CHALMERS

A Webpage Structure Processing Algorithm

Extending the Page Tailor Toolkit

Master of Science Thesis in Technical Communication

LARS ANDRÉN

IT University of Göteborg
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2007
Report No. 2007:2

A Webpage Structure Processing Algorithm

Extending the Page Tailor Toolkit

Lars Andrén

2007:2

A Webpage Structure Processing Algorithm - Extending the Page Tailor Toolkit
Lars Andrén

ISSN 1652-7674 2007:2
In cooperation with National Chiao Tung University, Hsinchu, Taiwan ROC

Department of Applied Information Technology
IT University of Göteborg
412 96 Göteborg

Department of Applied Information Technology
Göteborg 2007

A Webpage Structure Processing Algorithm - Extending the Page Tailor Toolkit

Lars Andrén

Department of Applied Information Technology
IT University of Göteborg
Chalmers University of Technology

Abstract

Research in user preference-based automatic processing on the web, web page content adaptation for a small screen and informative value of web pages have resulted in the design and implementation of an algorithm, called the Domain Heritage-algorithm. This algorithm extends the functionality of the Page Tailor toolkit; a program that is the result of C-Y Tsai's thesis "*Web Page Tailoring Tool for Mobile Devices*". The algorithm extending the toolkit enables automatic processing of web pages where preferences on which parts to be displayed have not been stored. The Domain Heritage-algorithm will not work unless at least one web page of the specific domain visited has been personalised previously. This extended toolkit has then been tested on ten subjects and a number of web sites. The test results were pretty much in accordance with the expectations, but the test subjects' experience in using the Page Tailor toolkit was found to be quite influential on the rate of successful running of the algorithm. Three major conclusions are made. The first one is that too much editing of the appearance of web page content can result in loss of informative value and successful totally automatic extraction of web page content needs semantic processing. Further, XPaths has been a good choice of data for the algorithm to process as the results of the Big O-analysis of the running time were acceptable, and that it was possible to implement the algorithm in the existing software. Finally, previous experience in usage of the Page Tailor toolkit, as well as more than one personalised web page is essential to the successful running of the Domain Heritage-algorithm.

Keywords

Informative value, Web page structures, Algorithms, XPath, DOM, Ruby, Domain heritage, Technical Communication

Table of contents

1. Introduction

Most homepages on the Internet are designed to be accessed and watched from a laptop or desktop computer, so unless the web master can maintain two versions of the page, it causes the design to be suitable for those large screen sizes. However, should you access a homepage using a small screen device like a mobile phone or PDA, you will experience the inconvenience in scrolling around to be able to see all of the content. Following the increase in popularity of using handheld devices to browse pages on the Internet, many new applications have recently been developed to solve these problems. One of them is the Page Tailor toolkit, developed and laid forth as a Master Thesis by Chi-Yang Tsai, "Web Page Tailoring Tool for Mobile Devices" (2006) at National Chiao Tung University[1] (NCTU) in Taiwan.

Although elegant and useful in its way, as Chi-Yang Tsai himself mentions in his thesis, algorithms could be developed to further increase the functionality of Page Tailor. These algorithms could be used to automatically detect and present the Page Tailor user with web pages that have not been previously visited. This thesis will shed light on the process of developing such an algorithm, show how it operates and finally present the results of its usage.

This section will provide the reader with an introduction of Taiwan, Hsinchu and NCTU, followed by a presentation of the background of the thesis itself, further explaining how Page Tailor works. Following this, there will be a presentation of the purpose, scope and limitations of this study.

1.1 Background

Presented to the Institute of Computer Science and Engineering at NCTU in 2006, the thesis "Web Page Tailoring Tool for Mobile Devices" presents the background, implementation of and a few test results of what was to become the Page Tailor toolkit. In the late summer of 2006, discussions with Professor Shen-Ming Yuan at NCTU, the advisor of the thesis mentioned above, and Magnus Axelsson at Chalmers, led forth to the birth of this thesis. Later on, Karin Wagner at Chalmers replaced Magnus Axelsson as supervisor. Purely application-wise, it is an extension of the Page Tailor toolkit.

The proposal for this thesis was created in September 2006, and presented in front of the Distributed Computing Systems-lab in the following month.

1.1.1 Taiwan, Hsinchu, NCTU and the start of this thesis

Taiwan Republic of China[2] is an independent economy off the eastern coast of China. It is today only recognised as a sovereign nation by some 25 countries. It was colonised by Japan for 50 years, but after the Second World War it was returned to mainland Chinese rule. However, the population rebelled against the Chinese rule (228-incident[3] among others) up until the Chinese nationalist Kuomintang[4] fled here in 1948. Since then it has been caught in the status quo and power struggle created by the communist take-over of the Chinese mainland. Thanks to support from the USA, Taiwan was China's representative in the UN until the 70's when mainland China was recognised as a sovereign state by USA.

It was later to become one of the first democracies in Eastern Asia and has seen a magnificent economic and industrial growth during the latest decades. However, the question of claiming independence or not and China's threat to invade if trying to become independent still shadows the politics and debates among the Kuomintang and other parties. It has been said that today Taiwan is the single most important link for global trade, because if an item is not made in Taiwan, a component of it most likely is. As an example, 76% of the world's laptops are made here as well as most of the world's semiconductors and bicycles. Although of different ethnical origin, the Taiwanese people are famed for their hard-working approach, friendliness and welcoming attitude towards foreigners. Western Taiwan has more arable land than the east and therefore higher population density. One of the many cities along the west coast is the "Wind City".

Hsinchu[5], known as the "Wind City", is a city located on the northwest coast of Taiwan, an hour away by car from the capital Taipei. It is roughly of the same size as Gothenburg and previously most famous for its rice noodles and glassworks. Where there up until the 80's stood a windswept bamboo forest, there is now a massive science park, which has made Hsinchu become the technological center of Taiwan. The science park hosts more than 600 companies, of which most are in the semi-conductor, computer, network and communications or software business. The more famous ones include TSMC[6], ASUS[7], ZyXel[8], BenQ[9] and ACER[10]. Hsinchu also hosts two of Taiwan's top three universities, NTHU[11] and NCTU.

Located in connection to the science park, National Chiao Tung University, or NCTU for short, is internationally among the top 50 in the world. Its most famous college is that of Nanotechnology and Electrical Engineering. NCTU recently celebrated its 111th birthday, and is one of Chalmers University of Technology's partner universities in Eastern Asia.

Starting from 2003, each year up to ten students respectively from Chalmers and NCTU are sent to the other university as exchange students for one or two terms. The author of this thesis took part in this bilateral exchange and spent the autumn of 2006 and spring of 2007 at NCTU. In October 2006, when searching for possibilities of doing a Master Thesis, the author met Professor Shen-Ming Yuan at the Computing Science department, and was by him introduced to three ongoing projects at the Distributed Computing Systems lab. After having attended further presentations of these projects, and discussing together with Professor Yuan, he recommended that an algorithmic extension of an application that handled adaptation of web pages to handheld devices would be a good thesis. This application was the work of C-Y Tsai (2006), one of Professor Yuan's students that had graduated half a year earlier. The work of this thesis was meant to develop and implement algorithms that were not used in the thesis presented by Tsai, but these algorithms were to be implemented into his Page Tailor toolkit in order to enhance the functionality of it.

1.1.2 The Page Tailor toolkit

The Page Tailor toolkit is aimed at filling the hole of a previously non-existent application. This application will allow a user using a desktop or laptop computer, to select what he or she wants to see in a web page, then using his or her handheld small screen device to access, through a proxy server, the same page, and being presented with his or her previously defined version of the page adapted for a small screen. In this thesis, mentioning the "Page Tailor toolkit" means the application the toolkit provides as a whole, it is however made up of three components.

The three components of the toolkit are: Page Tailor, Configuration Manager and Mobile Proxy. The first piece, the Page Tailor is javascript mobile code that can be downloaded and executed in a user's browser. The primary feature of the Page Tailor is that it does not need to be installed; it is triggered as a bookmark in the user's browser. It provides some visual manipulations helping the user to specify his or her preferences of a certain web page. The user personalises the web page by selecting which blocks of the page he or she wishes to see when accessing it through a mobile device. It also serves as the link to the backend database, where these configurations of web pages are stored. In short, what Page Tailor actually does is to add a new javascript-part to the current web page, and this specific part is later on what is returned to the user when accessing the web page through a mobile device. Chronologically, this is the first part of the toolkit that a user encounters; the second one is invisible unless specifically looked up.

This second component, the Configuration Manager, is the visual representation of the backend database. It is a web-based management interface which lists all the web pages that the user has personalised. Here, the user can create, edit and delete his or her preferences. Finally, when using his or her handheld device to look at the personalised web page, the third component comes into view.

This component is called Mobile Proxy. It serves as the link between the user's personalisation and the presentation of web pages on a handheld device. The Mobile Proxy monitors HTTP-requests made from the handheld device, and queries the Configuration Manager for the web page configurations. It filters the content and arranges the presentation before returning an HTTP-reply containing the user-defined web page. This is shown in Figure 1-1.

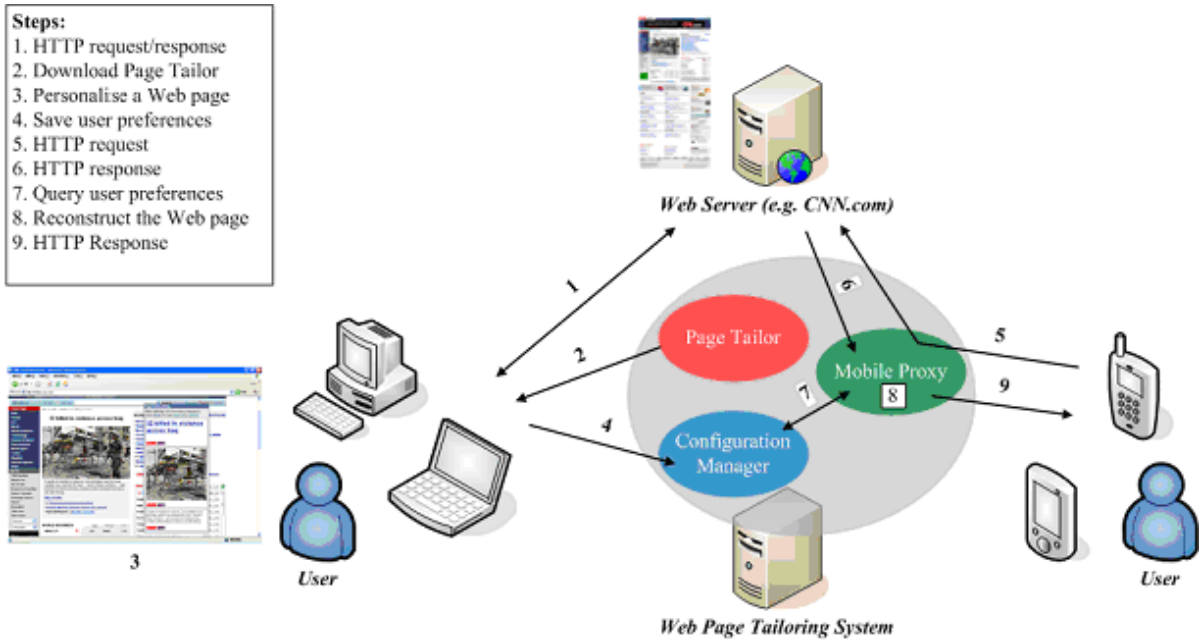

Figure 1-1 (C-Y Tsai, 2006, p 17) The chronological communication between the three components in the Page Tailor toolkit

1.2 Description of problem

Inherent in its functionality, using the Page Tailor toolkit requires each web page to be accessed through the proxy server is personalised and stored in the backend database. As Chi-Yang Tsai himself puts it, since many web pages are similar in structure, there could be a possibility to extend this functionality by algorithms that takes this into consideration;

"... perhaps the user preferences of a web page could be [sic] applied to others directly."

"... algorithms of detecting the structure of web pages could be developed ..."

(C-Y Tsai, 2006, p 46) Following this lead, this thesis sets out to research the possible development and usage of such algorithms.

1.3 Purpose and description of task

The purpose of this thesis is described by the following statements

- To examine how to maintain the informative value of web pages on a small screen in order to
- Study the possibility of developing algorithms for automatic detection of web page structures thus extending the usage of the Page Tailor toolkit, then
- To implement such an algorithm(s) into the Page Tailor toolkit, and evaluating it by
- Using a number of test subjects using the Page Tailor toolkit on web pages of their own preference to determine the usefulness of the implemented algorithm(s).

To denote the scope, it was decided early on that the algorithms developed were supposed to use "domain heritage" when applying a user's preferences to detect web page structures. This means that the algorithms would only function inside a top-domain where the user has already personalised at least one page. This denotation is further explained in section 3. In addition, such an algorithm should stick with the "tailoring concept", see section 2.1.4, in order not to lose the benefits of the Page Tailor's functionality. The stated four research questions are

- How can the informative value of a web page be maintained on a small screen?
- How can a Page Tailor user's web page preferences satisfactorily extend over several pages beneath the same domain?
- How is the algorithm using domain heritage to extend the functionality of the Page Tailor toolkit operating?
- Is the extended Page Tailor toolkit working satisfactory?

1.4 Outline of the report

After this brief introduction, the reader is provided with the theoretical framework, which might be considered as an auxiliary expansion to this report. It has two parts, the first one serving as an

explanatory reference for web technology and programming languages related to the thesis, the second part contains some related research.

The following section describes the research methods used to devise the algorithm and the web page information base that serves as its foundation. This section is followed by a look at the results achieved through the work with the thesis, and in close relation to this there is a discussion of these results and some conclusions. Finally, there is a section that briefly describes possibilities of future work and some legal issues.

2. Theoretical framework

This section serves two purposes; the first one is to present important concepts, such as research related to this thesis, the second one is an explanatory reference concerning the technologies involved in the Page Tailor toolkit.

2.1 Important concepts

This chapter will present some major basic issues. At the end, an important concept that has worked as a guideline with the work of the Domain Heritage-algorithm this thesis aims at implementing is presented. Before this, research related to this thesis is briefly summarised in three subsections. The research related to this thesis is divided into the three major areas that the research questions relate to; informative value of web pages, automatic detection and processing of web page structures, and finally automatic processing based on user preferences.

2.1.1 Informative value of web pages

The first problem to tackle when using the constantly growing World Wide-Web is to find web pages that are of interest to you; that can provide you with the information you are looking for or the service you need. Lots of research on how to help the human user to find what he or she finds interesting in the ocean of information the Internet is has been done. Using artificial intelligence in so called “agents” were especially popular in the 90’s, one is presented in Pazzani et al (1996). As time went by, the development of powerful and accurate search engines such as Google has now filled this gap that the “agents” were not able to. However, even if one finds a page with the information that one desires, that does not necessarily mean that everything on that page is of value to you.

Anyone who has surfed the web will most likely have had the sometimes quite frustrating experience that large parts of a web page are not of any value to you. Most web sites display banners, advertisements and other items that generally speaking are not of actual informative interest to the reader. In addition, the content of the advertisements, banners and notices can widely differ from that of the rest of the web page. Moreover, many web pages display images, link bars, navigation frames, banners and various other non content-relevant blocks.

For any individual person, it is easy to distinguish what is of value and what is not. Simply, what you appreciate finding in a web page, read or investigate further, is of value to you. This opinion, however, most likely differs from person to person, from case to case. Therefore, successful fully

automated detection and extraction from a web page what is of value to an individual is, if possible, very hard to do. The algorithm of this thesis, however, will not be fully automated, as it uses some data of one or more other web pages provided by the user to extract content that is of value to him or her.

Informative value, just as "value", is a general definition without specific denominators. However, concerning content on the World Wide-Web, it can be defined through "entropy"; used to measure the value of information.

Using the term "entropy" in "Information theory[12]" usually means a value from 0 to 1 that determines how much of the information in, for instance a message, that can be excluded with the recipient still understanding all of the message. In short, high entropy (a value close to 1) in a piece of information means that few or no parts of it are redundant, low entropy (a value close to 0) means the opposite; most of it is redundant. An example is the message "She will go there the following morning", treating all vowels as redundant and removing them gives us "Sh wll g thr th fllwng mrnng", a message that, although shorter, to most still can be completely understood.

Lin, Ho (2002) measures entropy in blocks of information from a set of web pages of the same site, and then partitions these into either informative or redundant. Informative content blocks are distinguished parts of a page, redundant blocks are common parts. By using this method on, for example, a news site, banners, advertisements and similar items can automatically be extracted and separated from the actual news; which are deemed to contain higher level of informative value.

Faster algorithms for doing the same thing are presented by Debnath et al (2005). There is yet another measure to mechanically extract some informative value of a web page; processing its images. This can be done using a technology known as image analysis, an algorithm for this is presented by Fletcher, Kasturi (1988).

2.1.2 Automatic web page structure processing

There are plenty of Internet services and related software that handles large amounts of information extracted from the Internet. However, as new web technology and more advertisements are added to web pages, and as people use other means than the traditional PC to browse the Internet, this new area of research has appeared. Automatic processing of web page structures has many areas of usage; two of these are to facilitate searching on the web and "data mining", a technology used to search for hidden patterns in large amounts of data to predict future behaviour. Another area is the adaptation of web pages to small screen devices, of which three different approaches will be presented in this section. Firstly, an application that is not yet available to public usage has been developed at Microsoft[13] Research Centre in Peking, China.

A way to enable easy browsing on a handheld device is shown by Chen et al (2003). The web page is processed, which presses its content into a standard template adapted for watching on a small screen. These blocks are then coloured differently, and the user can by clicking on a block "zoom in" this particular block. The "zooming" is actually an automated processing of this specific part of the web page to fit a small screen. Figure 2-1 shows the funcionality of this method. The second approach is today used by two publicly available services.

Figure 2-1 Chen et al (2003), a block in the processed web page is "zoomed in"

Two handheld device web browsers available today use technology to adapt web pages to better fit a small screen. These are Opera Mini[14] (Figure 2-2) and NetFront[15], respectively using technologies called Opera Small Screen Rendering[16] and ACCESS Smart-Fit Rendering[17]. The functionality of these two is quite similar and therefore has the same results and problems. The effect of the processing these browsers perform is in short that they eliminate the need for horizontal scrolling; all of the content of the web page is left as before. There is a third service available today, although quite unknown to the public.

Figure 2-2 Browsing cnn.com through Opera Mini-simulator

This service uses a different approach to process the structures of web pages, and has very quietly been released by Google[18]. Since this service still lacks official statements, it is here referred to as it is in the "Web Page Tailoring Tool for Mobile Devices"-thesis; "Google Mobile Proxy"[19], shown in Figure 2-3. This proxy server provides more features compared to the Opera Mini and NetFront browsers. Among these features the two most important are pagination of the content and links collapsing, meaning that all links hidden under a single one are shown.

Figure 2-3 Google Mobile Proxy and its links collapsing of www.dn.se

Finally, to broaden the scope outside the focus of small screen adaptation, we will take a short look at a totally different method. Vieira et al (2006) presents a way to detect and remove web page templates using mappings of DOM-trees. The processing of DOM-trees is also used in the Mobile Proxy part of the Page Tailor toolkit in order to present the user with the blocks he or she has selected. This is quite far from the functionality this thesis aims at extending Page Tailor with, but points to the fact that by extracting information of a few web pages as sample, a great effect can be achieved when using a good algorithm. A similar effect is shown by Crescenzi et al (2003) when trying to detect structures of large web sites in order to transform HTML into the more structured form XML, using a small portion of the web site's pages as sample.

2.1.3 User preference based automatic processing

The area of automatic handling or processing of web content or services based on user preferences is very wide. In short, what it has in common is that it is separated from other automatic processing due to the fact that it operates on the premises that a human user has somehow specified the basic conditions. Research varies not only in level of how much human specified data that is provided from the start, but also in aim and focus.

Presented by Agarwal, Lamparter (2003) is a very interesting algorithm that aims at combining different web services that can achieve a certain goal. This algorithm models user preferences in a formal way and then automatically ranks combinations of web services. The algorithm itself is elegant, but due to the scope of its usage, it is very general and quite formalised. However, there is in particular an area of user preference based automatic processing that can be regarded as more closely related to what this thesis aims to design and implement.

There is a specific technology of this that might be easier to comprehend to the reader since he or she probably have encountered it in daily life already. This is the so-called "Recommendation technology", which is a very new area of research connected to the recent increase of popularity in e-commerce. "Recommendation technology" has many different levels, the more advanced techniques includes data mining; as mentioned earlier, used to search for hidden patterns in large amounts of data to predict future behaviour. Although "Recommendation technology" at a first glance has little to do with this thesis, it has been studied during the work of this thesis and is the closest related technology in actual use today. So, it is presented here in order to enlarge the reader's comprehension of the complexity of web technology that uses automatic selection based on a user's preferences.

2.1.3.1 Recommendation technology in e-commerce

Large online shopping sites like Amazon.com[20] and CdNow[21] as well as the online movie rental store Netflix[22] have for many years used recommendation technology as a service to their customers. As analyst Patti Evans of Jupiter Research[23], a marketing research firm said in an article at Leavitt Communications[24] currently fewer than 25 percent of online shoppers make unplanned purchases, a far smaller percentage than customers at traditional stores. All kinds of

vendors wish to increase the possibility of the customers to make unplanned purchases, and online vendors has to work outside the common frame of visual impressions and the physical shelves and structures of a store that customers are used to. Therefore, recommendation technology has surfaced as the online alternative of making customers make unplanned purchases.

In general, when a customer makes a purchase, or searches for one at the online store's web page, he or she will be presented with recommendations that might suit his or her taste. The accuracy and relevancy of the recommendations experienced by the customer recommendations vary greatly. These variations are the cause of the amount of information the recommendation algorithms can handle, the vendor's range of content, if the user is registered at the online store etc. However, more and more customers of Amazon.com are experiencing these recommendations as relevant. This is an effect brought on by Amazon.com using more powerful and advanced recommendation engines than other stores, one of the causes that have made their sales increase over the last years according to Hof (2005).

2.1.3.2 Recommendation engines

There are four specific types of recommendation engines used on the World Wide-Web today:

- *Implicit engines* provide recommendations based on analysis of multiple customers' activities while browsing a company's Web site. These engines basically tell users that customers who bought Product A subsequently purchased Product B.
- *Explicit engines* make recommendations based on customers entering words or phrases to describe the type of products they are searching for.
- *Content-based systems* recommend items similar to ones the customer has preferred in the past. These systems gather information about a customer's preferences via questionnaires or past purchasing histories stored in databases.
- *Collaborative-based engines* provide recommendations derived from the purchasing preferences of customers with similar interests or demographics, based on questionnaire responses or profiles generated from consumers' online activities.

Furthermore, data mining algorithms, such as the one presented by Guo, Muller (2005), analyse customers' shopping and purchasing history, answers to user-preference questionnaires and surveys, and profiles resulting from this information. Customer activity is then translated into product and preference relationships that could indicate what consumers might be interested in buying in the future. Although Recommendation engines and data mining at a first glance might appear quite distant from the Page Tailor extension this thesis aims to implement, there are some parallels that can be seen.

The first of these parallels is that they both make decisions based on some sort of previous data, or "knowledge". In the case of recommendation engines, they will make decisions on what items of a product list or database that is interesting, based on "knowledge" of what the particular customer is interested in. In the case of this thesis, the algorithm will choose blocks of content from a web page based on previously chosen blocks of content from another one.

The second one is that they both try to present the user with something relevant, something of informative or desirable value to the user. Recommendation engines will present, or "recommend" an item to a customer that is somehow relevant to his or her needs or desires. When the algorithm this thesis aims to implement finds that a block of content from a web page is non-existent or lacks informative value, it is supposed to find another block of content that is better.

2.1.4 The tailoring concept

In the previous subsections of this section and in the result section 4.1, the difficulties of automatic processing of web page content are described. However, the Domain Heritage-algorithm will not handle a web page from a domain where there are no previous specifications; therefore it is not totally automated as it operates on the basis of some stored data. Totally automated processing is possible, as shown by software like Opera Mini or Google Mobile Proxy, but they do not allow the user to "tailor" his or her idea of informative value of a web page. There is a simile that can describe this very well; buying clothes.

When shopping for clothes you have two alternatives, either ready-made clothing or tailored. Ready-made clothing all come in the same size, a generalisation of how human bodies are formed, having the effect that if your shape differs from this generalisation too much, the clothes will not fit very well. You have to compromise your personal preferences and fit in order to spend less money. On the other hand, going to a tailor shop, your measurements are taken and not only the first shirt or pants will be according to your specifications, but all clothes made by the tailor be sewn to fit only you and using the material and colour you have chosen.

The Page Tailor toolkit is the tailor shop, but without the algorithm extension of this thesis, you have to go back to the tailor for every piece of clothing you want to buy, i.e. personally specify your preferences of every web page. The extension of this thesis is the customer book of the tailor, where your measurements are written down for future orders, i.e. the application of your former specifications to new cases for your convenience. So, the extension algorithm has to find and make good usage of some information of the user's preferences that is applicable in, if not all, most cases; thereby minimising your visits to the tailor shop, i.e. personalise pages with Page Tailor in a browser.

2.2 Related web technology and languages

This section is provided so that the reader who is unfamiliar with the technology of the internet can understand the following sections of this thesis.

Since the expansion of Internet in the 90's many new technologies and programming languages specifically designed for the web have seen the light of day. Many of these are only known to the ones working with web development, to the everyday user of services on Internet they are invisible, running somewhere in the background enhancing the usability of World Wide-Web-services like web mail, online shopping and banking services.

In order to allow the reader to get to know a few of these concepts and abbreviations, the ones that are to any extent used to create, or enable functionality in, the Page Tailor toolkit are briefly

described here. All but the last one, “semantics”, is somehow related to the actual functionality of the Page Tailor toolkit with the domain heritage-algorithm.

2.2.1 Proxy server

This process provides a cache, small memory holding recently accessed data, of items available on other servers which are presumably slower or more expensive to access.

This term is used particularly for a World Wide-Web-server which accepts addresses to web pages with a special prefix. When it receives a request for such an address, a URL, it strips off the prefix and looks for the resulting URL in its local cache. In this case, the Mobile Proxy is exactly such a server (part of it running on the user’s computer and currently hosted at NCTU), querying the Configuration Manager for that URL, if found, returning the small part of the web page that contains the user’s specifications; the part that has been added by the Page Tailor java script.

2.2.2 HTML/XML

HTML is short for “Hypertext Markup Language”.“Hypertext” is a term coined in the 60’s for a collection of documents, or "nodes", containing cross-references or "links" which, with the aid of an interactive browser program, allow the reader to move easily from one document to another. So, in short, HTML is a language using hypertext to create documents for the World Wide-Web. XML is short for “eXtensible Markup Language”, and is a more recently defined markup language with the same origin as HTML; the difference being that XML puts tighter demands on the order and numbering of the different parts of the document. If need be, Page Tailor reshapes an HTML-document the user has accessed into XML form. This is necessary for the usage of XPath’s.

2.2.3 XPath

A language that describes a way to locate and process items in XML documents by using an addressing syntax based on a path through the document's logical structure or hierarchy. When a user of Page Tailor selects blocks of content from a web page and saves this personalisation, what is saved is actually a number of XPaths that “points” to the blocks in the web page.

2.2.4 HTTP-request

HTTP is short for “Hypertext Transfer Protocol”. It is a client-server protocol used on the World Wide-Web for the exchange of HTML documents. A HTTP-request is simply a request for a certain web page address, a URL, from a client, for example a web browser, to a server. HTTP-requests from the users handheld device are monitored by the Mobile Proxy, which then queries the Configuration Manager for inputs in the database.

2.2.5 Ruby

A purely object-oriented programming language, originating from Japan in the early 90’s. Its usage has grown during the latest years as more and more documentation is becoming available in English. Some parts of the Page Tailor toolkit are written in Ruby, inside the Ruby on Rails framework.

2.2.6 Ruby on Rails

An open source framework written in Ruby for developing database-backed web applications. The Page Tailor toolkit being precisely such an application, its main part is written using this framework.

2.2.7 Java

A high-level, object-oriented programming language developed by Sun Microsystems[25]. It is similar to C++, another very popular object-oriented programming language, but has been simplified to eliminate language features that cause common programming errors. Java is a general purpose programming language with a number of features that make the language well suited for use on the World-Wide Web, such as the fact that it is platform-independent. It was meant to replace C++, which it has not, but its popularity has grown in linearity with the expansion of the World-Wide Web. The Mobile Proxy part of the Page Tailor toolkit is written in Java.

2.2.8 Javascript

Formerly known as "LiveScript", it is Netscape's simple, cross-platform, World-Wide Web scripting language, a kind of language that is weakly typed therefore enabling it to be directly interpreted – executing other programs - instead of compiled into machine code. It is only very vaguely related to the Java. The part of the Page Tailor toolkit that is seen in the user's web browser when personalising a web page is a javascript.

2.2.9 AJAX

Short for "Asynchronous JavaScript and XML". This has become a hot topic in web design community due to the popularity of web applications such as "Gmail[26]" and "Google Maps[27]". It is actually nothing more than an approach to web interaction, focused on transmitting the smallest amount of information as possible in order to create the most responsive experience possible. In short, the difference between this approach and the classic web application model is that the http-requests and replies are made asynchronously to an "AJAX-engine" without stalling the user's interaction with his or her browser. The Page Tailor part of the Page Tailor toolkit uses an "AJAX-engine" in order to make the user have a very smooth and uninterrupted usage-experience.

2.2.10 SQL/MySQL

SQL, Structured Query Language, is an industry-standard language for creating, updating and, querying relational database management systems, i.e. what is to most people simply known as "databases".
MySQL is a multi-user SQL database management system which is supposed to have more than 10 million installations. MySQL is owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, which holds the copyright to most of the code.
SQL is used via MySQL to create the backend database of the Page Tailor toolkit; the list of web pages the user has personalised that is shown in the Configuration Manager. The installation and extension of the Page Tailor toolkit used in this thesis a uses software called Navicat[28] to manually load the database schema of the Page Tailor toolkit.

2.2.11 DOM

Short for Document Object Model. It is an interface which allows programs and scripts to access and update the content, structure and style of documents. In other words, it is a hierarchical representation of a document's structure. In the Page Tailor toolkit, XML- and HTML-DOM is used to reconstruct a web page according to the user's preferences when presenting it on a small screen device.

3. Research methods

This section describes the methods used in the work of this thesis to answer the four questions stated in section 1.3. Although the work was not in reality separated in time, there has been some overlap and the occasional start-over, it is here separated into three distinctive parts.

3.1 Web page informative value and structure

During the first period of autumn of 2006, it was necessary to do some theoretical research on the structure and informative value of web pages. On Professor Yuan's advice semantics was not to be involved in this thesis, so semantics was studied only for a very short time. Semantics is generally known as the "meaning of words", but in this context it is referred to as "semantic processing" of text performed by computers and their software. Semantic processing can be used to make conclusions about actual meaning of blocks of text, for instance if the text is an advertisement, a weather report or a news caption.
The motivation to exclude semantics in the work with this thesis was that Professor Yuan believed that including this would highly increase the level of complexity causing too much work for a single person. In addition, he claimed that useful results could be achieved without it, more on this in section 5.1.2.

This decision also led to the restriction that the algorithm was only supposed to process pages under the same domain, as without semantics, the task to automatically process blocks of any web page based on previous preferences would most likely not give satisfying results. The restriction is based on the studies performed during the work with this thesis that has given the impression that pages beneath the same domain usually have similar structure, therefore much facilitating their processing based on previous selections. This has caused the algorithms to mainly try to process the structure of web pages, not the specific informative content. However, the variations in web design make fool-proof generic conclusions on web page structures hard to make, and therefore the development of algorithms tried several different approaches.

3.2 Algorithms and programming

On the technical side, concurrently with the research on informative value and web page structure processing, the languages and source code of the Page Tailor toolkit was meticulously studied, and its functionality tested through alterations in the code as well as case studies. These studies were essential to the development of an algorithm that could be implemented into the existing software. The trial algorithms were at a beginning freely developed without any consideration of the actual Page Tailor software and its inherent possibilities and limitations. Through discussions, "pen and

paper"-testing, implementation in testing modules and half-way or full implementation into the Page Tailor toolkit, these algorithms were born, further developed or discarded. The final algorithm that was fully implemented and tested is from now on referred to as the Domain Heritage-algorithm.

3.3 User testing

In order to more formally test the functionality of what seemed to be the fastest and most useful algorithm, ten people were used as subjects and a test questionnaire was created, this questionnaire can be seen in Appendix 1.

The test subjects were all in the age of 20-25, used to browsing the Internet and all but two students of different disciplines at Fu Jen University, a university of applied arts in Taipei. The remaining two were students at NCTU. Eight of the subjects were Taiwanese, one from Palestine and one from Macau. Six of the test subjects were women, four were men.

In all cases, a Pocket PC[29]-simulator was used through a common desktop computer, in five of the test cases, a browser window was used to represent the PocketPC. The usage of a browser window was a limitation that was likely to affect the in the testing as the environment would not be as independent as the Page Tailor toolkit is supposed to be used results, more on this in 5.4.2. The desktop computer was either one located in the Chalmers Asia[30] office in the NCTU library, or one located in the Distributed Computing Systems lab of NCTU.

During the user testing, the test subjects got to pick two domains where they usually would visit more than one page. In addition, at least one of these domains was to be of the sort that would update its content on a daily basis. The user testing was then performed during three rounds.

During the first round of user testing the extended Page Tailor toolkit, each test subject saved one personalisation of a page beneath each domain, and then three pages (not the ones personalised) of the sites were accessed through the Mobile Proxy of the Page Tailor toolkit on two different days. In the second round, two more pages at each domain were personalised, and the same three pages as in first round were once more visited on two following days. Third round added two more pages that were personalised, and three previously not visited pages at each domain were visited during two following days. In all three rounds, the subjects also accessed the same pages they visited on the mobile device through a standard PC's browser in order to see the complete pages. The individual experience of the results the test subjects' had, i.e. if the content of the visited pages that was displayed on the small screen device was of value to the subject or not, were written down on the questionnaire.

4. Results

Due to restricted scope of the extension, the previous unfamiliarity of the Page Tailor toolkit and the small number of test subjects there might still be better and more general algorithms to be found. Even so, this study has resulted in a possible extension of the Page Tailor toolkit as described in the "Future work"-section of the "Web Page Tailoring Tool for Mobile

Devices"-thesis. The results are divided into four subheadings that correspond to the four stated research questions. All of the research questions have further subheadings in order to separate the different parts of the specific question and their results more clearly.

4.1 How can the informative value of a web page be maintained on a small screen?

Web pages are usually designed for screen sizes of 1024x768 and up. This means that frames, link bars and other graphical parts of the web page are adapted to specifically fit these proportions. To some extent, text is also formatted to fit inside certain areas of the web page. On a small screen it is often very hard to get a good overview of the page due to the constant scrolling in order to see the entire page. The focus of this research question is possible solutions to this problem; transforming or adapting the content to enable easier viewing on a small screen. The work with this question was some freely based research that could result in an enhancement of either the functionality of the Page Tailor toolkit or the algorithm this thesis aims at designing.

The first of this section's three subsections is about some test modifications made to the Page Tailor toolkit. The following two subsections deals with general issues of informative value that applies to large as well as small screens.

4.1.1 Text formatting

There are several measures of text formatting that can be applied to the contents of a web page when presenting it on a small screen. The ones that are of concern here are mostly aimed at saving physical space. Inherent in Page Tailor is some formatting of text blocks, such as the removal of the HTML-tag "
" which is a line break. In addition to this, these further measures were added and tested:

1. Removal of heading tags, i.e. tags that enlarge text size and thickness
2. Removal of text-size modifying tags
3. Removal of text-style modifying tags, for instance bold or italic style
4. Removal of font-specifying tags

Especially the first three do save physical space on the screen, but some informative value might be lost. For example, if a content block in a web page is made up of a headline, a caption in bold and a text block with one or two words in italic or bold, it will have been transformed into a single text block of coherent style. In this way, the actual text content is maintained, but some informative values are lost. It was decided to drop number 1 and 3, a compromise between the space saving level of formatting and the wish to maintain informative value. Applying the second and the fourth change to a text block, some typographical beautifying might be lost. However, when applying the first and third change, it becomes more difficult to interpret the writer's stressing of a word or a phrase, or separation of a title and following text. Thereby, the actual information in the text is not properly maintained, since some pragmatic values are lost. If all four are applied, more physical space is undoubtedly saved, which does adapt a web page better to a small screen, but with the possible loss of informative value.

4.1.2 Automatic extraction of parts with high informative value

An intuitive approach to adapt a web page for viewing on a small screen is to not display the entire page, but only display selected blocks of content with high informative value thus preserving the informative value of the web page. So, one can attempt to automatically retrieve these parts of high informative value, which basically means excluding parts that are somehow mechanically or semantically deemed being of less informative value. If one excludes image analysis, what is left is the text content of web pages. The text of a web page can be further divided into two parts; one is the code-tags (possibly HTML-, XML-, or Javascript-code) that determine the font, size, colour and positioning of the text; the other one is the actual text content itself. Extracting all the text content from within the code-tags from a web page is quite simple, and a trial algorithm to do this was designed in Java and tested on a few web pages with very good results.

What this algorithm simply does is to extract the complete document word by word and exclude the code-tags. Following this, to further narrow down the amount of text, one can calculate entropy-values of different text blocks and thereby exclude the ones with high levels of redundancy. However, this has not been done in this thesis, as the results of these procedures still leave you with the problem to determine what is of true informative value and what is not. Informative value is defined by every individual; although text blocks with high level of redundancy can be of no value to one person, it might still be interesting to another. Same goes for blocks of text with low redundancy levels. Therefore, replacing or following the entropy-analysis in this process, is semantics. However, semantics was not included in the work with this thesis, which is why the determination of the actual informative value of one or a set of web pages is left to the user. This is further discussed in section 5.2.

4.1.3 Physical positioning

There are at times evident clashes of what the designer of a web page wants to catch the user's attention, and what the user herself wants to see. This is especially so regarding commercial web sites, sites where the ultimate purpose of the web site's existence is making money; sources of income for the webmaster will be clearly displayed whereas other parts might have received lower priority in positioning and presentation. However, a certain level of convention has emerged in web design where there is usually a top frame, a main frame and a right- or left-positioned frame. The right- or left-positioned frames is by convention static over a larger set of web pages, whereas the content of the main frame changes from page to page. The main frame usually contains the highest informative value to the user, as it is often the frame that contains the largest part of information of that specific web page. It can be assumed that some automated extraction based on the positioning of content can be performed. Two of these design conventions are shown in Figure 4-1 below.

Figure 4-1 Observed web page design convention, the different web sites a) correspond to b), and c) to d).

A trial algorithm was made that extracted certain parts of web pages, for instance the right frame or the main frame. The problem is that although there are some design conventions, there are no specific naming conventions for different parts of web pages, which make it impossible to make general and correct conclusions about what part of the page that represents the main frame, right frame, link bar etc. This problem was the main reason why this automated position-based extraction was dropped from the Domain Heritage-algorithm.

Based on some empiric studies, the algorithm would simply assume that, for example, the main frame's XPath would include the character string "main-something" or "something-main", which works fine in many cases, but not all. It is simple to add more options to the character string matching into the algorithm, like "centre", "central" or "body", but that still does not cover all cases. During the trial runs when the algorithm had failed, it was seen that the XPath's of the main

frame were at times nameless or identified by a series of numbers. In addition to this, this position based extraction causes quite large parts of web page to be extracted, not taking into account the actual lower-level splitting of the contents, in effect causing the amount of the web page presented to be unnecessarily large.

4.2 How can a Page Tailor user's web page preferences satisfactorily extend over several pages beneath the same domain?

In this section information about the concept the Domain Heritage-algorithm is based on is further explained. The section has two subheadings, the first one describes the general concept of Page Tailor that the algorithm should extend. The second explains exactly what makes the algorithm able to automatically extend a user's preferences over several pages.

4.2.1 The common denominator; XPath

Studying the functionality and technology of the Page Tailor toolkit and web page technologies and languages in general, a certain common denominator of all web pages and the Page Tailor toolkit was found; the XPath (section 2.2.3 for more information). This is a query language for XML documents, which inherently includes information not only applicable to a specific content block, but to all other blocks at that level of the DOM-tree (described in section 2.2.11) and the parent nodes on levels above. The beauty of it all is that since the DOM-trees of different web pages share the same basic structure, this usage is extendable to all, or most, web pages of the same domain. So, any given point in a web page can be reached with an XPath, and the corresponding point in another web page, with the same basic structure, can be reached as well. More over, with a little extra manipulation, the other children of a parent node can be reached as well. However, in the environment of the Page Tailor toolkit, it became evident that XPath appliance is not fault safe.

During the informal testing, the result was in some cases not useful as the content blocks could not be displayed. A probable reason for this effect is that very unstructured HTML is not compatible with JTidy, a Java-library that is used by the Mobile Proxy in the Page Tailor toolkit, causing the XPaths to be invalid when applied though the Mobile Proxy. The motivation for this is that the blocks are shown in the Page Tailor when selecting them from the browser, but not when accessing them through the Mobile Proxy.

To summarise, the successful and proper appliance of the XPath, not only the corresponding blocks of different web pages under the same domain can be extracted, but with a little extra work, other parts in the same area of the web page can be extracted too. Using the XPath, it is possible to "walk" through the DOM-tree, starting from the top and going down, or the other way around. This is known as back- and forward tracking respectively.

4.2.1.1 Forward tracking

This can be thought of as climbing a tree from the ground, starting at the bottom and then finding your way up through the branches finally reaching the leaves.

When tracking forward, the root-node of the DOM-tree is the starting point, followed by then matching the "address" of the following nodes with that of the next part of the XPath. This is

recursively repeated until the whole XPath has been applied, or until a leaf-node of the DOM-tree is reached. This method was approximated to have similar results as back tracking, but it is expected to run slower, and therefore not used in the final version of the Domain Heritage-algorithm. Figure 4-2 below visualises this.

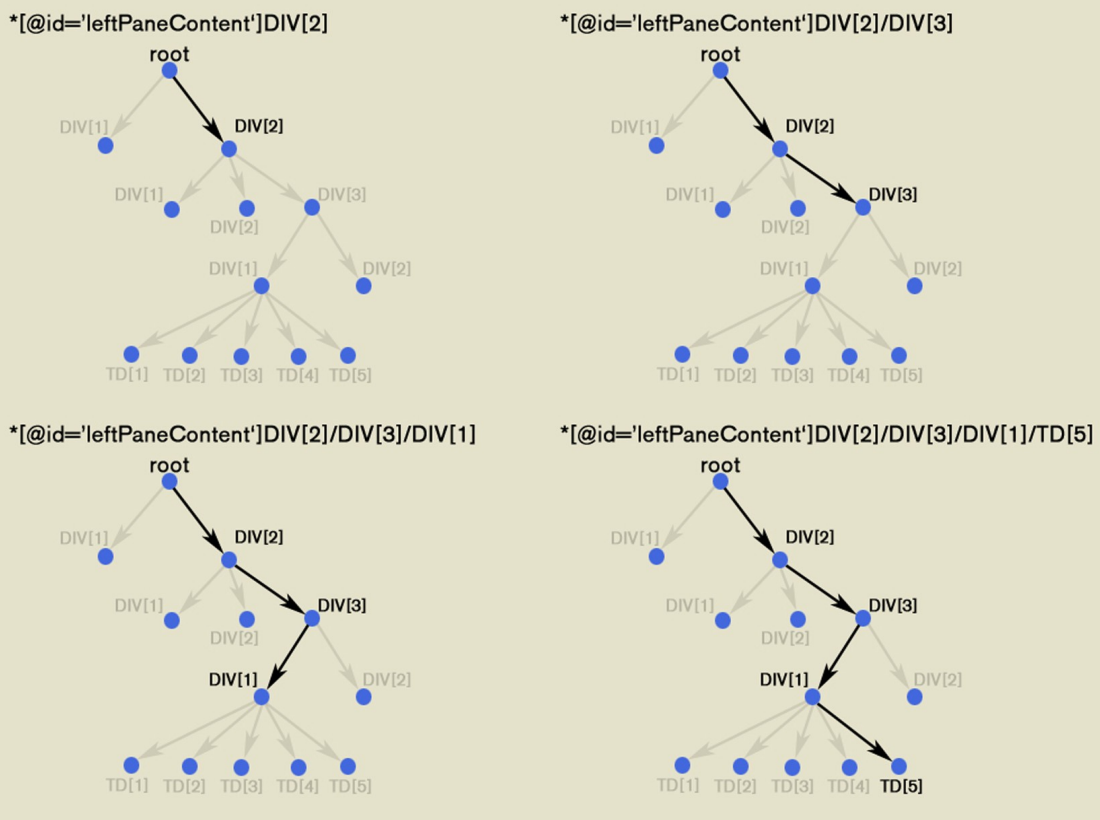

Figure 4-2 A visualisation of forward tracking through a DOM-tree using an XPath, the corresponding XPath is displayed at the top of each image

4.2.1.2 Back tracking

The Domain Heritage-algorithm uses back tracking, which can be likened to climbing a tree from the leaves, and if no leaves are found, moving on to the inner branch until either a leaf is found, or the ground is reached.

When tracking backward, the XPath is applied in its entirety. If a node has not been reached, the final part of the XPath is recursively changed, and if no node is found, the final part of the XPath is removed and the process starts over. What this actually does is to first search the DOM-tree for other children of a parent node, and if none are found, it will step up the next level of the DOM-tree. Figure 4-3 and 4-4 below visualises this.

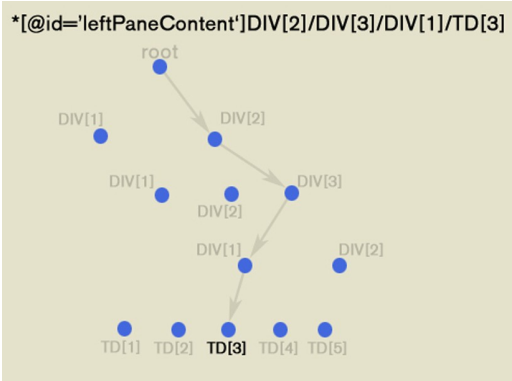

Figure 4-3 A visualisation of back tracking through a DOM-tree using an XPath, where a direct hit is found, the XPath is shown at the top of the image.

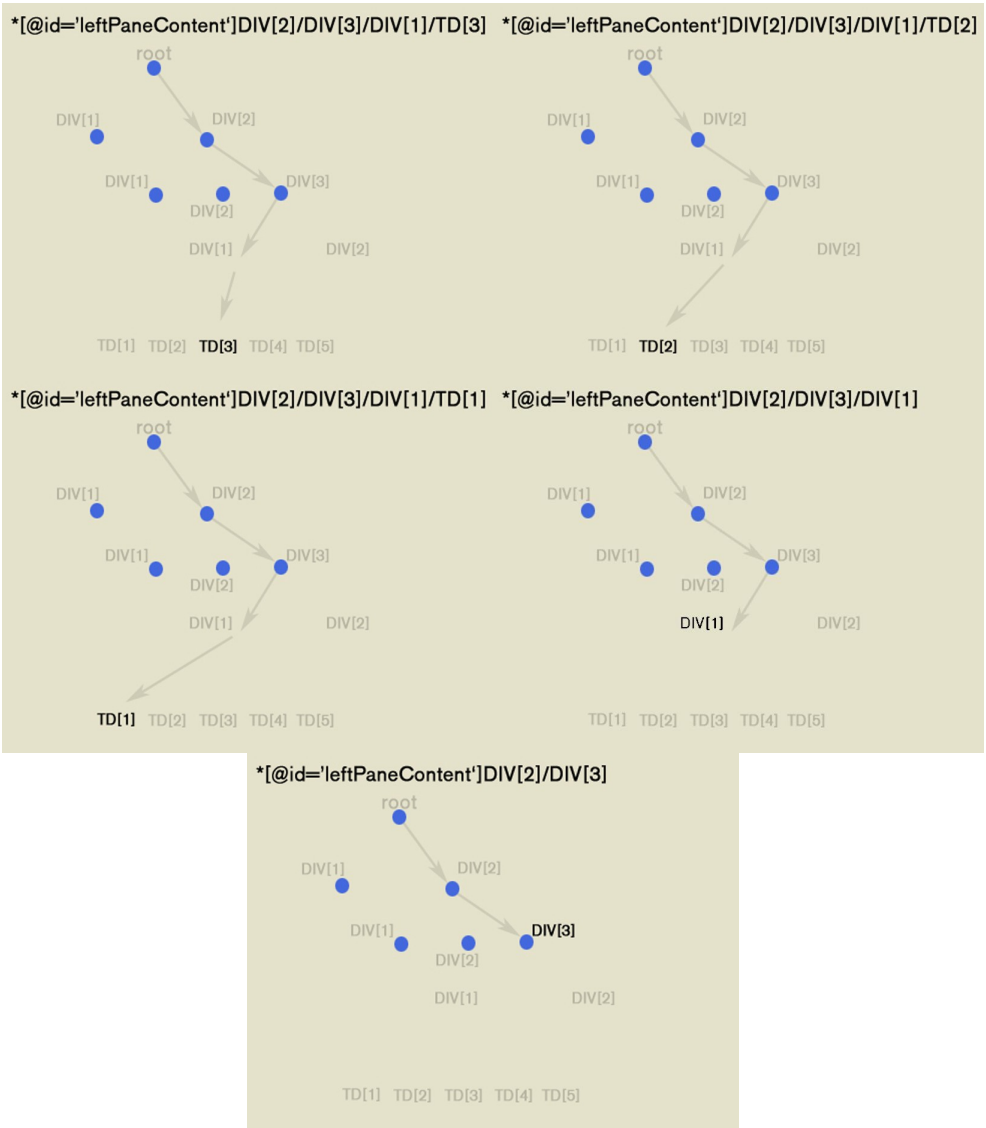

Figure 4-4 A visualisation of back tracking through a DOM-tree using an XPath that is corresponding to a non-existent content block. The XPath which is continuously altered and cut is shown at the top of each image.

4.3 How is the Domain Heritage-algorithm extending the functionality of the Page Tailor toolkit operating?

This section includes more technical information of how the Domain Heritage-algorithm actually operates. The first section is the algorithm in simplified pseudo-code, and the second one is a brief analysis of its running time.

4.3.1 Pseudo-code

Variables
XP: an XPath, in reality a just a character string
[XP]: a set of XPaths
Domain: a domain where at least one page has been personalised
[Domain]: a set of domains where at least one page has been personalised
URL: URL-address of a web page
Node: a node in a DOM-tree
newDOM: the resulting DOM-tree

Methods
void **XMLhttp-query**(string address, int id): a http-request
void **XMLhttp-reply**(string address, int id, int value): the reply of the http-request
string **toURL**(XP): transforms an XPath into a proper URL-address
integer **extractLeafNr**(XP): returns the leaf-number at the end of the XPath, returns 0 if no number is available
XP **decLeafNr**(XP): returns the same XPath but with decreased leaf-number
XP **cutXP**(XP): returns a shortened version of the XPath it takes as argument, the last item of the XPath is removed
boolean **checkXP**(XP): returns either true or false, false means that the XP has gone past reasonable limit of upwards parent climbing

Accessing a web page

```
//the function that handles the http-requests
integer function httpQueryReply(string URL) {

        send XMLhttp-query(address: XP, id <= random number);
        receive XMLhttp-reply(address: XP, id, value);
        return value;
}

//adding new node to the newly created DOM-tree
void function addNode(XP) {
```

```
        create Node newnode(XP);
        add newnode to newDOM;
}

//the running algorithm
void function algorithm(string URL){
        if URL in [Domain] {

        load [XP] of Domain;

                //start checking the XPaths, i.e. adding content to newDOM
                for every XP in [XP] {

                        //trying to find direct hit
                        int value = httpQueryReply(toURL(XP));

                        ////direct hit, add node to newDOM
                        if(value != 0){
                                addNode(XP);
                        }

                        //no direct hit
                        else{
                                position: LEAFCOUNT

                        //decreasing the leaf-number at the end of the XPath, i.e. looking for the
                        siblings at the same level in the DOM-tree
                                int count = extractLeafNr(XP);
                                until (count ==0){
                                        do
                                        XP = decLeafNr(XP);

                                        //checking with the new XP
                                        int value = httpQueryReply(toURL(XP));

                                        //success, add this new block
                                        if (value != 0) {
                                                addNode(XP);
                                                break;
                                        }

                                        //no success, decrease leaf-number and check again
                                        else {
                                        count--;
```

```
                    }
                }

                //when no more decrease is possible, cut the XP, i.e. step up a level
                in the DOM-tree
                XP = cutXP(XP);

                //if this XPath has been cut too far, break and proceed with next
                XPath in [XP]
                if(!checkXP(XP)) {
                break;
                }

                //if not, first check for direct hit
                else {
                        int value = httpQueryReply(toURL(XP));

                        //success, add this new block
                        if (value != 0) {
                                addNode(XP);
                                break;
                        }

        //no success, return back up to LEAFCOUNT to look for siblings
                        else {
                                goto LEAFCOUNT;
                        }
                } //closing else-clause
        } //closing for-loop

//URL not in [Domain]
else {
break;
}

} //end of function algorithm
```

4.3.2 Big O-analysis

Big O-notation[33] (also commonly known as O-analysis, Ordo-analysis) is a way to approximate asymptotic growth rate of mathematic functions. It is here used as a way to approximate the worst-case running time of an algorithm, worst-case meaning the possible maximum amount of time. The approximation only concerns the variable elements of an algorithm, and excludes constants.

The running time of the algorithm is a little roughly approximated to $O(n*(m))$, where n is the number of nodes in the DOM-tree and m the number of XPaths.

The calculation motivating this is as follows:

- The tree holds n-nodes, where n is an integer from 1 and up.
- If the number of XPaths is as many as the leaves of the tree, all leaves of the tree will be processed.
- For every XPath the following calculations will be done:
    - apply XPath to tree; time cost 1, i.e. constant
    - one httpQueryReply; time cost 1, i.e. constant (this is assuming that all http-requests take equally long time, which is naturally not true in reality)
    - in the worst case, each node has a number of children, which will cause the following calculations
        - leaf-number extraction and decreasing: time cost 1, i.e. constant
        - httpQueryReply; time cost 1, i.e. constant
        - a few other operations with time cost 1, i.e. constant
- If the XPath is shortened, i.e. the level above is accessed; time cost 1, i.e. constant

So, the worst-case represents a web page where there is a saved XPath to all available blocks, and where all the nodes in the DOM-tree are children to a single root node. In this case all leaves are first reached costing m, and then for every n, a number of constant time calculations are done, n*x, x being a constant. So, the total time cost is $O(n*x*m)$, which is equal to $O(n*m)$ when excluding constants.

4.4 Is the extended Page Tailor toolkit working satisfactory?

Before the user tests started, the subjects were given a brief instruction in how the Page Tailor toolkit works, and also the basic functionality of the Domain Heritage-algorithm extending it. They were all informed that it was the performance of the Domain Heritage-algorithm that was to be tested, and that their experience of its successfully presentation of unvisited web pages was to be evaluated through the questionnaire in Appendix 1. They were also told that the more content blocks they chose from a web page, the better the presentation of an unvisited one would become.

There was some overlap in the domains chosen by the test subjects. This was inevitable as the instruction on choosing web sites presented to the subjects was only was “to choose web sites that you visit often, preferably on a daily basis, and that these web sites’ content is updated often”. Also, the subjects were told that when choosing which pages to personalise, anyone would do as long as it was not the one that was going to be or had been visited in the previous round.

For every web page of every round in the user testing, each subject was set to answer the questions on the questionnaire, and if they felt they could not answer a question, they should leave it blank and comment why they could not answer. The first two questions of the questionnaire are related to evaluate the functionality of the algorithm itself, the following two are meant to evaluate the subjects’ experience of the changes in positioning and character of the content that is brought on by using the Page Tailor toolkit to watch the web pages. This gave the following results, the results are summarised and divided in the three rounds that have been described further in section 3.3. In some

cases, the calculation of percentages has been approximated upwards or downwards.

4.4.1 Round one

In the first round, only two test subject claimed that everything, or close to everything (grading 1 and 2) they wanted to see from a total of seven pages, or approximately 12% of the visited web pages in this round was there. Out of the rest, two stated that in six cases, equalling to 50% of the pages these subjects visited and 10% of the total pages visited, nothing they wanted to see in the pages was available in the mobile phone's version of the page.

Furthermore, in the remaining 47 cases, or 78%, something was missing (grading 3 and 4, meaning that "some", or "a lot" was missing) and that these parts were to 100% essential to their experience of the web pages (grading almost completely 4 and 5).

On the issue whether the content was differently presented, and if this had any influence on the subjects' understanding of the content, half of them did not answer the question as too little content was displayed to them. Of the remaining half, four claimed that their understanding was to 100% not affected, and one claimed that the vertical ordering of the content in two of the web pages she visited, 3% of the total for the round, was different which caused her understanding to be a little affected. The second day re-visiting was not experienced to have brought any changes to any one of the subjects.

4.4.2 Round two

As more pages at the two domains of each of the test subjects were personalised, there were no test subjects claiming that nothing of the content they wanted to see of the complete web pages was presented in the mobile phone's version of the page. Four subjects claimed that everything, or close to everything (grading 1 or 2) they wanted to see was presented in a total of 43 pages, equalling to almost 72% of the pages visited in this round, was presented in a very successful way.

However, the whole round included 60 pages to be visited, and in six cases, the result was graded a 4, meaning that out of the remaining 17 pages, or 28%, 10% were still presented quite poorly. In all of these six cases, or 10% of the total, the subjects claimed that what was missing was quite, or completely essential (grading 4 or 5) to their experience of the web page.

In eleven pages, or 18% of the total, subjects experienced that the content was differently presented in the mobile phone (grading ten 3's and one 4). This understanding was in five cases, graded to have some affect of the subjects understanding of the material (grading 2-4) which corresponds to two subjects' experience. These two subjects claimed that the vertical ordering and the completely vertical positioning as such affected their understanding of the content of the web page in a total of five cases, equalling to 8.5% of the pages visited in this round.

In difference to round one, the second day visits brought a decrease to the pages successfully presented to 40, or 67%, and an increase to the number of pages very poorly presented (grade 4) by two, a rise to 13%. In addition, the number of cases where the subjects claimed the missing part to be essential rose to seven, or almost 12% of the total.

4.4.3 Round three

In the final round, yet another two pages were personalised by the test subjects, bringing up the number of personalised web pages to five for each domain. Now, previously unvisited pages were visited, and the successfully presented pages numbered 54 (grading 1 or 2), equal to 90%. In the remaining six cases, equalling to 10% of the total was the presentation of the web page deemed to lack something the subject wanted to see (grading 3 and 4).
In four of these cases, equalling to some 7%, what was missing was deemed to be somewhat or completely essential (grading two 4's and two 5's).

In ten pages, or just below 17% of the total, subjects experienced that the content was differently presented in the mobile phone (grading seven 3's and three 4's). This understanding was in eight cases graded to have some affect of the subjects understanding of the material (grading 2-4). Also in this round two subjects' experience correspond to five of these cases where the subjects claimed that the vertical ordering and the completely vertical positioning as such affected their understanding of the content of the web page equalling to 8.5% of the pages visited in this round. The remaining two cases, or some 3.5%, are cases where no motivation was given as to why the understanding was affected. In this round the second day visits caused a minor rise in the number of successfully visited pages with one to 55, or close to 92%. The cases were the presented page lacked something decreased to five, or roughly 8%.

4.4.4 Summary

With 10 test subjects, each of them visiting two domains each and in each domain visiting a total of six different pages, the total times the algorithm was tested was 120 times. Adding the fact that each page was visited on two different days and that the first three pages were visited twice, the total amount is 300 times.

The percentage of successfully displayed web pages through the rounds was 12%, 72% and 90%. Accordingly, the percentage of poorly presented pages through the three rounds was 88%, 28% and 10%. Only in the first round were some pages deemed to be a complete failure, 12%.

On the issue if the missing part or parts were essential to the subject's experience of the web page, the percentage was at first 100%, then 10% and finally 7%.

As to the effects on the informative value, the subjects' understanding was affected in 3% and then 8.5% in round two and three. The second day visits only brought changes in round two and three. In round two the successfully presentation percentage dropped to 67%, poorly presented rose to 33% and very poorly presented rose to 13%.
In round three the percentage of successfully presented pages rose to 92%, and poorly presented dropped to 8%. A summary of the results is shown in the tables of Figure 4-5 below.

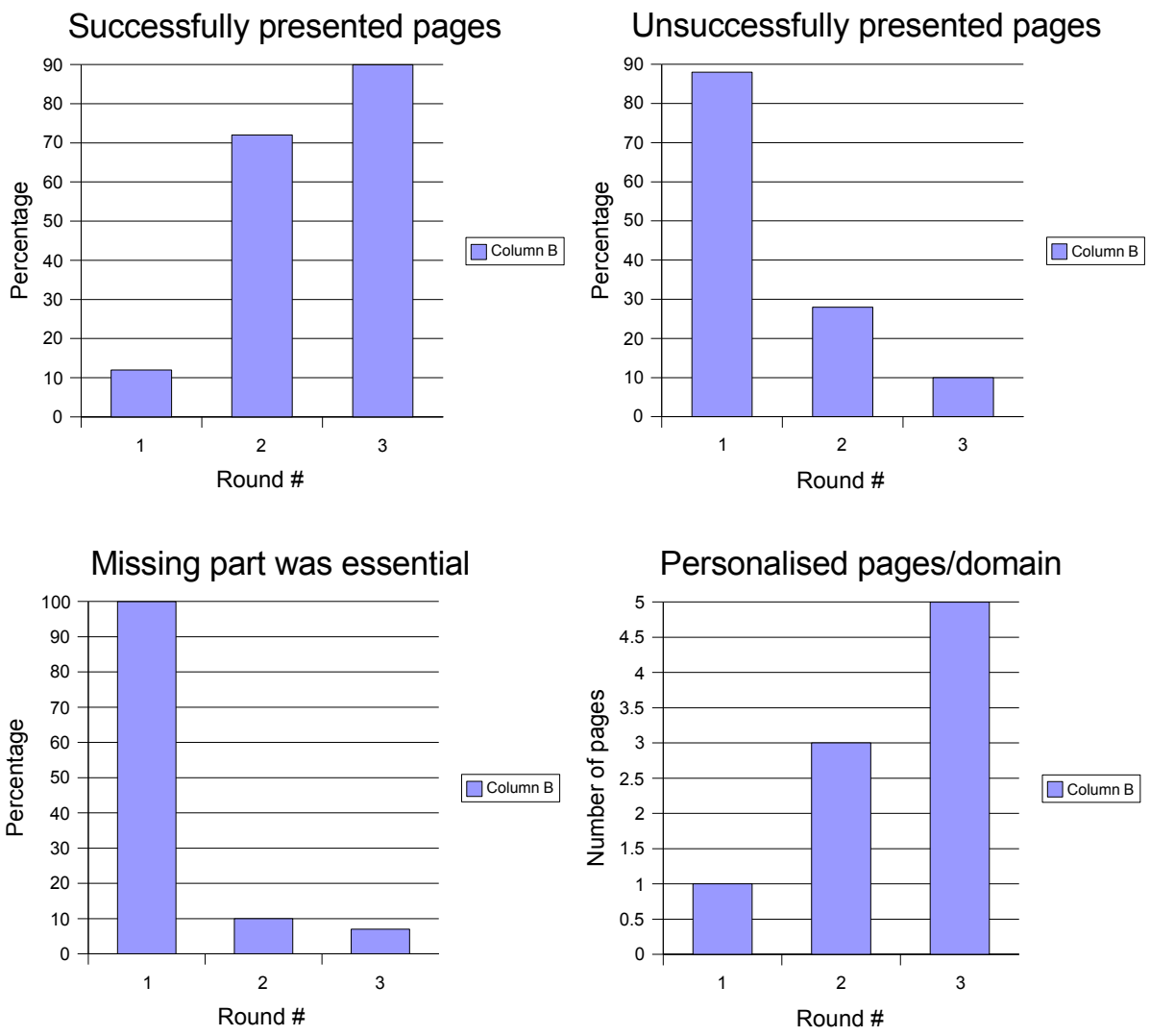

Figure 4-5 Tables representing the summary of the user tests

5. Discussion and conclusions

In this section the outcome of the results will be further discussed and interpreted. The sections are divided into three groups; the first section, 5.1 discusses layout and informative value. The second and third section, 5.2 and 5.3 handles the more technical aspects, assumptions and ideas. The final fourth section, 5.4, interprets and explains the test results. Based on the discussion in these three groups, some major conclusions have been formulated at the end of this section, and in the next section ideas and suggestions of future work is presented along with a few legal issues.

5.1 Layout and informative value; their relationship, compromise and automatic processing in the small screen environment

There are a number of web sites that exist in **two** versions, one for desktop/laptop screen viewing and one for small screens. Due to the compromise in screen space in these cases, in the mobile version of the web page both functionality and content have been reduced. In this way, the webmaster can choose what content that is most important to maintain the informative value of the web page and remove excessive graphics to adapt this content for the small screen. Many pages, however, only exist in one version, which is why there are programs attempting to handle the content and make compromises in this way automatically. The Page Tailor toolkit and the Domain Heritage-algorithm is an example of this.

In the pursuit of saving space on a small screen device, a measure that might be considered is leaving out all images from a web site. Sometimes, pictures will not be displayed properly on a small screen device, since they are too large, so leaving them out could save some space. They could also be shrunk, but this will in many cases make the picture worthless as it becomes too distorted. Leaving out images has simply not been handled during the work of this thesis, as images are a part of the informative value of a web page. In addition, a user of the Page Tailor toolkit might have selected images from a web page, and excluding them would then be obstructing a proper usage. A parallel to this can be drawn to the usage of web browsers and low transfer rates in the 90's. At this time, it was common for browsers to provide an option for the user not to display images, saving time but with great loss in the user's understanding of the content of the web page.

Any conclusions on the meaning of the physical positioning of web content are rather hard to make. When adapting a web page for viewing on a small screen device in Chen et al (2003), it was chosen to handle every part of a web page in order to minimise the loss of any relevant information. Although the Domain Heritage-algorithm tested in this thesis was trying to extract certain quite large parts of a web page, such as a right or main frame, it was discontinued as its level of success was quite low.

Apart from changing the layout, the other method of adapting web content to a small screen is to simply exclude parts of low informative value and only include parts with high informative value. Being able to do this maintains the informative value of a web page while freeing up a lot of space, thus allowing the usage of this space for preserving layout that might be contribute to the informative value of the page.

The measures to find high informative parts of web pages discussed here only applies to text, although image analysis might be of use too. Through image analysis, it is possible to extract text from images, such as Fletcher, Kasturi (1988) does, which can be either added to the text content of the web page, or used to determine the meaning or purpose of an image (if it is a photography, a link etc). Image analysis was however not considered to be used as a part of the functionality of the Domain Heritage-algorithm; all tested processing of web page contents was regarding text. Two text processing methods that have been considered and researched are entropy and semantics, although none of these two were incorporated into the Page Tailor toolkit, extended or not by the Domain Heritage-algorithm.

Applying entropy calculations to extracted text from web pages, such as done by Lin, Ho (2002) and Debnath et al (2005) some sort of separation between the text blocks of a webpage is achieved. Separation of entropy levels can be very useful when it comes to extracting the actual information content in the text blocks of web pages. However, these algorithms work on sets of thousands of web pages, and are generally aimed at reducing the size of data needed to be stored by search engines, not the adaptation to small screen viewing. In short, the problem of minimising the amount of data needed to be stored from a few thousand web pages and the one of extracting a few specific blocks from ten pages, are quite different. Semantic analysis might be considered as an alternative. Having semantic analysis to follow or replace the entropy calculations, as suggested in the future work section of Debnath et al (2005, page 8), can further narrow down the amount of text that has been deemed to be of higher informative value. Applying semantic analysis, in the case of this thesis, has its problems as well.

The problems of semantics include the considerably larger amount of original data needed to process new data. For instance, let us say that a user of the Page Tailor toolkit personalises a few web pages and have chosen some text blocks from his favourite news site, some that are pictures and headings from a photography site and yet some from the blogs of personal friends. When the user then tries to access a formerly unvisited sports page at the new site, the semantic analysis will not be strong enough to determine what is of informative value to the user, since the previous information stored by the user's choices are not only too little, but vastly different in content. Another problem is simply the low level of natural language processing[31] technology today, which is what semantic analysis is all about, as mentioned in Chen et al (2003), which make it hard to achieve satisfying results.

5.2 Linking the technical foundation and the target usage of the Domain Heritage-algorithm

In this section, the tailoring concept, the usage-basis linking the aim to maintain the elegant functionality of the Page Tailor toolkit with the design foundation of the Domain Heritage-algorithm of this thesis, is further discussed here. This is then followed by motivation for a major decisions made when designing the algorithm, and a minor decision that followed it.

5.2.1 The tailoring concept; defining and preserving your idea of informative value

What a person pays attention to in a web page is what that person thinks is of some value; it can be called informative value in the sense that it is not deemed redundant and worth spending time to read or watch. In the case of this thesis, the informative value of a web page is defined by the user; the informative value "input" so to speak.

This creates a problem and an advantage; the advantage is that each user gets to apply his or her personal definition of informative value; the problem occurs when this definition of high informative value of a web page is applied to another web page.

An example that can describe this problem is the instance when a user selects blocks from a part of

the structure of a web page that is used for different purposes in different pages under that domain. For instance, at the index page of a news website (i.e. the first page reached when accessing the website) some short summaries of the longer blocks of text in the main frame are presented in the left frame. The user selects these blocks, assuming that it will be the same way in the other pages. Surfing the website with his or her pocket pc, connecting through the extended Page Tailor toolkit's Mobile Proxy, the previously unvisited "travel & living" page at that very news site might display a number of pictures instead, not providing any useful information for the user, since he or she still does not anything about the actual text contents of the web page. In that particular page, the left frame was used as a row of thumbnails used to illustrate some of the places described in the main frame.

A way of avoiding this problem would be to make sure to select a few blocks from different frames of a web page, making sure that a larger proportion of the web page will be displayed. As the algorithm of this thesis operates, it is possible to select blocks from different frames at different pages of the site; which essentially shows how the algorithm works better with more user preference data to process. In accordance with the previous simile, the more measurements the tailor has of your body, the better suiting clothes he can make. This is not to be confused with a "learning algorithm[32]", as it is strictly mechanical and only applies to a single case, i.e. web pages beneath this domain.

5.2.2 Why use XPath and a possible alternative

When using a user's preferences as input to some kind of automated processing of other data, some kind of generally applicable properties have to be defined, as done by Agarwal, Lamparter (2003, page 2-3). These properties then form the foundation on which "unfamiliar" data is processed. In this case, that means finding a web technology that is generally applicable to most web pages and possible to integrate with the Page Tailor toolkit; a "common denominator".

Further, the final objective of finding a common denominator is to achieve something similar to the effects shown by Crescenzi et al (2003), where sample data is applicable to other cases than at where it was originally extracted from. Using positioning to automatic processing of the web page structures, which using XPath is, has the advantage that it is independent of which language the content of the web site is written in. This is a serious problem that semantic processing would encounter. Another factor why XPath is deemed to be the useful common denominator is the fact that it is already used by the Page Tailor toolkit; thereby an algorithm working with this technology is more likely to be successfully implemented into the existing software. The unlikely successful implementation of the other most promising alternative caused XPath to be chosen.

This alternative, which was considered for quite some time, was the mapping of DOM-trees; if one observes the structure of a DOM-tree and which parts that has been selected, it should be possible to apply the same selection to other DOM-trees, i.e. other web pages. This mapping should be possible to perform in line with the ones presented by Vieira et al (2006). The advantage of this is that although the specific XPath pointing to a node in the tree may look different in another tree, the corresponding part in the other tree can still be found as it is the location in the tree that matters, not the exact path to it. Another advantage is that if DOM-tree mappings are stored, the algorithm

has some possibility of finding blocks of content in web pages of domains where no page has been personalised.

The big drawback, however, is the same of semantics, quite large sets of data is needed to increase the possibility of successful appliance of an algorithm. In addition to this, an estimation of the possibility of implementing the algorithm into the existing software of the Page Tailor toolkit deemed that if possible to implement, it was very likely to be quite difficult since big changes in the software would be needed.

5.2.3 Trying to save time when tracking through the DOM-tree

One assumption that one can easily make about the XPaths chosen by the user is that they often point to a leaf in the DOM-tree, or close above one. This assumption is based on the fact that the beginning of the XPath contains HTML-tags, or XML-tags, that defines tables, separates headings from text etc. The leaves will often represent the final containers, such as images or blocks of text. However, this might not be evident to the first time-user of the Page Tailor toolkit, but after a selection of some blocks, the Page Tailor-javascript will display what has been chosen and one can easily learn the methodology of selecting useful blocks. For instance, instead of choosing the table-column including an article, its headings and footnotes, pictures and links to similar articles at the bottom, the user learns to select only the table-row in which the article body is placed. So, the question is how to fastest reach these leaves, or their parent nodes, that the user has selected.

Essentially, in this instance, forward and back tracking achieve the same results, but forward tracking is expected to take longer time to perform. For every node reached, extra time doing calculations to see if the node is a leaf-node or not, and if not, which one of its children that is the next step downwards in the tree are needed. This causes approximately $O(n)$ calculation cost for every XPath, where n is the number of nodes in the tree. Even though that is the worst case, the expected running time is not very different.

Although having the same Big O-approximation (even $O((n+x)$ where x represents the total number of non-existing sibling-nodes that might be checked)), the running time of back tracking is expected to be shorter. This is so since the possibility of a direct hit, i.e. when no extra climbing or calculations have to be made in the tree, is much higher. In addition, the checking if the XPath have been cut too far after the checking of sibling-nodes and upwards climbing in the tree causes the calculations to be less likely to go on for long.

In other words, whereas the forward tracking means climbing the tree always starting from the ground up all the way to its branches and leaves, backward tracking starts with looking at leaves and then often stops on a branch, before reaching the actual tree trunk.

5.3 Further analysing the pure technical aspects of the Domain Heritage-algorithm

Some issues of the Domain Heritage-algorithm's pseudo code and two alternatives to its running time analysis are further explained and discussed here.

5.3.1 A closer look at the pseudo code

The pseudo code showed in section 4.3.1 is mainly focused on providing the reader with an overview of how the algorithm works and displaying the run-time calculations, which motivates the time costs shown in section 4.3.2. It becomes quite high-level in the aspect that it does not clearly show usage of the actual different parts of the Page Tailor toolkit. For instance, the check for an URL-address in [Domain], and the retrieval of the XPaths comprises communication with the backend database. Furthermore, the function httpQueryReply() is a high-level pseudo-function that comprises several function calls to and inside the Mobile Proxy.

The "LEAFCOUNT"-part of the code is the part that handles the search for sibling nodes, which in a recursive way means to change the numerator of a tag. For instance, changing <DIV[5]> to <DIV[4]>, then to <DIV[3]> and so on.

Note that an increasing count is not attempted here. The decreasing count works on the natural assumption that if, for example, <DIV[3]> did not correspond to anything useful, <DIV[4]> or DIV[5] will not do that either. This assumption is proven to work correctly due to the way JTidy structures and properly names the parts of an "untidy", i.e. not properly structured web page.

A comment in the code on page 23 describes how it is checked if the XPath has been "cut too far". This usually means that too many levels in the DOM-tree have been passed, for example, originally, in a web page a part of frame is selected, the next level means the whole frame and the following level means all frames. Since the XPath might be cut from its end one, two or three times (usually), the number of content blocks included and presented to the user might be experienced as different from the user's preferences. In fact, the number of blocks remains the same, but their size might differ. For instance, instead of a single part of a frame, the whole frame is shown. Where to draw this limit is tricky since different web sites have quite different structures. In this implementation of the algorithm, the limit is drawn either at the end of the XPath or when a <TABLE> has been passed and a second one is reached. In either case, the XPath is deemed as not applicable and discarded. This limitation is strictly based on some empirical studies of web page structures, and is therefore not fool proof.

Finally, some operations in functions like extractLeafNr() and cutXP() are in reality not of constant time. An example is the time to calculate where to cut an XPath, which depends on the length of the XPath. However, on any computer of today, operations like these run in a few milliseconds, which is why they have been approximated to run in constant time.

5.3.2 Alternatives to the Big O-analysis

The worst-case described in section 4.3.2 is very unlikely to occur, which makes the resulting Big O-approximation a little exaggerated. Big O-notation being the "upper bound", it is the slowest possible execution of the algorithm. Another approximation is the Ω-notation[33] (Omega-notation) which is the "lower bound" on the running time, i.e. the fastest possible execution of the algorithm, or the Õ-notation[33] (soft O-notation) which is sometimes used in computing science.

As before, n is the number of nodes in the DOM-tree of a web page, and m is the number of XPaths

stored for the domain of that particular web site. In the case of the algorithm presented in 4.3.1, the Õ-notation is $\tilde{O}(n*m)$, since it simply ignores all logarithmic factors of the Big O-notation. The Ω-notation is approximately $\Omega(m)$, meaning all the XPaths are successfully directly found on the first attempt.

5.4 Interpreting the user tests and their results

The outcome of the user testing are discussed here, and also some clarifying of the test procedures are presented.

5.4.1 Purpose and expectations

The first round was supposed to test the algorithms functionality with a simple page personalised for each domain, accessing three pages at each domain. The success rate was expected to be quite low, maybe only 15-20 percent. This expectation was estimated on the fact that most web sites have at least two different basic structures, and a personalisation of one page can only be corresponding to one of these structures. An example of this is news sites, which often has a page where the items are structured by headlines and short caption, and another page structure where the whole articles are presented.

In the second round, the same pages as in round one were once more accessed, but this time the algorithm was expected to perform better, maybe 60% to 70% success rate, as it had a total of three personalised pages stored for each domain.
The only actual difference between the second and third round was that even more pages were personalised in the third round. In this round, a total of six pages were personalised, vastly increasing the amount of data the Domain Heritage-algorithm could work with, thus also increasing the probability of it successfully presenting web pages to the user. This round was expected to bring a higher rate of successfully presented pages than the other two rounds. The lowest expectation was that it would have the same success rate as the second round, although this would be somewhat of a failure as the theory that the Domain Heritage-algorithm works better with more data would have been rejected.

The decision to let the subjects visit the same pages in round one and two was supposed to show the difference in the rate of successfully presented pages starting with one and then three personalised pages at each domain. In this way, the subjects also got the chance to adapt the content of these web pages to their preferences, i.e. if it was experienced that some part was missing in all the three pages, this part could be chosen in the second round's personalising.

In the third round new pages were visited, a decision based on the assumption that it was likely that these pages would be successfully presented even though the subjects were not able to specify the content according to their preferences.

The decision to have the subjects access the web sites on two different days was supposed to test the Domain Heritage-algorithm's ability to function when the web sites had updated their content, and possibly structure too. The changes that were found in round two and three did however not show much effect, which could be interpreted as successful functionality. However, more likely

reasons for this is possibly that during all of the three test rounds, some of the web sites did not update their content and only a few of the web sites had some change to their structure.

5.4.2 Attempting to explain the results and its limitations

In the first round, the rate of successfully presented pages was very low, even lower than expected. There are possibly two major reasons why this was so; the first one is the subjects' lack of experience with using the Page Tailor toolkit and their understanding of it. The fact that a subject which said that the vertical ordering of the content affected her understanding points to the fact that she was not accustomed to using the Page Tailor toolkit. This is so as it is the user who determines which order the blocks should be arranged in, and not the program itself. Being the first time for the subjects to use the program may have caused them to select very few blocks, or select blocks with too little coverage, for instance a headline or a picture instead of a whole article. The second one is that the algorithm simply does not work very well with just one personalisation, especially if the structure between different pages of a domain is vastly different.

The success rate was vastly in the second round, and this is possibly because the reasons are the same as the ones that is believed to cause the low success rate in round one, but inverted. Firstly, the algorithm has a larger set of data to work with, and secondly the subjects are possibly more used to the Page Tailor toolkit and the workings of the algorithm extending it.

Although the second round had a higher rate of success, there were more cases where the subjects stated that their understanding of the content on the web page was affected by the diagonal positioning and ordering. This was also observed in round three, and points to an interesting issue. The web pages presented through the Mobile Proxy of the Page Tailor toolkit will look differently as the content is aligned totally diagonally, thereby losing its place in the complete web page. If the designer of the web page wishes to express something with the positioning of content blocks, images etc, this will not be transferred into the page presented in the mobile phone. Some subjected commented on the notion that the "feeling" of the page was different when displayed on the mobile phone. Although some changes are made in the appearance of the content (removal of font-size and font-modifying tags), the appearance of the web page in a larger aspect will be affected too.

This final round was pretty much as expected; even higher percentage of successfully presented web pages. Since there were so many pages personalised for each domain, the algorithm is very likely to find content to present even though if it does not get many "direct hits". The XPaths stored from the personalising of web pages have by this round accumulated to be quite a few, which is why the content presented in the mobile phone is quite a lot too; on a few occasions there were subjects that experienced it to be too much. This is caused by the problem that if the XPaths corresponding to the content blocks chosen in the personalisation process have little or no overlap with each other, there is a risk that when the Domain Heritage-algorithm runs on a web page, a large set of content blocks will be presented.

To summarise, the results of the user testing was pretty close to the expectations. However, the very low success rate in the first round was a little unexpected and considered in large part to have occurred due to the test subjects' inexperience of using the Page Tailor toolkit. The very high success rate in round two and three is thought to be caused by the subjects' enhanced ability to

select "useful" blocks of content during when personalising a web page and the fact that the algorithm has more data to process as more pages have been personalised. There were, however, some limitations of the testing environment that needs to be considered as well.

As stated in section 3.3, the user testing was not done on an actual mobile phone, but in a web browser that was adapted in size and shape to emulate the screen of a handheld device. This may have caused the some of the subjects to experience the web page differently as the viewing of the page is done on a "small screen" inside a big screen, thereby adding some distraction in the form of other things displayed on the big screen that might have distracted the subject.

In addition to this, in the environment where the tests were performed, there were most often other people present which on one hand might have distracted the test subject when personalising web pages. On the other hand, the actual viewing of web pages through the Mobile Proxy is supposed to be done with a handheld device, and these devices are most often used in non-solitary places, which is why the testing environment might have been more "true" when viewing the web pages.

5.5 Conclusions

Based on the previous result section and the discussion of these results, the following conclusions can be drawn.

Too much editing of web page content in order to save space on the small screen can result in loss of some informative value. Furthermore, successful totally automatic extraction of web page content is not possible without using semantic processing, although it is no guarantee for success.

Using the XPaths stored in the database as common denominator and basis from which the algorithm would process unvisited web pages was an adequate choice. This is so as it was possible to implement the Domain Heritage-algorithm into the existing software and that the running algorithm was not experienced to be too slow.

The Domain Heritage-algorithm works satisfactory, but with some experience in using the Page Tailor toolkit and more than one personalised web page, the result is more likely to be satisfying.

6. Future work and legal issues

This final section looks forward; what might become of the Page Tailor toolkit in the next phase and what problems it still has.

6.1 Future work

As stated by C-Y Tsai in the thesis "Web Page Tailoring Tool for Mobile Devices", there are many possible extensions for the Page Tailor toolkit. However, only the issue of automatic processing will be considered here.

This phase of extending the Page Tailor toolkit is only using positioning of previously selected content blocks as data to process web pages in which no content blocks have been saved. In order

to further enhance the functionality of the toolkit, semantic processing of the content that the user has selected from different web pages might be useful. However, this is not necessarily so as a larger set of data would be needed for the processing to be successful, and even with lots of content to process there are no guarantees for a high success rate. There is a notion that a combination of the algorithm of this thesis and semantic processing would be good; one method could be used where the other one has failed.

Another possible extension of the now extended version of Page Tailor is to incorporate RSS-feeds. RSS-feeds are provided by some major news sites and the function of these are similar to what Page Tailor does, and during the presentation the idea of incorporating this function into a future version of Page Tailor was discussed.

Another minor problem that can be solved is that URL-addresses containing special characters like '&', '/', '?', ':', '@', '=', '+', '$' and '#' will not be completely stored; the URL up to the first instance of such a special character will be stored. It is seldom that this causes problems, but does have the effect that if the only difference in the URL-address between two or more pages is following such a special character, these pages will share the same personalisation. The exact source where this trouble arises is in the Javascript pagetailor.js, line 262 (and with Domain Heritage-algorithm implemented, also line 266, same file) as the built in-function encodeURI() processes the URL-addresses. The function encodeURIComponent() might provide help with solving this problem.

At present, there is no simple package available that installs and runs all the Page Tailor toolkit. A level of familiarity with MySQL and code compiling is needed in order start up all of the three parts of the toolkit. The following is needed for full functionality in addition to the source code: a ruby interpreter to run the server, a MySQL-database linked to the ruby-server, a few java libraries and a java compiler to run the Muffin-extractor of the Mobile Proxy. The software package "InstantRails", at www.rubyforge.org, provides a ruby compiler and MySQL-server, and a java compiler can easily be found at Sun Microsystems website, www.sun.com. However, without a single package to install, many potential users will probably find it difficult to use the Page Tailor toolkit and lose interest before having got it up and running. Therefore, it could be considered to create such a single installation package with all the code and compilers included.
Apart from these obstacles, there is another issue to consider if one wishes to setup a server for public usage. The current state of the toolkit is only functional for a single user for each installation, meaning that if two users store different personalisations of the same page, only one (the latest) instance will be stored. Functionality could be added to enable several users to access the same server and create a database for each user.

6.2 Legal issues

At the time of writing this, there is no server running the Mobile Proxy of the Page Tailor toolkit available to the public. Should someone intend to setup such a server for public use, it has to be taken into account that this someone might be legally responsible for the fact that the web pages of different sites will be presented in a different way from how they are supposed to. It is therefore recommended that the user of the Page Tailor toolkit simply sets up such a server for his or her own

usage. Note that it is not illegal to setup a server for public usage, but whether there are legal issues to take into account remains unclear.

On the issue of turning the Page Tailor toolkit, extended or not, into a commercial product or using parts of it in a commercial product, one has to consider the GNU[34]. GNU is a free software foundation, using "copyleft"-licensing instead of copyright. Any software that is licensed under GNU, the whole program and/or parts of it, and its source code is freely available to the public. However, the purpose of "copyleft" is that GNU-software, and/or its parts, may only be used for non-commercial purposes. For instance, the Mobile Proxy is based on Muffin[35], which is licensed under GNU.

7. Acknowledgements

I would like to acknowledge the following persons for their help with the work of this thesis. The order is purely alphabetical.

Karin Wagner, Chalmers; my supervisor who despite the distance between us has managed to help me a great deal with this report and to arrange the oral presentation.
Ray Wang, NCTU; who has been of help to me in the DCS-lab.
Shyan-Ming Yuan, NCTU; who put me in contact with the DCS-lab at NCTU and helped me find a suitable project, and then guided me while working with it.
William Yeh, NCTU; who has helped me with some technical issues during the work with the proposal of this thesis.

I would also recognise all the technical and inspirational help I have received from the members of the "Ruby on Rails forum" at http://www.ruby-forum.com/forum/3, and the members of the "Rails forum" at http://railsforum.com.

8. References

This section is divided into two parts, one with references to related research, and one with miscellaneous references to web sites, web services, cities, universities etc.

8.1 Related research

C-Y. Tsai (2006) *Web Page Tailoring Tool for Mobile Devices*, Department of Computing Science, Institution for Distributed Computing Systems, National Chiao Tung University, Taiwan ROC

S. Agarwal; S. Lamparter (2003) *User Preference Based Automated Selection of Web Service Compositions*, Institute of Applied Informatics and Formal Description Methods (AIFB), University of Karlsruhe (TH), Germany
http://www.aifb.uni-karlsruhe.de/WBS/sag/papers/Agarwal_Lamparter-UserPreferenceBasedAutomatedSelectionOfWebServiceCompositions.pdf

Y. Guo; J. P. Muller (2005) *A Personalized Product Recommendation Algorithm Based on Preference and Intention Learning,* Intelligent Autonomous Systems, Siemens AG, Corporate Technology, Munich, Germany
http://ieeexplore.ieee.org/iel5/10218/32584/01524110.pdf?arnumber=1524110

GroupLens Research, University of Minnesota, USA
http://www.cs.umn.edu/research/GroupLens/index.html

K. Vieira; A. Pinto; N. da Silva; E. S. de Moura; J. M. B. Cavalcanti; J. Freire (2006) *A Robust Method for Web Page Template Detection and Removal* University of Utah, USA
http://www.cs.utah.edu/~juliana/pub/CIKM630-vieira.pdf

Y. Chen; W-Y. Ma; H-J. Zhang (2003) *Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices* Microsoft Research Center Beijing, China PRC
http://delivery.acm.org/10.1145/780000/775184/p225-chen.pdf?key1=775184&key2=5754684611&coll=&dl=ACM&CFID=15151515&CFTOKEN=6184618

V. Crescenzi; P. Merialdo; P. Missier (2003*) Fine-grain Web Site Structure Discovery* Universita Roma Tre, Rome, Italy http://delivery.acm.org/10.1145/960000/956703/p15-crescenzi.pdf?key1=956703&key2=5766684611&coll=&dl=ACM&CFID=15151515&CFTOKEN=6184618

S-H Lin; J-M Ho (2002) *Discovering Informative Content Blocks from Web Documents* Institute of Information Science, Academia Sinica, Taipei, Taiwan ROC
http://cscl.iis.sinica.edu.tw/documents/bobby/infodisc.pdf.

S Debnath; P Mitra; C. L. Giles (2005) *Identifying Content Blocks from Web Documents* Department of Computer Science and Engineering, School of Information Sciences and Technology, Pennsylvania State University, USA
http://clgiles.ist.psu.edu/papers/ISMIS-2005-Content-Blocks.pdf

R. D. Hof; April 27 2005, BusinessWeek, online at:
http://www.businessweek.com/technology/content/apr2005/tc20050427_3567_tc024.htm

M. Pazzani; J. Muramatsu; D. Billsus (1996) *Syskill & Webert: Identifying interesting web sites* Proceedings of the Thirteenth National Conference on Artifical Intelligence, Department of Information and Computer Science, University of California, Irvine, USA
http://www.ics.uci.edu/~pazzani/RTF/AAAI.html

L. A. Fletcher; R. Kasturi (1988) *A robust algorithm for text string separation from mixedtext/graphics images* Dept. of Electr. Eng., Pennsylvania State University, USA
http://ieeexplore.ieee.org/iel1/34/462/00009112.pdf?tp=&isnumber=&arnumber=9112

8.2 Miscellaneous references

[1] National Chiao Tung University, www.nctu.edu.tw
[2] Taiwan Republic of China, http://en.wikipedia.org/wiki/Taiwan_ROC
[3] 228-incident, http://en.wikipedia.org/wiki/228_incident
[4] Kuomintang, http://en.wikipedia.org/wiki/Kuomintang
[5] Hsinchu City, http://en.wikipedia.org/wiki/Hsinchu
[6] Taiwan Semiconductor Manufacturing Company, TSMC, www.tsmc.com
[7] Asus, www.asus.com
[8] ZyXel, www.zyxel.com
[9] BenQ, www.benq.com
[10] Acer, www.acer.com
[11] National Tsing Hua University, NTHU, www.nthu.edu.tw
[12] Information theory, http://en.wikipedia.org/wiki/Information_theory
[13] Microsoft Corp, www.microsoft.com
[14] Opera Mini, mini.opera.com
[15] NetFront, www.access-company.com/products/netfrontmobile/browser/33_wm.html
[16] Opera Small Screen Rendering, http://www.opera.com/products/mobile/smallscreen/
[17] Access Smart-Fit Rendering, www.access-company.com/products/netfrontmobile/contentviewer/mcv_tips.html
[18] Google, www.google.com
[19] Google Mobile Proxy, www.google.com/gwt/n
[20] Amazon.com, www.amazon.com
[21] CdNow, www.cdnow.com
[22] Netflix, www.netflix.com
[23] Jupiter Research, www.jupiterresearch.com
[24] Leavitt Communications, www.leavcom.com
[25] Sun Microsystems, www.sun.com
[26] Gmail, www.gmail.com
[27] Google Maps, maps.google.com
[28] Navicat, a GUI MySQL administration and development tool, www.navicat.com
[29] Pocket PC, www.microsoft.com/mobile/pocketpc/default.asp, http://en.wikipedia.org/wiki/Pocket_PC
[30] Chalmers Asia – Taiwan Office, www.asia.chalmers.se
[31] Natural language processing, NLP, http://en.wikipedia.org/wiki/Natural_language_processing
[32] Learning algorithms, machine learning http://en.wikipedia.org/wiki/Machine_learning
[33] Big O-, Õ- and Ω-notation, http://en.wikipedia.org/wiki/Big_O_notation
[34] GNU, GNU's Not Unix, www.gnu.org
[35] Muffin, Open-source cross-platform filtering Web proxy server written in Java muffin.doit.org

APPENDIX 1: TEST QUESTIONNAIRE

1. Comparing the mobile phone's version of the page with the complete one, do you feel that

something you would have wanted to see is not present in the mobile phone's page? Grade 1-5; 1 everything I wanted to see is there, 5 nothing I wanted to see is there.

2. Comparing the mobile phone's version of the page with the complete one, do you think the missing parts are essential to your experience of the web page?
Grade 1-5; 1 the missing parts are not essential, 5 the missing parts are essential

3. Do you experience the presentation of the content in the mobile phone's version of the web page as different from the complete one?
Grade 1-5; 1 the content is exactly the same, 5 the content is vastly different
Do you have any comment on this?

4. If you experienced differences in the presentation of the content of the web page, did these affect your understanding of the content, i.e. the informative value?
Grade 1-5; 1 my understanding was not affected at all, 5 my understanding was greatly affected
Do you have any comment on this?

5. On the second day. Has the presentation in some way changed from the previous day? If so, mark accordingly new grades above.

APPENDIX 2: ABSTRACT OF "Web Page Tailoring Tool for Mobile Devices"

It is not uncommon to browse Web pages via mobile devices because of their popularity today. However, most Web pages were designed for desktop computers that are equipped with big screens. When browsing on the mobile devices, a user might have to scroll up and down all the time to find the information they want simply because of the limited visible area, which is really not user-friendly at all. Although some famous websites have already provided additional condensed version of their content, it would be cumbersome to spend the time on maintaining multiple versions of the same content from the Web developers' perspective. Besides, some commercial products choose to reformat Web pages to fit the screen width of mobile devices, thereby eliminating the need for horizontal scrolling. However, the result of reformatting Web pages doesn't necessarily help the user a lot. Since the result page still contains the irrelevant information. On this basis, we propose a system to help users personalize Web pages. For example, a user can determine which blocks of content in a Web page should be retained when he/she browses that page on mobile devices. The sequence of those blocks can also be altered according to individual preferences.