding, the web page is transcoded by using a common scanpath for the corresponding web page. The common scanpath is provided as a JSON file. In role based transcoding, the roles of visual elements in the web page are detected and the page is transcoded based on a role path. This role path is a generalized version of multiple common scanpaths and applies for all web pages.

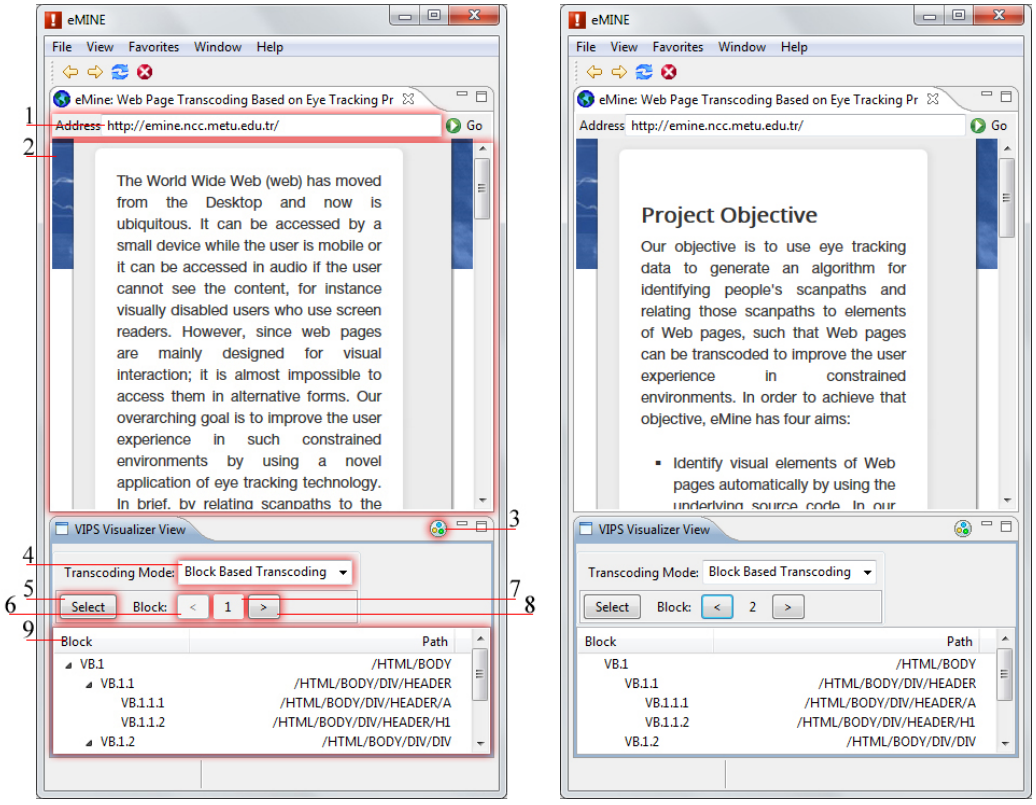
3.1 Block Based Transcoding

Block based transcoding enables users to transcode a web page by using a common scanpath, which is specific to the corresponding web page. The common scanpath is loaded as a JSON file to the application. This file is exported from eMINE Eye Tracking Application and contains a list of block names in the scanpath in a simple structure. Each block in the segmented web page structure has a block name which is assigned by the application. In block based transcoding application, these names are used to refer a particular block. An example of JSON file content is as follows:

```
{ "main_path": ["VB.1.1","VB.1.2.1","VB.1.3.2"]}
```

Figure 2a represents the demo application of block based transcoding approach. First of all, user types the URL of the web page in the address bar(1) and clicks on the ”Go” button. This loads the normal version of the web page in the content panel (2). Block based transcoding mode is the default mode of the application; however, user may change the mode by using transcoding mode selection box (4). If the user selects block based transcoding in the select box and clicks on segmentation button (3), the web page is segmented and file selection button is enabled (5). This button enables user to select the corresponding JSON file in a file selection panel. In order to transcode the web page, user must select a JSON file in an appropriate format from his/her local file system. After file selection, the content of the file is parsed and the list of block names are extracted. Based on this list, user can travel on the common scanpath of the web page by using previous block button (6) and next block button (8). These buttons are enabled or disabled with respect to the fact that, whether there is a block available after or before the current block. The current index is also displayed (7). In addition, hierarchical structure of blocks of the web page is represented in block structure panel (9).

Figure 2 illustrates navigation between blocks. In Figure 2a, user loaded the JSON file for the common scanpath. The blocks are in this common scanpaths are ”VB.1.2.1”, ”VB.1.2.2.1” and ”VB.1.3”. At the initial state of the application, VB.1.2.1 is displayed and other blocks are hidden. Since there are 3 blocks in the scanpath and current index is 1, next block is enabled, but previous block button is disabled. If the user clicks on the next block button, the block in the current index, VB.1.2.1, becomes hidden and current index is incremented to 2. Current block is VB.1.2.2.1 in the new state. Therefore, VB.1.2.2.1 becomes visible in the content panel as in Figure 2b. In this state, current index is 2 and there are three blocks in the common scanpath. This means that, both previous and next block buttons are enabled and user may use these two buttons. If user clicks on previous block button, the content panel returns to its initial state in Figure 2a. On the other hand, if the user clicks on the next block button, VB.1.2.2.1 is hidden and VB.1.3 is displayed.



(a) 1st Block in Common Scanpath (b) 2nd Block in Common Scanpath

Figure 2: eMINE - Home page in block based transcoding application

Using file selection panel, users can select only JSON files. If user selects a JSON file which is not well formed, or contains none of the block names, then the application returns an error message.

3.2 Role Based Transcoding

The transcoding application also includes a role based transcoding mode. In this mode, the roles of the visual elements in the web page are detected and grouped. Then, using the roles in the common role path, these visual elements are accessed directly or sequentially. Unlike block based transcoding, users do not load a JSON file in this mode. Instead, the JSON file is stored in a specific folder in the local file system. The transcoding application parses this JSON file in each execution of the transcoding algorithm. Since the knowledge base in the role detection algorithm may change with respect to the web design trends or the new web technologies, the role path may also change. Moreover, the purpose of transcoding may vary in different models of the application. Therefore, the JSON file for role path should be customizable.

Figure 3 represents the demo application of role based transcoding approach. It consists of an address bar (1), a content panel (2), several buttons for user interaction and a block summary panel. In order to navigate to a web page, user types the URL of the web page in the address bar. It loads the web page without transcoding it. In order to use the application

in role based transcoding mode, user must select this mode in the selection box (14). After user clicks on transcoding button (13), the application first segments the web page, detects the heuristic roles of the visual elements and then transcode the page, so that, only the first visual element which has the first role in the scanpath is displayed. User may change the current role to be displayed, by using next role button (5) and previous role button (3). Next role and previous role are selected with respect to the role scanpath. In each role change, current role is displayed in role label (4). There may be more than one visual element which belongs to a particular role. In order to access these visual elements, user may use next instance button (8) and previous instance button (6). Similarly, the index of current index among all indexes corresponding to the current role is displayed in instance index label (7). Moreover, user may navigate to the roles, which do not appear in the role path. In order to do so, user selects the role by using role summary select box (9). Also, user may jump to a specific instance of the current role, by using the instance summary select box (10). It is possible to specify some useful short cuts for users. For example, by using menu button (11), users can jump directly to the menu or menu item visual elements of the web page. Additional to the content and interaction buttons, hierarchical structure of visual elements of the web page is represented in block structure panel (12).

Figure 4 illustrates navigation between visual elements. In Figure 4a, user is in the initial state and application displays the first instance of *Article* role. There are two *Article*

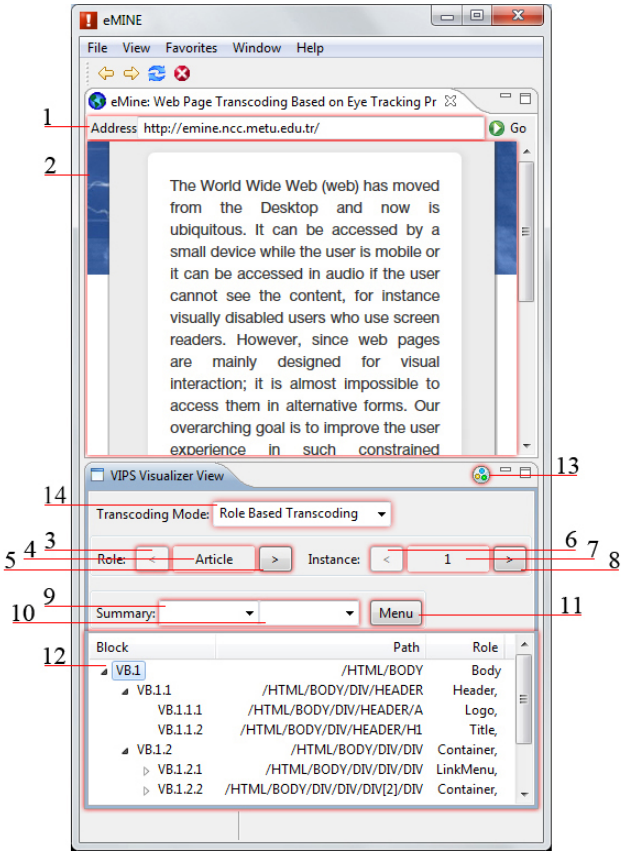
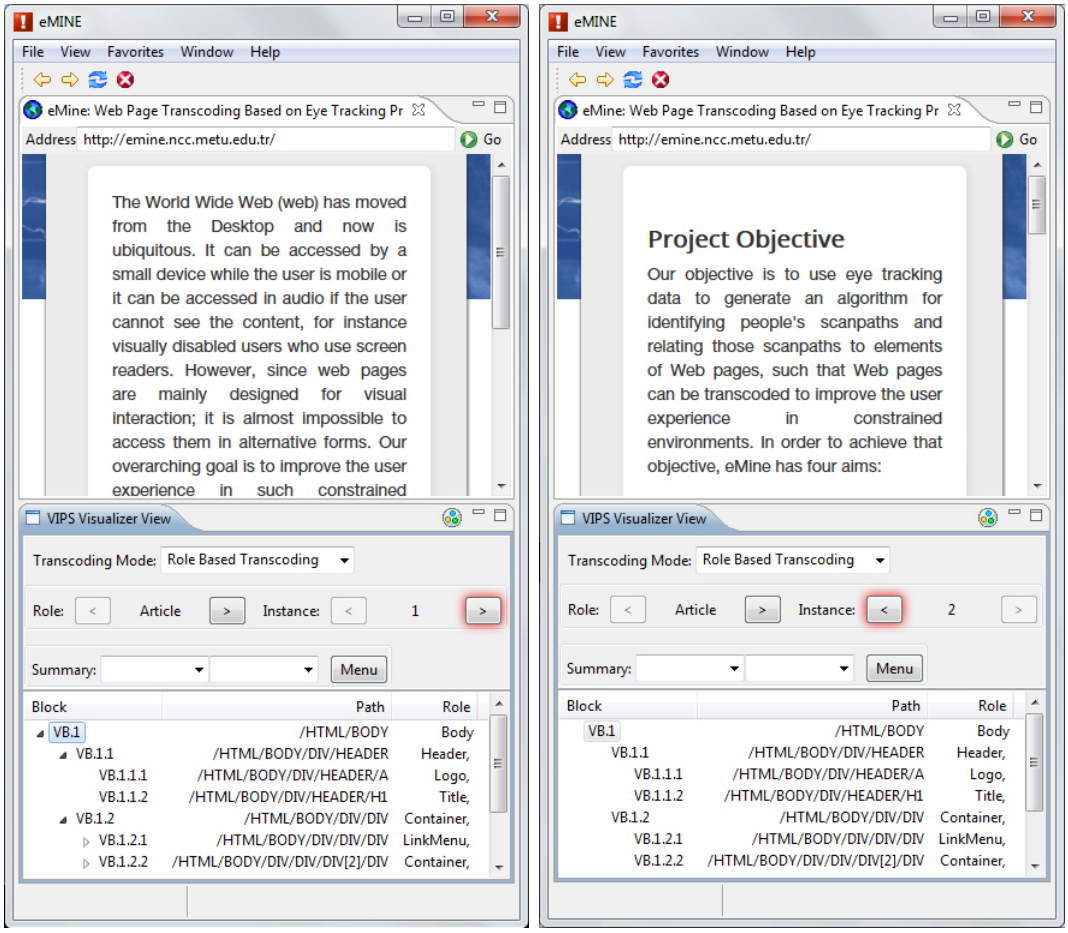


Figure 3: Role Based Transcoding Demo Application

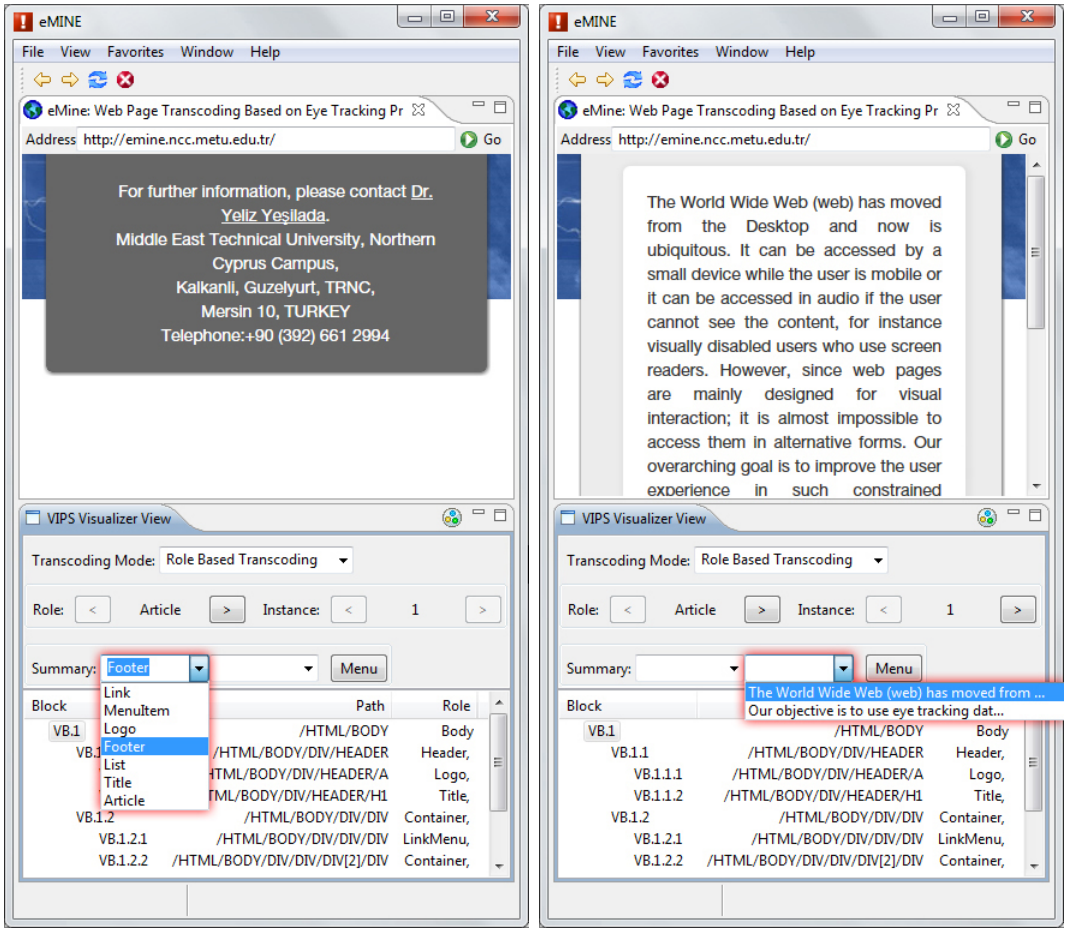


(a) 1st instance of article role (b) 2nd instance of article role

Figure 4: eMINE - Home page in role based transcoding demo application

instances in the page; therefore, user may jump to the next *Article* instance. However, since the user views the first instance, the previous instance button is disabled. Similarly, user may jump to the next role, however, the previous role button is disabled, since user is in the first role in the scanpath. After user clicks on the next instance button (highlighted in Figure 4a), current instance of *Article* role is removed and the next instance of *Article* role in the page is displayed. In other words, user jumps to the next instance. In this state, since there are only two instances of *Article* role and user is already in the second instance, next instance button is disabled. However, previous instance button is enabled, since there is an instance to jump. In this state, if the user clicks on the previous instance button (highlighted in Figure 4b), the second instance will be removed and the first instance which was displayed in the first state (Figure 4a), will appear in the screen again. It is important to note that, previous and next terms refer to the order of instances and should not be confused with user travel history. In conclusion, sequential access function of our transcoding method is implemented with a several buttons. These buttons enable user to move back and forward with respect to the role order in role scanpath and definition order of instances of the roles.

This transcoding method also provides direct access to the users. It is achieved by



(a) Role summary (b) Instance summary

Figure 5: eMINE - Home page in role based transcoding demo application

providing both a summary of the page structure in roles and instances, and direct links to specific parts of the page. Figure 5 represents the summary functionalities of the demo application. As highlighted in Figure 5a, the list of the roles which appear in the web page is provided as a select box. If user select a role in the list, for example *Footer*, the current visual element is removed and only the first instance of corresponding role is displayed. In this example, only one instance of *Footer* role appears in the page, therefore, next instance button is disabled. However, even if the user jumps to a role in the summary list, he/she can access all instances of the role. Similar to the role summary, the summary of instances of the current role is also accessible. Figure 5b represents the summary of instances of *Article* role. If the user selects one of the instances in the list, current instance is removed and only selected instance is displayed.

Another type of direct access is handled by providing specific buttons for some roles. In Figure 6, the button which enables users to access menu block is highlighted. After user clicks on this button, the first instance of *Menu* or *Menu Item* is displayed. It is possible to extend this functionality with respect to the needs of the users.

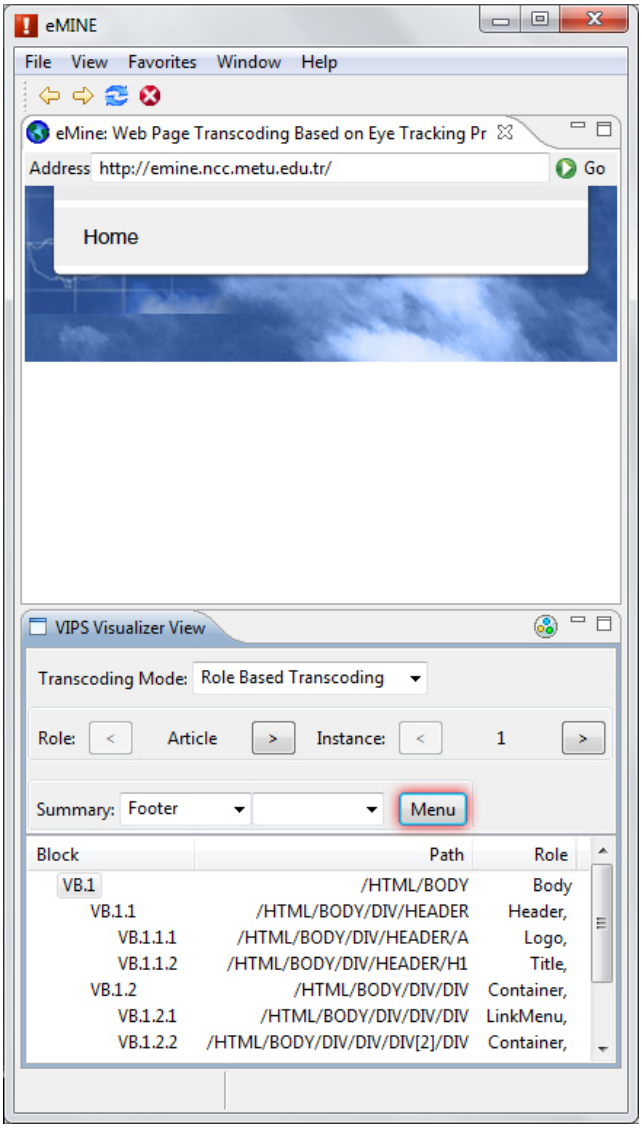


Figure 6: Direct access to menu

4 Evaluation

Our purpose in this study is to improve the browsing time and effort, which users spend to access to the initial visual element of their travel in a web page. Therefore, in order to measure the success of the proposed approach, we can compare the access time in both the original web page and the transcoded view for disabled people and mobile users. Moreover, it is possible to compare the technical feasibility of both original web page and transcoded web page by using simulation applications.

In our preliminary evaluation, we conducted a technical evaluation on block based transcoding approach. We used aDesigner to calculate the time required to access specific

Page	Complexity	Visual Element	Access Time
Apple	Low	1	9 seconds
		2	5 seconds
		3	10 seconds*
Babylon	Low	1	15 seconds*
		2	10 seconds
		3	17 seconds
		4	25 seconds
		5	36 seconds
Yahoo	Medium	1	17 seconds
		2	29 seconds
Godaddy	High	1	53 seconds
		2	51 seconds
		3	66 seconds
BBC	High	1	17 seconds
		1	69 seconds

Table 3: Access times to visual elements in common scanpaths

visual elements. aDesigner[2] is a disability simulator for designers to check accessibility of their content and applications. It helps to simulate the web environment in low-vision and blind people perspective, so that, designers can see how their web pages are accessed in case of inefficient conditions of voice browsers and screen readers. In blind people perspective, aDesigner calculates the time required to access a particular visual element from the top of the page. Moreover, we used visual comparison module of WebPageTest[3] to calculate the loading time of the page, CPU used to process the total request sent to the server and total file size downloaded for the page for original web page and transcoded version including visual elements in the common scanpaths. The pages were tested from Dulles, VA in IE 9 environment by using cable connection. Finally, we simulated the mobile environment by using Mobitest module of Akamai[4]. The tests were conducted in iPhone 5, iOS 6, AT&T device simulator. Each test was run three times from Cambridge, MA.

4.1 Access Time Comparison Results

We conducted evaluation for six pages including two pages from each complexity group. However, for home page of AVG, aDesigner returned 'Visualization Error'; therefore, we could not calculate the time required to access visual elements. Access times to visual elements in common scanpaths of the other pages are listed in Table 3. The results with * sign indicates that, these visual elements are graphical elements and do not provide any textual content or alternative text. Therefore, they are inaccessible for disabled people; therefore, aDesigner does not calculate an access time for such elements. In this study, we calculated an average access time based on the access times of previous and next visual elements.

With this study, we evaluated the time required to access certain visual elements in a web page. When we calculated the time required to access to the visual elements which users look first in low complexity pages, we saw that, time intervals are acceptable. In

[2] http://www.eclipse.org/actf/downloads/tools/aDesigner/
[3] http://www.webpagetest.org/
[4] http://mobitest.akamai.com//

Apple home page, the 1st visual element which is highlighted in Figure 7, is accessed in nine seconds. In Babylon home page, it took 15 seconds to access to the 1st visual element which is highlighted in Figure 8.

When the complexity of the page gets higher, time required to access to the preferred visual element also increases. In Yahoo home page, which has medium complexity, this time increases to 21 seconds. In Godaddy home page, which is among high complexity pages, time required to access to the visual element which is highlighted with red borders in Figure 10 is 53 seconds, which is not acceptable for web browsing. Although the 1st visual element is accessed in 17 seconds in BBC, which has also high complexity, it takes 69 seconds to access to the 2nd visual element in the common scanpath.

On the other hand, when we transcode the web pages by using our transcoding algorithm, the transcoding application displays the first visual element in the common scanpath. It reduces the time required to access to these visual elements.

4.2 Visual Comparison Results

The visual comparison tests aim to compare loading time of the web pages, the time in which CPU of the device is busy, total number of requests and total size of files downloaded for the web pages. In these tests, we first calculated these values by assuming that, client application does not cache previously downloaded CSS and JavaScript files. Then, we repeated the tests by assuming that client application only downloads these files for the first visual element, and does not download again for later visual elements.

Table 4 represents the comparison of loading time, usage, total number of requests and total downloaded data size for original web pages and transcoded versions. As can be seen from the table, transcoded pages send less number of requests and download less amount of data. Therefore, they load in a shorter time by using less amount of CPU. If the client application caches the previously downloaded resources, such as CSS and JavaScript files, efficiency of the transcoding approach increases. Appendix B, includes bar charts of visual comparison results for each page.

4.3 Mobile Simulation Results

In mobile environment simulation tests, average load time and average page size of the web pages were calculated for both original web pages and transcoded versions. Similar to the visual comparison tests, the results were calculated by assuming that, client application does not cache previously downloaded CSS and JavaScript files. Then, the tests were repeated by assuming that client application only downloads these files for the first visual element, and does not download again.

Table 5 represents the results of the mobile environment simulation tests. As can be seen from the table, load time and average page size decreases when the web page is transcoded. Moreover, if the client application caches previously downloaded resources, including CSS and JavaScript, load time decreases more. Similar to the load time, average file size decreases, when the web page is transcoded.

		With CSS and JavaScript				Without CSS and JavaScript			
Page	**Visual Element**	**Load Time (s)**	**CPU Busy Time (s)**	**Requests**	**File Size (KB)**	**Load Time (s)**	**CPU Busy Time (s)**	**Requests**	**File Size (KB)**
Apple	Original	7.148	1.216	39	445.603	-	-	-	-
	VE1	4.309	0.686	19	131.109	-	-	-	-
	VE2	4.071	1.014	18	131.414	2.131	0.374	2	8.106
	VE3	4.158	0.951	18	185.235	1.970	0.234	2	61.894
Babylon	Original	4.890	1.216	42	262.269	-	-	-	-
	VE1	3.433	0.218	9	142.107	-	-	-	-
	VE2	2.391	0.296	7	12.199	1.051	0.702	2	3.993
	VE3	1.970	0.218	7	14.596	0.652	0.109	1	1.214
	VE4	1.976	0.795	7	13.131	0.670	0.436	1	1.249
	VE5	2.001	0.717	7	13.158	0.653	0.358	1	1.195
AVG	Original	8.633	1.029	77	427.53	-	-	-	-
	VE1	4.282	1.263	23	232.277	-	-	-	-
	VE2	4.240	1.310	24	248.006	3.153	0.499	12	52.037
Yahoo	Original	2.728	1.918	50	474.294	-	-	-	-
	VE1	2.393	0.499	23	209.916	-	-	-	-
	VE2	1.098	0.733	16	105.822	1.102	0.265	16	70.697
Godaddy	Original	6.262	1.684	44	417.346	-	-	-	-
	VE1	3.898	0.358	19	179.289	-	-	-	-
	VE2	3.027	1.092	18	135.437	1.932	0.234	3	4.768
	VE3	3.046	0.655	18	138.089	1.329	0.795	3	5.699
BBC	Original	8.613	2.059	70	584.507	-	-	-	-
	VE1	5.742	0.686	25	187.706	-	-	-	-
	VE2	5.651	1.294	31	246.801	2.231	0.858	6	74.978

Table 4: Visual Comparison Tests Results

		With CSS and JavaScript		Without CSS and JavaScript	
Page	**Visual Element**	**Avr. Load Time**	**Avr. Page Size**	**Avr. Load Time**	**Avr. Page Size**
Apple	Original	4.85s	474.55KB	-	-
	VE1	2.97s	127.24kb	-	-
	VE2	3.11s	127.55kb	1.41s	7.88kb
	VE3	3.75s	180.11kb	1.73s	60.40kb
Babylon	Original	3.72s	259.29kb	-	-
	VE1	2.58s	138.18kb	-	-
	VE2	1.41s	11.36kb	0.99s	3.86kb
	VE3	1.58s	13.70kb	0.70s	1.17kb
	VE4	1.40s	12.27kb	0.69s	1.20kb
	VE5	1.48s	12.30kb	0.69s	1.15kb
AVG	Original	7.20s	401.41kb	-	-
	VE1	4.34s	222.22kb	-	-
	VE2	3.81s	237.95kb	2.03s	47.86kb
Yahoo	Original	2.90s	461.44kb	-	-
	VE1	2.53s	178.31kb	-	-
	VE2	1.48s	101.24kb	1.15s	69.30kb
Godaddy	Original	4.74s	404.79kb	-	-
	VE1	3.22s	171.95kb	-	-
	VE2	2.73s	114.45kb	1.66s	4.19kb
	VE3	2.53s	131.70kb	1.32s	5.09kb
BBC	Original	4.97s	574.68kb	-	-
	VE1	4.08s	170.13kb	-	-
	VE2	3.29s	234.83kb	1.66s	73.26kb

Table 5: Mobile Simulation Tests Results

5 Discussion

When we look at the popularity of the pages used in this study in Alexa[5], we see that these pages are globally very popular and have high ranks. yahoo.com is the 4th most popular web page, rank of apple.com is 37, rank of bbc.co.uk is 52 and rank of babylon.com is 120. In other words, these pages are accessed very frequently in a daily internet use. However, especially for high complexity pages, it may take unreasonable amount of time to access to a content, which a user may be interested most. On the other hand, our transcoding approach directs the user to these parts of the page. Therefore, we can say that, our transcoding algorithm has advantages over classical web traveling for disabled people and mobile web users. Moreover, transcoding approach improved the efficiency in both desktop and mobile platforms.

In this technical report, we presented the technical evaluation of block based transcoding which show the improvement in accessibility, load time, average file size and CPU usage. Further investigations can be conducted with mobile and disabled users to better demonstrate these improvements. However, we believe these preliminary technical evaluations are good in demonstrating the benefits of transcoding web pages.

In this study, we saw that, the success of the transcoding application highly depends on the success of the role detection algorithm. The evaluation results of the role detection algorithm showed that, the overall success of the algorithm is about 80% (Akpinar and Yesilada, 2013). This error rate may affect both the accuracy of general role path and the accuracy of the role assignments for a web page to transcode. The error rate in role path construction might be eliminated with manual validation of the role assignments. However, it is not possible for the web pages in transcoding process. Even though a visual element is falsely labeled, it is still accessible in the application, however, the error affects the time required to access it. Therefore, in order to improve the success of the transcoding approach, we need to improve the accuracy of the role detection algorithm.

6 Summary

This technical report presented implementation and preliminary evaluation details of our eye tracking based transcoding approach. First of all, we explained how we used the output of the eye tracking experiments to construct a common role path for all web pages. Then, we explained the implementation process of the demo application for web page transcoding. Finally, we conducted a technical evaluation on the proposed approach.

Our transcoding application has two modes: block based and role based transcoding. In block based transcoding, user loads a JSON file which specifies the scanpath of the web page. Application navigates user with respect to the path in the JSON file. In role based transcoding, there is a common role path which includes a set of roles indicating the sequential order of the navigation in the page. After segmenting the web page, the application navigates user based on the role order of the common role path.

Our technical evaluation shows that, block based transcoding improves the accessibility by reducing the time required to access a visual element which users are mostly interested in. Moreover, load time, CPU usage, total number of requests and total size of downloaded

[5] `www.alexa.com`, retrieved on 18.11.2013

data are reduced with our approach. For role based evaluation, further investigations are required.

A Pages

A.1 www.apple.com

Figure 7: Apple Home Page

A.2 www.babylon.com

Figure 8: Babylon Home Page

A.3 uk.yahoo.com

Figure 9: Yahoo Home Page

A.4 www.godaddy.com

Figure 10: Godaddy Home Page

A.5 www.bbc.co.uk

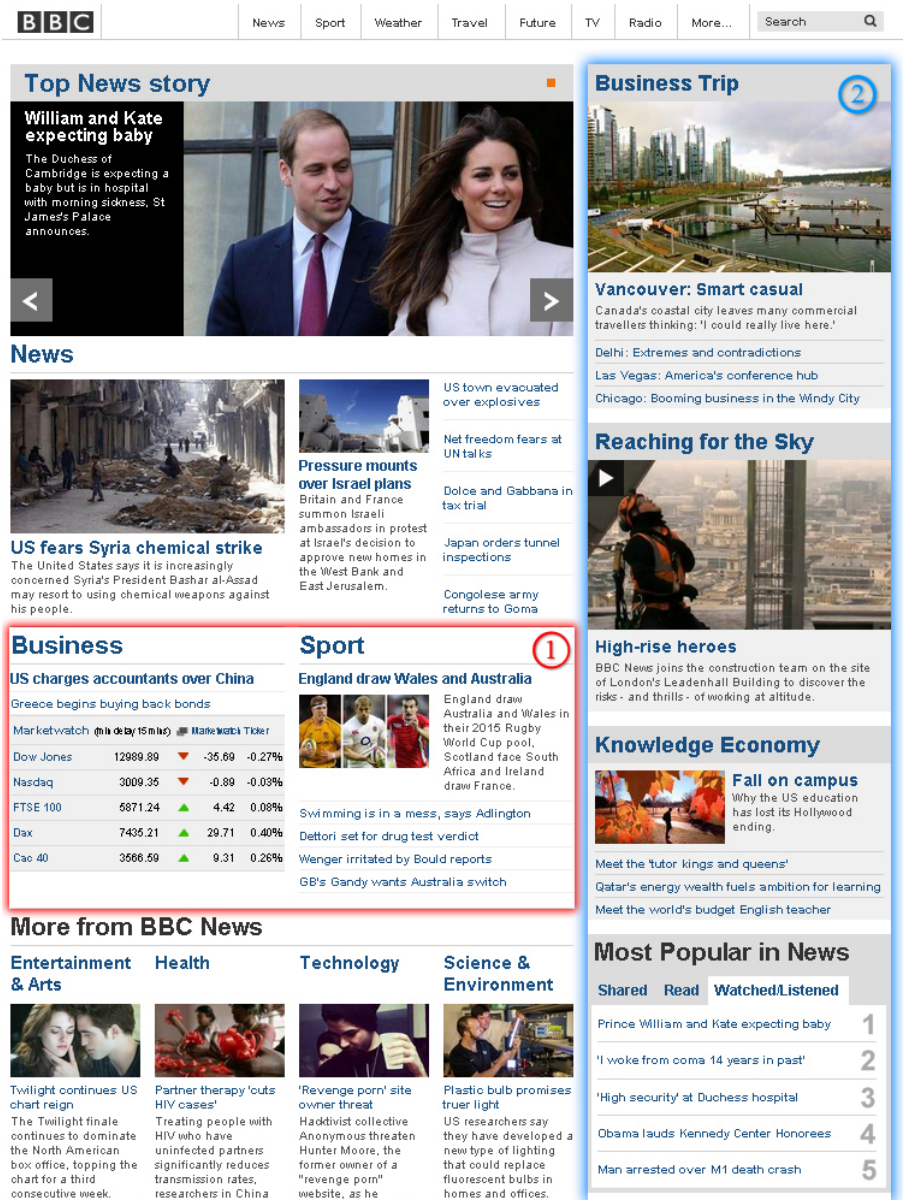


Figure 11: BBC Home Page

B Charts

B.1 www.apple.com

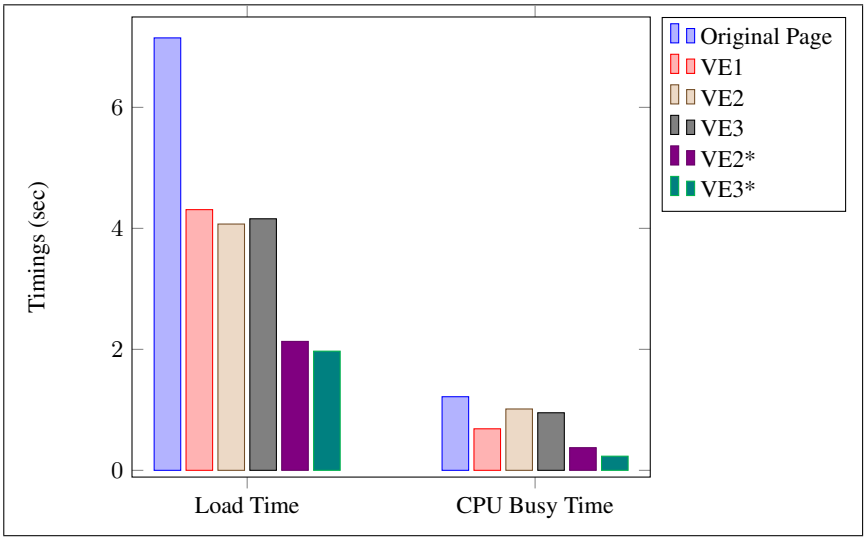


Chart 6: Load Time and CPU Busy Time for Apple Home Page

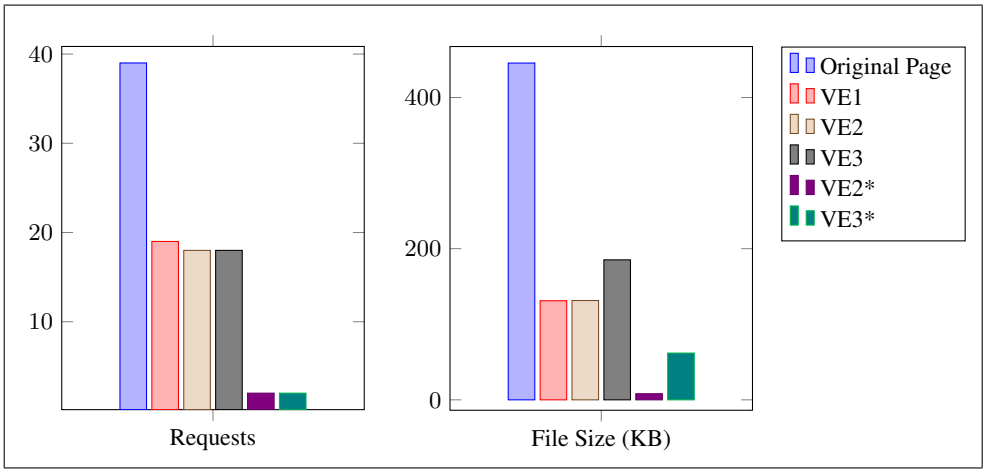


Chart 7: Number of Requests and Downloaded File Size for Apple Home Page

B.2 www.babylon.com

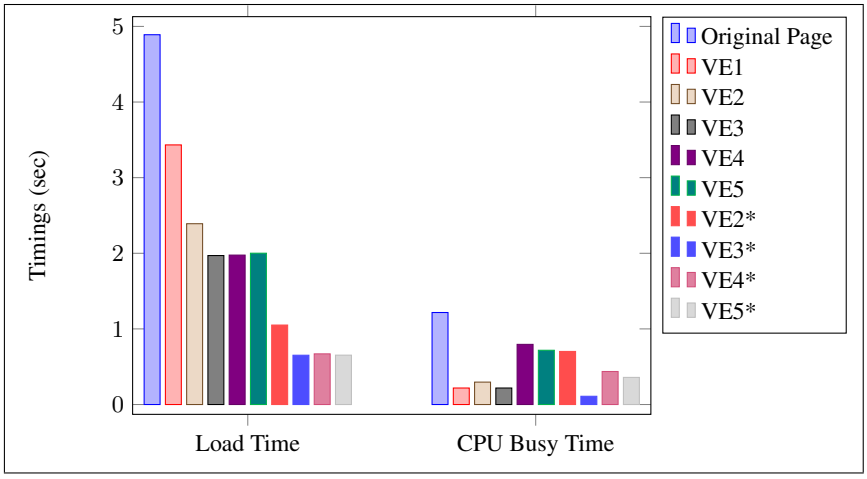


Chart 8: Load Time and CPU Busy Time for Babylon Home Page

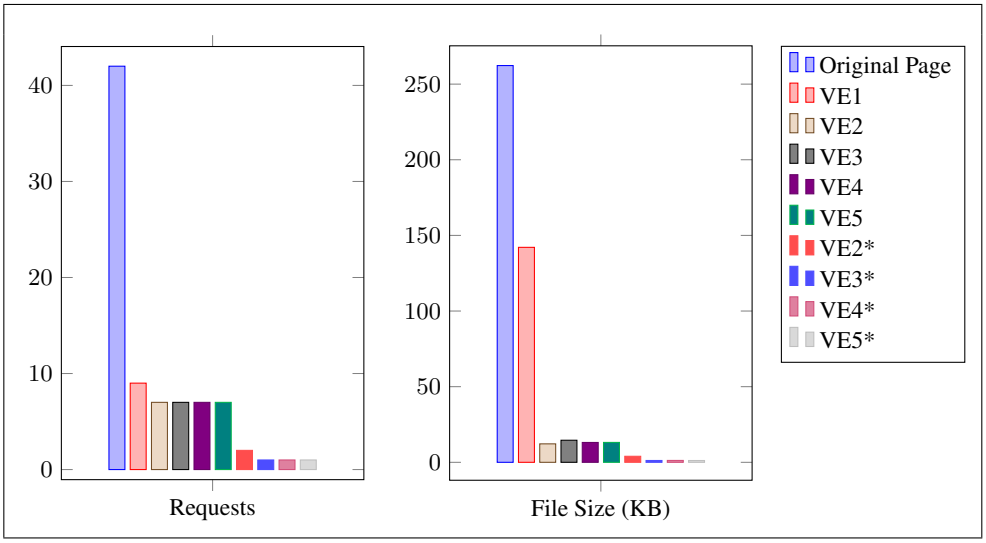


Chart 9: Number of Requests and Downloaded File Size for Babylon Home Page

B.3 www.avg.com

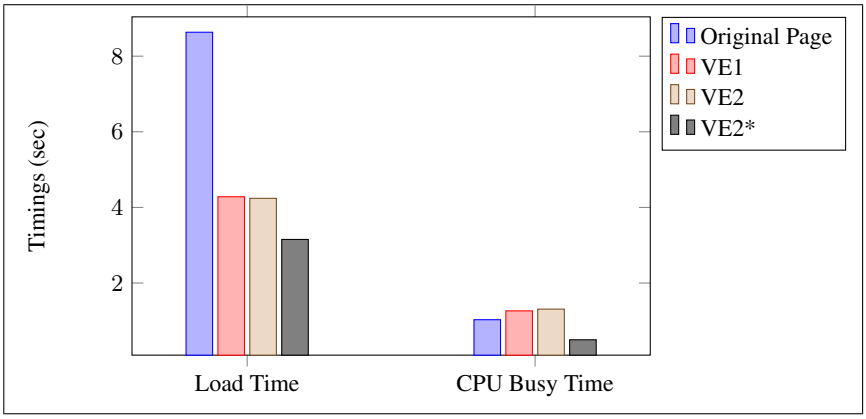


Chart 10: Load Time and CPU Busy Time for AVG Home Page

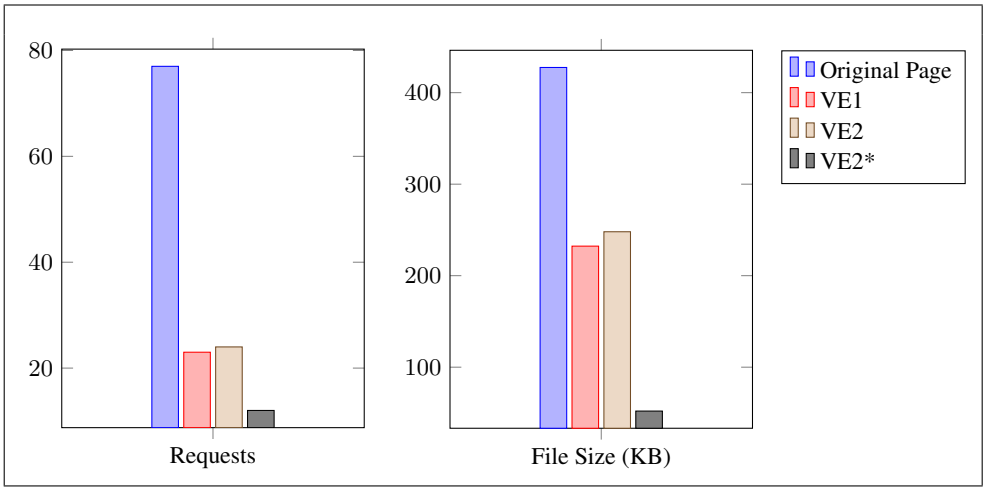


Chart 11: Number of Requests and Downloaded File Size for AVG Home Page

B.4 uk.yahoo.com

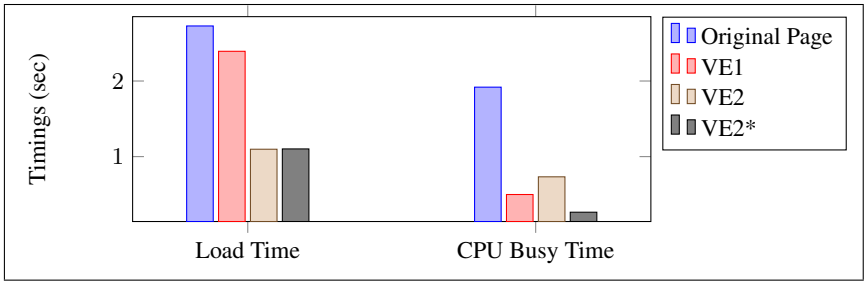


Chart 12: Load Time and CPU Busy Time for Yahoo UK Home Page

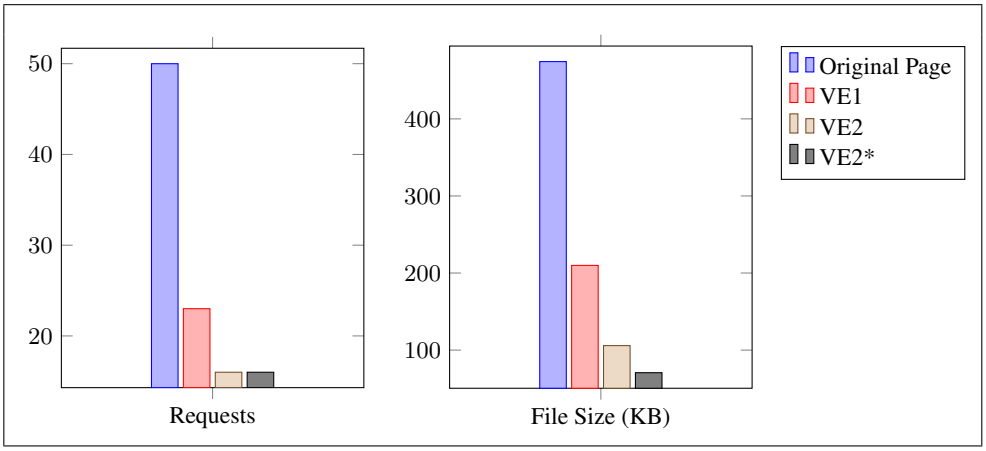


Chart 13: Number of Requests and Downloaded File Size for Yahoo UK Home Page

B.5 www.godaddy.com

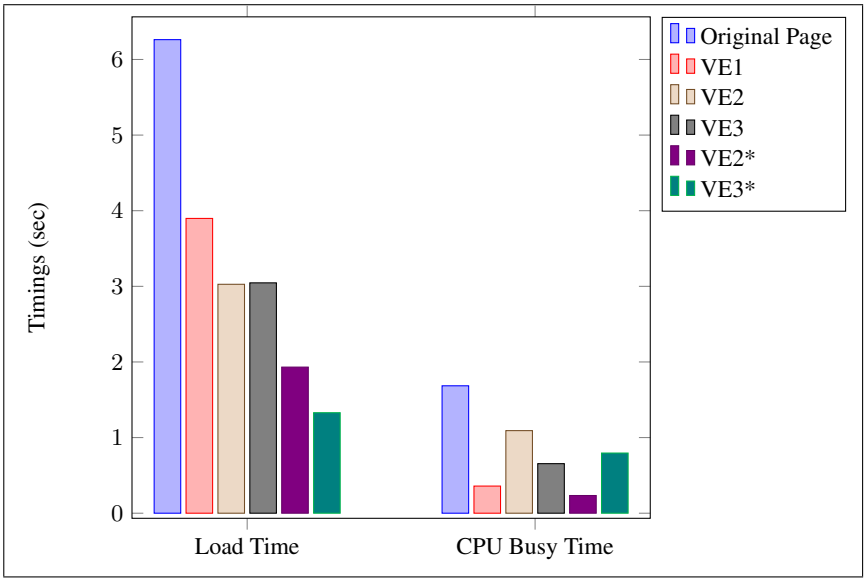


Chart 14: Load Time and CPU Busy Time for Godaddy Home Page

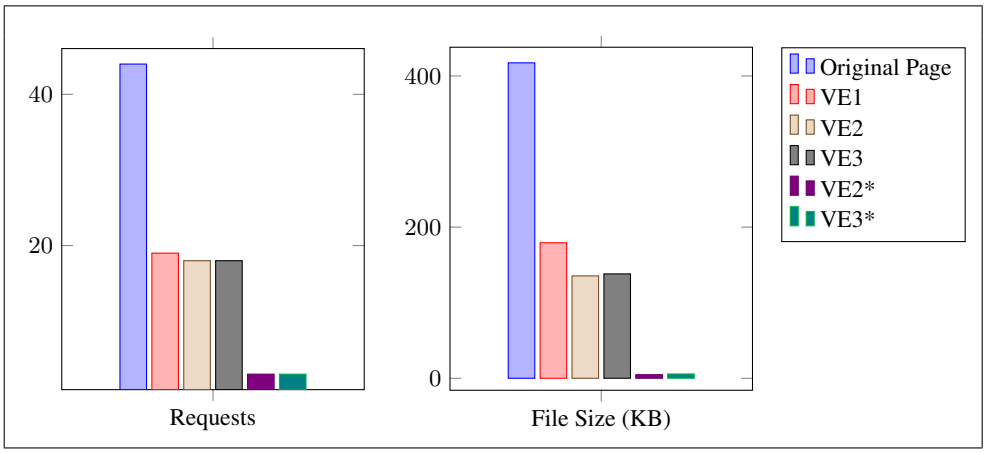


Chart 15: Number of Requests and Downloaded File Size for Godaddy Home Page

B.6 www.bbc.co.uk

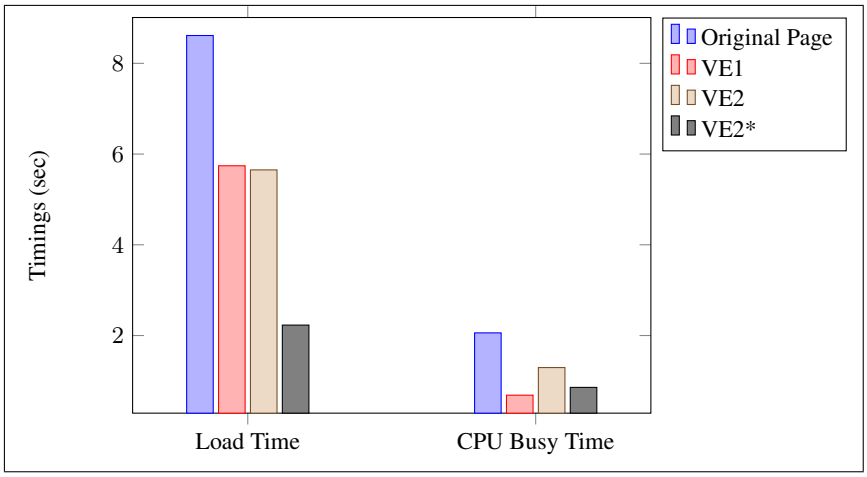


Chart 16: Load Time and CPU Busy Time for BBC Home Page

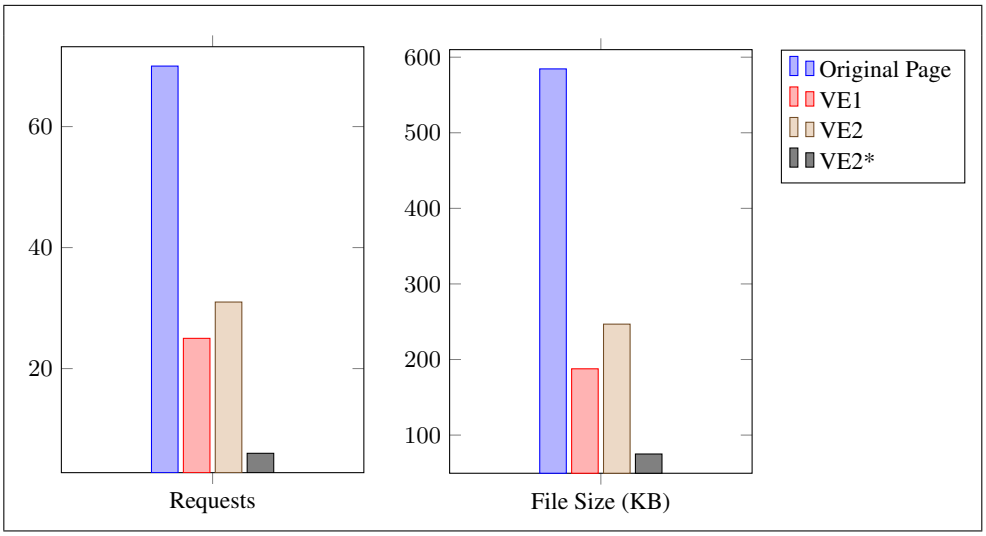


Chart 17: Number of Requests and Downloaded File Size for BBC Home Page

References

Ahmadi H, Kong J (2012) User-centric adaptation of web information for small screens. J Vis Lang Comput 23(1):13–28

Akpınar E, Yeşilada Y (2013a) Automatic discovery of visual elements of web pages. Technical report, Middle East Technical University Northern Cyprus Campus

Akpınar E, Yeşilada Y (2013b) Heuristic role detection of visual elements of web pages. In: Daniel F, Dolog P, Li Q (eds) Web Engineering, Lecture Notes in Computer Science, vol 7977, Springer Berlin Heidelberg, pp 123–131

Akpınar E, Yeşilada Y (2013c) Web page transcoding based on eye tracking. Technical report, Middle East Technical University Northern Cyprus Campus

Akpinar E, Yesilada Y (2013) Evaluation of automatic discovery of visual elements of web pages. Technical report, Middle East Technical University Northern Cyprus Campus

Borodin Y, Mahmud J, Ramakrishnan IV, Stent A (2007) The hearsay non-visual web browser. In: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A), ACM, New York, NY, USA, W4A '07, pp 128–129

Chen T, Yesilada Y, Harper S (2009) What input errors do you experience? typing and pointing errors of mobile web users. International Journal of Human-Computer Studies, Elsevier 68:138–157

Eraslan S, Yesilada Y, Harper S (2013a) Correlation between scanpaths and visual elements of web pages. Technical report, Middle East Technical University Northern Cyprus Campus and University of Manchester

Eraslan S, Yesilada Y, Harper S (2013b) emine scanpath analysis algorithm. Technical report, Middle East Technical University Northern Cyprus Campus and University of Manchester

Hattori G, Hoashi K, Matsumoto K, Sugaya F (2007) Robust web page segmentation for mobile terminal using content-distances and page layout information. In: WWW '07: Proceedings of the 16th international conference on World Wide Web, ACM Press, New York, NY, USA, pp 361–370

Mahmud J, Borodin Y, Das D, Ramakrishnan IV (2007a) Combating information overload in non-visual web access using context. In: Proceedings of the 12th international conference on Intelligent user interfaces, ACM, New York, NY, USA, IUI '07, pp 341–344

Mahmud JU, Borodin Y, Ramakrishnan IV (2007b) Csurf: a context-driven non-visual web-browser. In: Proceedings of the 16th international conference on World Wide Web, ACM, New York, NY, USA, WWW '07, pp 31–40

Milic-Frayling N, Sommerer R (2002) Smartview: Flexible viewing of web page contents. In: Poster Proceedings of the Eleventh International World Wide Web Conference

Xiao X, Luo Q, Hong D, Fu H (2005) Slicing*-tree based web page transformation for small displays. In: Proceedings of the 14th ACM international conference on Information and knowledge management, ACM, New York, NY, USA, CIKM '05, pp 303–304

Xie X, Miao G, Song R, Wen JR, Ma WY (2005) Efficient browsing of web search results on mobile devices based on block importance model. In: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, IEEE Computer Society, Washington, DC, USA, pp 17–26

Yesilada Y, Harper S, Eraslan S (2013) Experiential transcoding: an eyetracking approach. In: Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility, ACM, New York, NY, USA, W4A '13, pp 30:1–30:4

Yin X, Lee W (2004) Using link analysis to improve layout on mobile devices. In: Proceedings of the Thirteenth International World Wide Web Conference, pp 338–344

Zakas NC (2013) The evolution of web development for mobile devices. Commun ACM 56(4):42–48