

Adaptation of a Webshop for Mobile Devices

by

Muriel Bowie

A thesis submitted in partial satisfaction of the
requirements for the degree of
Master of Science

in

Computer Science

at the

UNIVERSITY OF FRIBOURG, SWITZERLAND

Professor:

Andreas Meier, Head of Information Systems Group (IS), Fribourg

Supervision:

Dr. Henrik Stormer, Assistant Professor at IS, Fribourg

Dr. Donyaprueth Krairit, Assistant Professor at AIT, Thailand

October 2005

Abstract

The mobile web is growing and almost daily new web-enabled devices hit the market. Nevertheless the mobile web is not as successful as predicted, with only very few mobile users that browse the internet on their devices. This is largely due to the fact that websites are not tailored to the need of the mobile community. In this thesis we are showing step by step how an existing webshop can be optimized for mobile devices. We present two different solutions for the implementation of the presentation layer that offers web page rendering according to device capabilities. We will show how the adaptation of the layout can facilitate browsing with mobile devices, especially small-screen mobile phones. In a second stage we address the topic of content personalization by providing a personalization scheme that is based on the Semantic Web technology. We demonstrate how Semantic Web technologies can be utilized for personalization and we present how these methodology considerably simplifies the task of building a personalization system. In the last part we cover the topic of mobile payment from a theoretical point of view and give hints about how such a payment method could be implemented for our webshop.

Contents

1	Introduction	5
1.1	Problem Definition	6
1.1.1	Adaptation of the Presentation Layer	6
1.1.2	The Personalization Module	7
1.2	Thesis Outline	8
2	Related Work	9
2.1	Adaptation of the Presentation Layer	9
2.1.1	Layout Selection Techniques	10
2.1.2	Layout Adaptation with Struts and Tiles	11
2.2	Personalization and Customization	11
2.2.1	Collaborative Filtering	12
2.2.2	Item-based Filtering Techniques	12
2.2.3	User Profiling	13
2.2.4	Personalization and the Semantic Web	13
3	The eSarine Webshop	14
3.1	Struts and Tiles	14
3.2	eSarine Functionalities	15
3.2.1	Storefront	16
3.2.2	Product Categorization	16
3.2.3	Payment methods	16
4	Mobile Devices on the WWW	17
4.1	WML, XHTML, and HTML	18
4.2	UAProf and CC/PP	19
4.3	WURFL and WALL	20
4.4	RIML - Another Device-Independent Markup	22
4.5	Device Independence	24
4.6	Interaction Sessions	25
5	Adaptations for a Mobile Web	26
5.1	Principles of Mobile Browsing	26
5.1.1	Mobile Web Design	27
5.1.2	Opera	28
5.2	Graphical Adaptations	29

6	Personalization	30
6.1	Semantic Web	31
6.2	Content Personalization Techniques	31
6.2.1	Personalization Strategies: An Overview	32
6.3	Web Data	33
6.3.1	User Identification	34
6.3.2	Sessions	34
6.3.3	Clickstream	34
6.4	Personalization and the Semantic Web	35
6.4.1	Introducing Ontologies	37
6.5	User Profiles	38
6.5.1	RDF User Profiles	39
6.6	Explicit versus implicit profiling	39
6.6.1	P3P: Platform for Privacy Preferences	40
7	Mobile Payment	41
7.1	Defining Mobile Payment	41
7.2	Existing Payment Solutions	42
7.3	Micropayments and Macropayments	42
7.4	Mobile Payment Security	43
7.4.1	New Generation Mobile Networks	43
7.4.2	WTLS: The WAP Security Layer	43
7.5	Need for Standards	44
7.6	Mobile Payment Business Models	44
7.7	Simpay, Paybox, and Paypal	44
7.8	A Mobile Payment Scheme for e-Sarine	45
8	Implementation Details	46
8.1	Presentation Layer	46
8.1.1	Adapting the Markup	48
8.1.2	Adding the WURFL Database	51
8.1.3	Struts Actions with WALL	51
8.2	Personalization Module	52
8.2.1	Gathering User Data	52
8.2.2	The Webshop Ontology	53
8.2.3	Creating RDF user profiles	53
8.2.4	The Semantic Inference Reasoner	55
8.3	Conclusion	56
9	Future Work	58
9.1	Device Independence	58
9.2	Personalization	59
9.2.1	Personalization Refinement	59
9.3	Mobile Payment	60
10	Conclusion	61
	Bibliography	63
A	The Webshop Ontology	66
B	A RDF user profile	69
C	XMLSchema for User Profiles	71

D SMIL Examples	73
D.1 Text Positioning	73
D.2 The Switch Element	73
E The eSarine Mobile CSS	75
F WALL: Example JSP-Code	78
G Mobile Screenshots	82

Chapter 1

Introduction

Most web sites are optimized for desktop clients, but are delivering contents poorly suited for mobile devices that can display only a few lines of text transmitted over a slow wireless network. As the number of mobile internet users is growing, the adaptation of web sites for mobile devices such as mobile phones and Personal Desktop Assistants (PDA) becomes increasingly important. In particular e-commerce can take advantage of this trend by providing a customized web site for these customer segments. For users it could be attractive to make small purchases from their mobile devices, independent of time and location. All they need is a mobile device with an internet connection that allows them to access the e-commerce web site. However, as long as the web sites are not adapted to the needs of the mobile community, browsing remains a very uncomfortable, time-consuming, and expensive task due to the long downloading time for large images and HTML tables to the client's device.

Customizing a web site for mobile devices does not only address the re-scaling of traditional web sites to device screen size, but also has to take into account the amount of information to be displayed. A straightforward solution to this problem could consist in displaying only as much text as can be showed on the screen. This method is widely used as it is easy to implement and doesn't require any additional modifications of a web site. Nevertheless, it is not very attractive for users as they might not find what they are looking for and additional steps (or links) are necessary to locate the corresponding information. This may take a lot of time on a slow network. Of course, different approaches to this issue can be considered. A simple solution could consist in providing a search mechanism for a web site to facilitate the access to specific topics. Other solutions have been discussed in the literature [9].

In this thesis, the implementation of a personalization mechanism is proposed, which aims at facilitating the location of information that is of interest for the respective user. The discussion about whether personalization of web content is really worth the effort is still going on. Some argue that most personalization mechanisms that are being implemented nowadays are not suitable for their users, meaning that users are often disappointed and frustrated as they cannot access the information they want with a few mouse clicks. An ideal personalization system would exactly know what the users want and would only provide them with that specific information, saving time and network bandwidth. But as users might change their preferences (especially in a traditional web shop where a user would purchase different items,

not just one), the whole content of a web site has to be easily accessible, even though the customized information should be given a higher priority.

As there is a large variety of mobile devices available on the market, our personalization system should be able to adapt the way content is presented to the respective devices. This requires a personalization module that has to take into account the individual device configurations, such as screen size, network speed (with the emerging UMTS¹ standard more efficient browsing capabilities might become more readily available), and preferences that have been stored on the device (like language settings). Ideally, all users should be able to get a version of the web site that has been optimized to suit their specific interests and device capabilities.

In this thesis, we are not going to present an ideal personalization mechanism that solves all the issues mentioned above, but we will show how such an implementation could be realized, and which methods would allow such an implementation. Therefore, we are going to treat the topic of mobile web personalization from a rather theoretical point of view, and we will present an implementation that has been realized in the context of an existing web shop, eSarine, to show how a desktop-optimized web shop could be adapted for use with mobile devices. We are going to discuss the whole range of issues coming up in this context, from graphical web design adaptations of the presentation layer and content delivery and personalization, to mobile payment.

1.1 Problem Definition

The thesis is concerned with presenting an adaptation mechanism to allow the customization of a web shop for mobile devices, using an additional personalization layer that will process the user's specific interests, preferences, and device capabilities. The implementation has been realized in the context of an existing web shop system, eSarine, which was implemented at the Information Systems group at the University of Fribourg², Switzerland.

1.1.1 Adaptation of the Presentation Layer

Let us first consider the adaptation of the presentation layer. The presentation layer consists of all techniques that are concerned with the way the web site is presented, i.e. displayed to a user.

For the purpose of this thesis, different possible realizations have been investigated:

1. Implementation of a separate skin for mobile browsers with an additional personalization layer.
2. Implementation with dynamic generation of CSS style sheets³.
3. Implementation of a declarative mechanism via XML which allows the dynamic generation of web pages, depending on the properties of the browser and possibly the user preferences.

¹Universal Mobile Telecommunications System

²<http://diuf.unifr.ch/is/>, last accessed in September 2005

³see <http://www.opera.com/products/smartphone/dev/multiple/> for details

However, each method has its advantages and drawbacks. The first solution is quite straightforward and seems to be a natural choice for such a system. Especially as we can take advantage of the given presentation architecture of the webshop that is based on the Tiles framework [24]. Tiles build on the "include" feature provided by the JavaServer Pages⁴ specification to build a framework for assembling presentation pages from component parts. Each part (a "Tile") can be reused throughout an application. This reduces the amount of markup that needs to be maintained and makes it easier to change the presentation of a web site. But this approach has the disadvantage of forcing the separation between the skin, i.e. the appearance, and the personalization mechanism, which should be interacting components in our application. Different browsing preferences, for example, cannot be handled if skin and personalization customization are treated separately.

A very interesting approach to handling presentation could consist in using dynamically generated CSS style sheets. This method would facilitate the implementation considerably, as personalization and appearance could be partly addressed simultaneously, simply by hiding certain objects which are not of interest for a user. Unfortunately, the CSS media type mechanism is not yet supported by all (mobile) browsers and the personalization that could be performed by such a system would not be as powerful as described earlier. A combination of dynamic CSS generation together with an additional personalization module to take care of the content personalization could nevertheless lead to an attractive solution. The idea of using dynamic CSS as a customization mechanism has been proposed by Stormer in [36]

Another way to implement both, appearance and personalization, would be to store the clients' browser and user information in an XML file and to generate the web pages dynamically based on the contents of this XML file. Such an XML file could be generated automatically, combining the user's mobile device configuration file and the user profile. This solution could be used as a personalization mechanism that is not only optimized for use with mobile devices, but also for desktop clients which could adapt to the different connection speeds, the user's screen configurations, and the browser.

1.1.2 The Personalization Module

The personalization module implements an adaptation of the eSarine web shop web site to user preferences and interests. It is important to emphasize the difference between personalization and layout customization: Customization adapts a web site layout to user preferences which is usually done in a static way. Personalization is concerned with the dynamic adaptation of a web site's content and structure. For this thesis, we will use the term 'personalization' as defined above.

Although several solutions for personalization have been proposed in the literature⁵, some of which have been quite successful, we are going to present a different personalization mechanism that is based on the Semantic Web technology. The idea to store meta-data in an RDF⁶ file is not new, but the Semantic Web is not yet widely used for personalization purposes. For a semantic description to be complete, an ontology has to be provided as a base. But not only should the items be labeled with a semantic meaning in a machine-readable format, also user profiles should be stored in the same format. We therefore propose

⁴JSP

⁵see chapter 2 for details.

⁶Resource Description Framework, see <http://www.w3c.org>.

to use RDF for the profiles. For the adaptation of device configurations we are presenting an approach based on CC/PP⁷ which itself is expressed in RDF. Having two different files in RDF format, it could be easily merged into a single RDF user profile as described above. Or the two files could be kept separately, creating a personalized solution based on the contents of both files.

The creation of the user profiles will be done implicitly using a web usage mining algorithm. Given the semantic information no additional classification is necessary. As the information about the user is going to be stored on the server, it is required that the user registers at the web shop. For identification, cookies can be used on the client side, but unfortunately, not all devices provide support for cookies. Therefore, the web shop should provide two levels of customization to the device which is also accessible for not-registered users, as many users accessing via a mobile device will not log in immediately if their browsers don't provide support for cookies.

1.2 Thesis Outline

The thesis aims at proposing an adaptation mechanism that facilitates the access to the web site using mobile devices. First, we are going to discuss related research that has been done in the field, then we are going to give an introduction into the eSarine webshop framework and show how our implementation could make use of already implemented mechanisms. In chapter four we are discussing the mobile web and the vision of device independence on the WWW, thus discussing current and future technologies. The following chapter is dedicated to the adaptation of websites for mobile devices and shows how the presentation layer of our webshop could be adapted for the purpose of this thesis. Chapter 6 is going to address the topic of content personalization and argues how important personalization could become for the mobile web. We are also going to present a personalization mechanism for eSarine that is based on Semantic Web technologies. We are then going to proceed to mobile payment where we will give a short survey of the state of the art and the technologies available. We are then going to outline a straightforward mobile payment solution for eSarine based on Paypal, a payment mechanism that is already implemented in the webshop framework. In chapter eight, we are giving a more detailed overview of the implementation that has been realized in the context of this thesis, discussing several technologies and the way they have been integrated into the webshop system. Chapter nine discusses further improvements that could be implemented as well as technologies that could become prevalent in future and which could be used to improve our implementation. In the last chapter we conclude this thesis with a discussion about the difficulties met during the realization and an overview of what has been done.

⁷Composite Capabilities/Preference Profiles, see www.w3c.org.

Chapter 2

Related Work

The idea of the mobile web has been around for a while. A couple of years ago when the first WAP-enabled mobile phones were available, the mobile web was thought to be the killer application to come. But so far, these high expectations still have to be fulfilled. Of course, the reasons for this lack of interest in the mobile web are obvious, given that the internet is still largely built according to desktop specifications. Japan seems to be the only country where people regularly access the internet from their mobile phones. But it turns out that even though the internet usage is significantly higher than in other countries, Japanese hardly ever browse for information, but rather use their phones as a device for viewing their email, chatting, and reading Japanese Mangas¹ [2].

2.1 Adaptation of the Presentation Layer

Authoring techniques that support device independence should provide mechanisms that allow authors to express the layout of material that varies between different devices with different delivery contexts. In particular, they should support different spacial and temporal layout of material.

W3C: Authoring Techniques for Device Independence [39]

Nevertheless, the idea of the mobile web is not new, several research faculties around the world have conducted studies in the field. The publishing of information for a variety of heterogeneous devices is a big challenge for web developers as it requires the strict separation of data and presentation. Therefore, data has to be structured by storing them in data bases or in XML files [28] and subsequently providing a dynamic layout mechanism, customized to the specific device.

In their paper, "Authoring Techniques for Device Independence" (see [39]), the W3C describes methods that facilitate the development of device independent web pages. The W3C recommends to use CSS Media Queries for rendering to devices. CSS Media Queries allow defining rules based on contextual information. The rules are typically specified as classes which are attributed to the according (X)HTML tags. These rules consequently affect the presentation associated with the class when executed by the

¹Japanese comics

client device or the origin server. For example, this media query specifies that a style sheet applies to a color index device with 256 or more colors:

```
<?xml-stylesheet media="all and (min-color-index: 256)"
  href="http://www.example.com/..." ?>
```

A good overview of CSS Media Queries is given in [38]. Unfortunately, the full version of the CSS Media Queries are only a W3C candidate recommendation for CSS 3.0 which means that they cannot be used at the time of writing for the purpose of our implementation, but they seem to address the main issues for device independent authoring mentioned in this thesis. Another disadvantage of using CSS for device independent rendering is that CSS 2.0, which implements a more restrictive form of the Media Query mechanism is not yet widely supported by mobile browsers.

In order to provide support for a wide range of different delivery contexts, it is necessary to provide a variety of different layouts. As discussed above, the current CSS 2.0 specification only offers limited support for media types. Let us therefore consider layout customization techniques that are available today [39]:

- CSS: the `<div>` element allows to associate size and position information in a style sheet. `<div>` allows to target content to a particular place in the layout.
- Device dependent style sheets: selection of different versions of entire style sheets according to the requesting device capabilities.
- SMIL 2.0 [40] offers an explicit layout notion with the elements `<layout>`, `<root-layout>`, and `<region>`. SMIL contains several layout modules for specific rendering types. A SMIL-based implementation of device independence could be used in conjunction with a server-side adaptation mechanism that constructs the SMIL markup dynamically. An example of position text using SMIL can be found in D.1
- WALL (see chapter 4.3 or [32]) enables developers to define a single page markup that is automatically adapted for the different device capabilities which are saved in the WURFL data base, a data base that contains the device capabilities of most available devices. Therefore, the device independent rendering is done by the framework, not the developer.

2.1.1 Layout Selection Techniques

A different approach to layout customization can be achieved by consequently applying switch-statements in the markup that chooses the appropriate settings for the specific device. Here again, CSS Media Queries can be used for the selection of the appropriate stylesheet:

```
<link
  rel = "stylesheet"
  type = "text/css"
  media = "handheld"
```

```
    href = "mobile_device.css"
>
<link
  rel = "stylesheet"
  type = "text/css"
  media = "screen"
  href = "desktop.css"
>
```

The two statements above can be added to any existing (X)HTML web site in order to provide different style sheets for different media types². Besides, the media type can also be specified in the CSS style sheets by applying the *@media* – rule:

```
@media print {
  BODY { font-size: 10pt }
}
@media screen {
  BODY { font-size: 12pt }
}
@media screen, print {
  BODY { line-height: 1.2 }
}
```

Another interesting method that is described in [39], is based on the SMIL switch-element that solves the problem of creating a presentation from available fragments based on client capabilities. In SMIL the *< switch >* element enables a developer to specify a list of alternative elements which are selected in boolean tests. A default selection can be defined at the last position by not defining any kind of constraints. For an example of the SMIL switch element see Appendix D.2.

2.1.2 Layout Adaptation with Struts and Tiles

Tiles [24] which will be introduced in chapter 3.1 is a tag library that enables the composition of a web page with different layout components. Additionally, Tiles allows to define so-called skins, i.e. a set of components that generate a consistent presentation of a web site. Tiles skins could be used for mobile device rendering by defining a separate skin for every class of devices. The selection of the appropriate skin has to be done dynamically based on the device capabilities.

2.2 Personalization and Customization

The continuing growth and complexity of Web-based applications has led to a rise of personalization tools on a variety of web sites. These personalized services help engage visitors to more effectively locate adequate information. One of the most successful and widely used technologies is collaborative filtering (see [27], [10], and [29]). Collaborative filtering is used for both, general personalization applications and recommender systems.

²A list of all currently supported media types is available on <http://www.w3.org/TR/REC-CSS2/media.html>, last visited in August 2005

2.2.1 Collaborative Filtering

Traditionally, collaborative-filtering-based systems compare the preferences of active users (such as explicit item ratings or implicit navigational patterns) with the records of other users in order to find the K most similar neighbors of the user. These records are then used to predict the preference value of the user on a particular item. These same records might also be used to recommend the top N items which could be of interest to the user. As these systems concentrate on comparing the correlations or similarities among users, they are often referred to as user-based collaborative filtering systems. Although collaborative filtering is very widely used, it suffers from some several major limitations as Sarwar et al. describe in [33], such as the lack of scalability of memory-based k -nearest-neighbor approach. For very large data sets this may lead to unacceptable delay for providing recommendations. The scalability problems are even worse when collaborative filtering is used with Web usage data. In this case, a users browsing patterns are required to implicitly deduce content preferences. Another drawback of collaborative filtering is that the systems are built on top of sparse data sets. Finally, a very important disadvantage of such systems is their inability to provide recommendations or predictions for new items. A recommender system can never generate predictions for new items which have not yet been visited or rated by other users. This problem is often referred to as the new item problem. Several strategies have been proposed to meet these problems like similarity indexing [5] that aims at reducing real-time search costs, or approaches to reduce dimensionality issues based on Latent Semantic Indexing (LSI) [33]. Other methods have attempted to redefine the models by using machine learning techniques, such as unsupervised clustering of user records [30] or supervised classification [10]. These approaches allow the separation of creating user models from the real-time recommendation generation.

2.2.2 Item-based Filtering Techniques

Lately, a new class of collaborative filtering algorithms have been introduced, the so-called item-based collaborative filtering algorithms. They have been proposed in order to meet the scalability problems in user-based CF algorithms (see [33] and [13]). Item-based CF algorithms avoid user-user computations by first considering the relationships among items. Rather than finding user neighbors, the system tries to find k similar items that are rated (or visited) by different users in some similar way. In consequence, predictions can be generated for each item, for example, by taking a weighted average of the target users item ratings (or weights) on these neighbor items. Thus, these algorithms address the scalability problem that exists in user-based CF algorithms. Additionally, item-based CF algorithms have been shown to provide better predictions than user-based CF algorithms. Item-based CF algorithms still suffer from the problems associated with sparse data, and they still cannot provide recommendations or predictions for new items. However, the item-based CF framework provides the necessary ingredients to introduce other sources of evidence about items (in addition to item ratings or weights). This flexibility comes from the fact that the computation of item similarities is independent of the methods used for generating predictions or recommendations. Therefore multiple knowledge sources can be used for item similarity computations.

2.2.3 User Profiling

A user profile can be thought of being "a set of data representing the significant features of the user" [28]. Kontinurmi [28] believes that user profiling will become more and more important in future, especially with the arrival of new heterogenous mobile devices. He argues that in future it will be crucial to provide more customized ways of accessing services.

Anderson et al. [6] propose to personalize web content for mobile device users for the following reasons:

- make frequently visited destinations easier to find
- highlight contents that might be interesting for a user
- omit uninteresting content and structures

Explicit Versus Implicit Profiling

Data for user profiling can be collected implicitly or explicitly [11]. Explicit collection generally requires that users actively participate by providing information about their interests and preferences. This method allows the users to control the information in their respective profiles. Explicit profiling can take different forms. The user may fill out a form, take part in a survey, or submit personal information at the time of registration. Additionally, users may provide ranking or rating of products. This approach has the advantage of letting the customers tell a website directly what they need and how they need it. Implicit profiling, in contrast, does not require the user's participation as information about user preferences are retrieved implicitly from server access logs or other sources. This method has the drawback that users are not necessarily aware that their profiles are being extracted and therefore raise several privacy issues.

2.2.4 Personalization and the Semantic Web

In 2001 Kontinurmi [28] proposed to implement a Semantic Web approach to user profiling, although his view of the Semantic Web is still quite imprecise, he believes that "managing user profiles like ontologies may be an answer to some problems we are facing today". Therefore, the idea of using Semantic Web technologies to implement a personalization mechanism is not new. Nevertheless, even today, only very few implementations are available. One of which is the GraniteNights project that has been proposed by Grimnes et al. ([20], [21], and [15]). GraniteNights is an application that provides a personalized evening scheduler for the city of Aberdeen based on a Semantic Web approach. The GraniteNight evening scheduler is implemented as a multi-agent platform that retrieves information from RDF user profiles, and compares them to a given ontology³ in order to provide a personalized time-table for an evening in Aberdeen.

³Ontology in HyperDAML

Chapter 3

The eSarine Webshop

eSarine is a web shop framework that has been developed by the Information Systems Group at the University of Fribourg, Switzerland¹. It aims at providing a modular architecture for a web shop system based on Java Technologies, that allows the independent handling of different aspects which are part of the system. The architecture hence supports different databases allowing to build a web shop on top of an existing database. Additionally, the extensive use of the Jakarta Struts [24] framework allows a separation of the skin, i.e. the presentation layer, of the website and its logical architecture [41].

3.1 Struts and Tiles

eSarine makes extensive use of Jakarta Struts and Tiles ([24], [34]), Java-based technologies that allow the integration of different aspects of web applications. The core of the Struts framework is the control layer that has been developed based on standard technologies like Java Servlets, JavaBeans, and XML. Struts assists application architectures to implement the so-called the JSP Model 2 approach, a variation of the classic Model-View-Controller (MVC) design paradigm.

Struts implements its own controller component and integrates very well with other technologies for the implementation of model and the view. Struts can be used with standard data access technologies, like JDBC and EJB. For the realization of the view-part, Struts works well with JavaServer Pages (JSP), and other presentation systems.

The Tiles framework allows building of web pages based on JSP by assembling reusable components. As an example, let us consider a simple web page consisting of four different components:

- A header that contains a web site logo and a welcome notice
- A menu that provides links to different web site locations
- A main window that contains the actual web site content such as product information etc.
- A footer for copyright information

¹<http://diuf.unifr.ch/is/research/esarine/introduction.php>

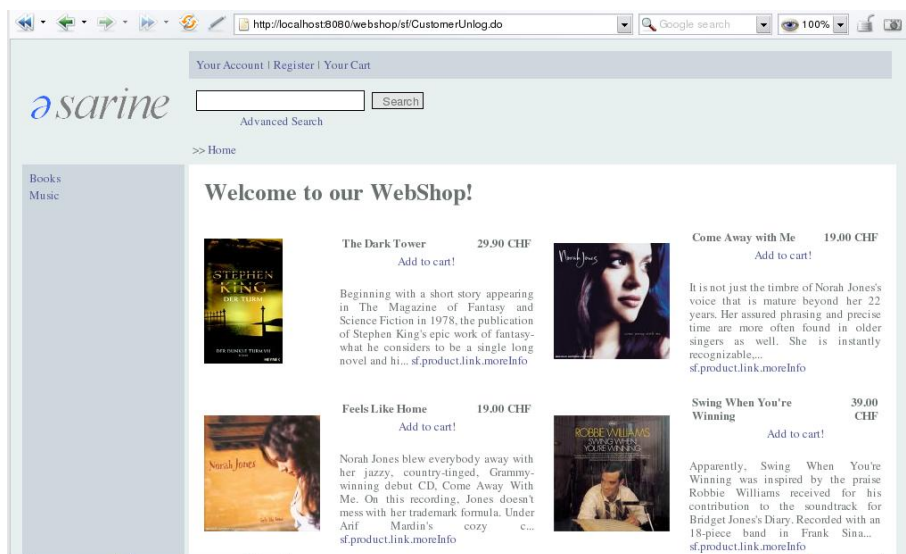


Figure 3.1: The eSarine web shop storefront as presented on a desktop computer.

Assuming that these are the components on which we want to build our web pages, it would be interesting to write the JSP code only once, thus reusing it throughout all available web pages. In Tiles, we can define the components ("Tiles") and import them to different web pages. In fact, Tiles builds on the "include" feature provided by the JSP specification to provide a robust framework for assembling presentation pages from component parts. The utility of tiles is quite controversial as similar functionalities can be achieved by simply using the JSP include feature. Nevertheless, Tiles was an ideal candidate for the implementation of the eSarine web shop as it permits to define different skins, i.e. a set of tiles that constitute a skin, which is one of the most significant philosophical statements of the eSarine web shop. The strict separation of model and view allows the realization of different views for a single model. This is especially interesting in the context of this thesis where we are going to present an approach for adapting the eSarine web shop for use with mobile devices. In Struts the controller is responsible for dispatching requests to the different models and views. The configuration of the controller is recorded in the XML configuration file. As a result, Struts propagates efficient component reuse and considerably facilitates adjustments in implementation layers without affecting the structure of other layers.

3.2 eSarine Functionalities

The eSarine web shop is composed of two different components: The storefront and the storeback. While the storefront is the visible part of the webshop that is presented to the customer, the storeback component is exclusively implemented for the remote administration of the webshop [41]. The storeback addresses the different web shop management requirements like product management, locale and the user administration, as well as the payment and the shipping maintenance.

This thesis aims at adapting the storefront part for use with mobile devices. In the next subsections we

will give a short overview of the functionalities that are relevant for the purpose of this thesis. A more complete guide to the eSarine web shop can be found in [41] and [19].

3.2.1 Storefront

The storefront constitutes the presentation layer and point of access for customers. As a reliable implementation of the presentation part is crucial for the success of a web shop, eSarine enables developers to integrate new views, or so-called skins, with minimal effort. This facility is largely due to the rigorous implementation of the Struts MVC pattern that ensures the strict separation of the view and the business logic.

The design of the web pages is based on JSP, using another helpful functionality that works very well in combination with the Struts framework: the Tiles tag library, see chapter 3.1 for more details about Tiles.

The eSarine web shop features all usual web shop functionalities such as a shopping cart, a secure log on and a registering mechanism², as well as cookies-based identification, a search function, and a consistent access to products through categories. Furthermore, eSarine offers support for different languages (and browser locales), currencies, and skins, which means that the web shop can be customized to a certain extent according to user preferences.

3.2.2 Product Categorization

eSarine implements a consistent categorization and product type concept that makes it mandatory for products to be part of a category. In addition, product types allow to define templates for a certain kind of products. A music CD for example doesn't need to have a weight attribute, whereas a notebook computer might need it. That is, each product type can be defined by a set of attributes, like product types, categories, products. Attributes all have unique names for reasons of identification.

The web shop offers two different access methods to the products: by search and by category. In either case the results will yield to a list of products with a short product description. If customers are interested in a product, a link will direct them to the product details with a more extensive description and a larger picture of the selected product.

3.2.3 Payment methods

The eSarine web shop system offers three different payment models: manual credit card payment, payment by invoice, and PayPal. The models are implemented as independent modules which allows web shop managers to enable or disable them, as well as adding new modules according to their requirements. The module for credit card payment does not include the implementation of a financial gateway interface as these gateways require a special contract with the providers which has to be negotiated with the provider individually. eSarine thus offers a way to send sensible credit card payment over a secure connection. The actual payment has to be handled manually by the shop owner.

²The Struts framework offers a secure log on mechanism

Chapter 4

Mobile Devices on the WWW

In the last couple of years, World Wide Web enabled devices started occurring on the market. These days, almost all mobile phones offer browsing capabilities. But mobile devices, in particular mobile phones, are still rarely used to access the internet. Some client devices adapt content at the device. For example, Windows-CE devices change color-depth (for example, from 24 bit color to 4bit gray-level) of images. The drawback is that network appliances have low network bandwidth, resulting in a slow access to pages with rich multimedia content. Another problem is that network appliances are often restricted in their computational power, which makes content adaptation at the device slow, or even impossible. Google¹ offers a very interesting approach to mobile browsing by offering websites in WAP and XHTML format, optimized for the respective device. Additionally, Google converts the target pages of a search into the appropriate format which makes some of the pages browsable with a mobile device.

Wireless networks are still quite expensive. Although the bandwidths have been considerably improved, it occasionally happens that it falls back to previous low levels [22]. The GSM² supplement General Packet Radio Service (GPRS) that is widely used nowadays addressed the two major problems that hindered the propagation of the mobile web:

- no dial-in over a circuit switched network is necessary, as the device is constantly connected to the packet-switched network. This considerably improves connection speed.
- transaction costs: with GPRS the cost management of an internet connection is facilitated. Today, network operators can charge for the amount of data being transferred which is generally cheaper for users than per-second-billing on a slow network.

It is important to stress that the price of network connections also play a crucial role for the success of the mobile web.

UMTS³ is part of the international vision of a global family of third generation mobile communication systems. It is often referred to as a 3G (for third generation) network. It is generally based on the

¹<http://www.google.com>

²Global System for Mobile (GSM) communications, the most widely used digital mobile phone system and the de facto wireless telephone standard in Europe.

³Universal Mobile Telecommunication System

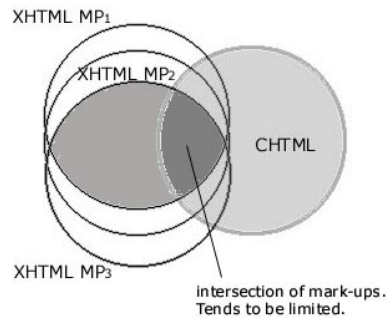


Figure 4.1: Overview of the different mobile markup languages (Image source: [32]).

W-CDMA⁴ technology, and allows for network speeds up to about 2 Mb/s. The first large scale real-life commercial UMTS network in the world was introduced in 2001 in Japan operated by DoCoMo⁵. Unfortunately, at the air interface level, UMTS is incompatible with GSM. Therefore, all UMTS phones sold in Europe (in 2004) are UMTS/GSM dual-mode phones, hence they can also make and receive calls on regular GSM networks. Generally, UMTS is believed to become an important promotor of the mobile web, as it allows connection speeds comparable to high-speed desktop internet connections. But, even though the problem of connection speeds might be solved with the proliferation of UMTS-enabled phones, the question of displaying and adapting web sites on mobile phones remains and has to be addressed in the near future in order to make the web attractive to mobile users.

4.1 WML, XHTML, and HTML

More and more new WAP 2.0⁶ devices hit the market. WAP 2.0 introduced XHTML MP 1.0⁷ as the markup for developing wireless applications. While XHTML MP gives up on many of the usability extensions introduced with WML 1.X, it promises to bring convergence with HTML. But situation for mobile web development remains confusing. Many different standards need to be integrated in order to make a web application device independent. It becomes even worse as the standards are not implemented in the same way on different devices. XHTML-MP implementations for example differ from browser to browser. These differences cannot be neglected as some can make an application fail on certain devices unexpectedly. In some cases, the differences are so significant that it is impossible to deploy an XHTML MP application which works very well for one device on a different WAP 2.0 enabled device. To add to that, another markup language is coming up on GSM networks: Compact HTML (CHTML), the mark-up language of NTT Docomo's I-Mode. CHTML also derives from HTML, but is very different and cannot inter-operate with WAP 2.0.

⁴Wideband Code Division Multiple Access

⁵see <http://www.nttdocomo.com/> (last accessed in August 2005) for details

⁶For a specification of WAP 2.0 see <http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html>

⁷XHTML Mobile Profile

4.2 UAProf and CC/PP

If we want to build websites that are device independent, we need to know more about the different devices and their capabilities. Therefore, we need to find a mechanism that allows us to extract device-specific information when a mobile device client requests a page from our web server. But is this information available? And if yes, where can we find it? A straightforward way to find out more about a client requesting a web page could consist in analyzing the contents of the HTTP headers that are sent with the request. And in the HTTP header we find a link to the browsers' X-Wap-Profile:

```
User-Agent: OPWV-SDK UP.Browser/7.0.2.3.119 (GUI) MMP/2.0 Push/PO
X-Wap-Profile: "http://devgate2.openwave.com/uaprof/OPWVSDK70.xml"
```

From the HTTP header we can see that the client is an Openwave browser, but in fact the second line, the X-Wap-Profile, is almost more interesting for us. But this might need some further explanation. The X-Wap-Profile property contains the URI of a UAProf/XML document. UAProf⁸ is a WAP Forum specification that is designed to allow wireless mobile devices to declare their capabilities to data servers and other network components. So this is in fact the information we need for our device independent webshop. Before heading further, we would like to introduce another device profile standard: CC/PP.

A CC/PP⁹ profile is a description of device capabilities and user preferences. This is often referred to as a device's delivery context and can be used to guide the adaptation of content presented to that device. CC/PP is a W3C recommendation that uses RDF as the format to describe device capabilities and preferences. A general CC/PP profile will be organized as follows:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ccpp="http://www.w3.org/2002/11/08-ccpp-schema#"
  xmlns:ex="http://www.example.com/schema#">

  <rdf:Description
    rdf:about="http://www.example.com/profile#MyProfile">

    <ccpp:component>
      <rdf:Description
        rdf:about="http://www.example.com/profile#TerminalHardware">
        <rdf:type
          rdf:resource="http://www.example.com/schema#HardwarePlatform" />
        <ex:displayWidth>320</ex:displayWidth>
        <ex:displayHeight>200</ex:displayHeight>
        </rdf:Description>
      </ccpp:component>

      <ccpp:component>
        <rdf:Description
          rdf:about="http://www.example.com/profile#TerminalSoftware">
          <rdf:type
            rdf:resource="http://www.example.com/schema#SoftwarePlatform" />
          <ex:name>EPOC</ex:name>
```

⁸UAProf: User Agent Profile

⁹Composite Capabilities/Preference Profiles

```

    <ex:version>2.0</ex:version>
    <ex:vendor>Symbian</ex:vendor>
  </rdf:Description>
</ccpp:component>

<ccpp:component>
  <rdf:Description>
    rdf:about="http://www.example.com/profile#TerminalBrowser">
    <rdf:type
      rdf:resource="http://www.example.com/schema#BrowserUA" />
    <ex:name>Mozilla</ex:name>
    <ex:version>5.0</ex:version>
    <ex:vendor>Symbian</ex:vendor>
    <ex:htmlVersionsSupported>
      <rdf:Bag>
        <rdf:li>3.2</rdf:li>
        <rdf:li>4.0</rdf:li>
      </rdf:Bag>
    </ex:htmlVersionsSupported>
  </rdf:Description>
</ccpp:component>

</rdf:Description>
</rdf:RDF>

```

The initial sub-tags of the CC/PP profile describe the main components of the client, such as hardware platform, software platform, and browser. Given that CC/PP is expressed in RDF/XML, general XML or RDF parsers and API can be used to read and interpret the profiles. If we could assume that all mobile devices implement the CC/PP profile, we could implement a device independent version of our webshop based upon the device information delivered by the CC/PP profile.

The CC/PP model generally follows UAProf, although its RDF schema is more prescriptive regarding class and property usage than UAProf. But the design of CC/PP is backwards compatible with UAProf, meaning that valid UAProf profiles are also valid CC/PP profiles. Not all CC/PP profiles, however, are valid UAProf profiles.

Once we have determined where we can find information about device capabilities, we have to solve the problem of analyzing the respective UAProf and CC/PP profiles. As we have seen the device profiles are given in form of a URI that points to the real profile. But how can this be implemented in a real-world application? It is almost impossible for a web server to first go and fetch a profile from an URI in order to tailor a website to the device capabilities. Ideally, the webserver should have access to a list of all device capability profiles which he can query when a new request is issued. Luckily, such a list already exists as we will see in the next section in which we will introduce WURFL, a device capability database.

4.3 WURFL and WALL

The WAP standard was established in 1998 when all the major players of the mobile phone industry came together to create a worldwide standard for the mobile web, the Wireless Application Protocol

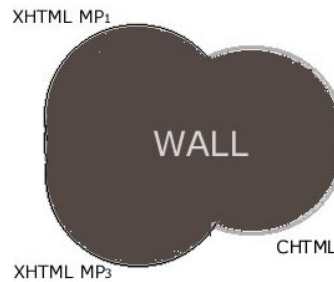


Figure 4.2: Wall supports the sum of all different mobile markups (Image source: [32]).

(WAP). Although the WAP protocol has been implemented and supported by all device manufacturers and network operators, practice shows that phone and browser manufacturer have been implementing different functionalities without conflicting with the WAP protocol. As a consequence, developing applications for wireless devices has become more complicated as the different interpretations of the WAP standard have to be handled separately. Mobile web developers need means to:

- define an abstraction for devices differences
- a way to add new device specifications without changing the whole web application

And this is where the WURFL¹⁰ [32] database comes in. WURFL is a global database that contains all available devices and their capabilities (UAProf or CC/PP). With WURFL we don't need to retrieve the capability profile from some notional URI, all we need is a way to efficiently look up the requested information in the database and include it in our application to dynamically tailor the websites to the devices specifications. And this can be done with the WALL tag-library that has been developed by L. Passani and A. Trasatti[32] who have implemented the WURFL database.

WALL: A simple Example

By exploiting the power of the WURFL device capability database (<http://wurfl.sourceforge.net>), WALL can transparently detect what a device can do and make sure that the best possible mark-up is delivered to it. To illustrate how WALL can be useful for the implementation of a device independent web application, let us consider a widely used HTML-tag: the line break `< br >`.

Before XML started coming up as a web standard, the HTML-tag for a line break was a simple `< br >`. But with the appearance of XML the W3C started adapting HTML 4.0 to the XML standards. This is when XHTML 1.0 (from which XHTML Basic and XHTML MP were successively derived) was born. As a consequence, the simple `< br >`-tag had to be expressed according to the new XML rules as `< br / >`. And now, when it comes to wireless devices, the situation is fairly complicated:

¹⁰Wireless Universal Resource File

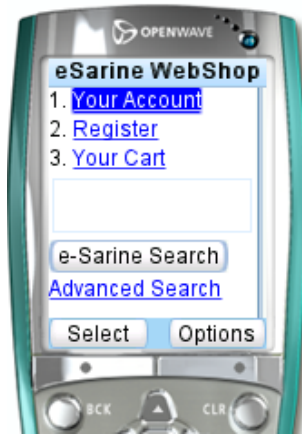


Figure 4.3: A WALL-based implementation on Openwave 7.

- most CHTML devices only accept the `< br >` tag and simply ignore `< br/ >`
- most XHTML MP devices will display both `< br/ >` and `< br >` tags.
- some XHTML MP device will throw an error if a single un-slashed `< br >` is found and no information will be displayed

When confronted to such tricky matters, web developers soon come to the decision to either not support device independence at all or to restrict the ease of access to the most commonly used devices. This is where the WALL tag library comes in, as it facilitates the situation considerably by taking care of all these matters. Web developers only need to write a single tag: the `< wall : br/ >` tag. And this tag will be sufficient to optimize the markup for a given device. It will look to check the user-agent, query the WURFL database to determine the ideal mark-up for the requesting device, and it will issue a `< br/ >` or `< br >`, depending on the capability of that device.

WALL: Another example

To illustrate the power of the WALL tag library, let us consider another example. The JSP code in appendix F produces the following outputs depending on the requesting device: As the images show, the WALL tag library really allows the implementation of device independence. Unfortunately, this same markup is not adapted for desktop rendering as desktop browsers do not have CC/PP profiles. This means that on a desktop browser, a CSS stylesheet has to be added for the presentation.

4.4 RIML - Another Device-Independent Markup

The Consensus project¹¹ is a European consortium of companies that was founded in order to define the Rendering Independent Markup Language (RIML) specification. The project aimed at

¹¹The project website: <http://www.consensus-online.org> was no more available at the time of writing.



Figure 4.4: The same WALL markup as in 4.3 on Openwave 6, notice that the search form is not being displayed as these methods are not supported by WML 1.x, a work-around to this problem is proposed in chapter 8.



Figure 4.5: The JSP code from appendix F displays a menu with icons on a typical WAP 2.x-enabled mobile phone screen (image source: [32]).

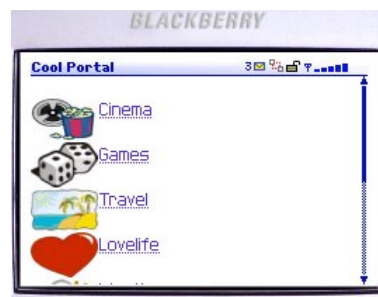


Figure 4.6: The same WALL markup produces a different output on an XHTML-enabled PDA (image source: [32]).



Figure 4.7: On WAP 1.x devices, only WML is rendered to the device (image source: [32]).

enabling access to Web content for mobile devices. The idea, similar to that of the WURFL/WALL implementation, was to define a list of devices and their UA features in order to provide device-specific content rendering. RIML is based on W3C technologies like XHTML 2.0, XForms, and SMIL. However, the adoption of these technologies by the World Wide Web is very improbable in the years to come. The XHTML 2.0 specification, as an example, is not backwards with the current web-markup languages HTML and XHTML 1.0. The RIML definition includes a set of attractive features such as pagination [35], layout, and offers navigation in the device-specified language. Although, the approaches of WALL and RIML are very similar, RIML suffers from several serious drawbacks [22] as the implementation of the language features itself is very heavy-weight and it requires a proxy server in order to avoid for users to install RIML support on their devices. At the time of writing it is not clear how such a service could be integrated into today's architecture of the World Wide Web.

4.5 Device Independence

The range of capabilities for input and output and the range of markup languages and networks supported greatly complicate the task of authoring web sites and applications that can be accessed by users whatever device they choose to use. Device Independence encompasses the techniques required to make such support an affordable reality.

W3C, Device Independence¹²

The mission of the W3C Device Independence Working Group is to avoid fragmentation of the Web into spaces that are accessible only from subsets of devices.

Web developers can no longer afford to develop content that is optimized for use with a single device type such as desktop-computers. The key challenge facing this problem is to allow for content or applications to be delivered to a variety of devices with a minimum of effort. The implementation of a web site that follows the principles of device independence could potentially save costs, and could help developers in providing users with an improved user experience that is independent of time, place and device.

Ideally, from a developers point of view, it should be possible to provide a functional user experience, in

¹²<http://www.w3.org/2001/di/>

response to a request for a web page, in any given context optimized to the capabilities of the respective device. Nevertheless, the developer should be aware that sometimes it is not possible to provide some content or functionality in a given delivery context because of device limitations. It is therefore necessary to first determine the device capabilities and limitations before providing the user with the requested information. Images for example should be resized before sending in order to avoid unnecessary transmissions over a slow network.

With the intention of meeting the requirements for device independence, W3C proposes an XML-based "Device Independent Markup Language Profile"¹³ which allows to tailor content to different device types. Unfortunately at the time of writing the development of the markup language was still in working draft 2 stage with no concrete implementations available. But this could be a very interesting approach to device independent authoring that could be pursued further.

4.6 Interaction Sessions

A session is a sequence of interactive steps that influence an information state (context) that persists between each step in a sequence¹⁴. Typically, a session is started by a user and lasts over the subsequent requests of that same user in interaction with a server. Sessions are a major component in many web-based applications, as they allow a "dialogue" between a user and a server by saving relevant information that are available for future requests. The stored information might be important for security, authentication, or user preferences.

Most session techniques rely on the cooperation of the client application to maintain sessions. Generally, the server will produce a unique key to identify the session. The success of a session initiation depends on the willingness or capacity of a browser to supply the key on subsequent requests. In HTTP, cookies are normally used to transmit session keys. Unfortunately, not all mobile devices provide support for cookies and sessions. Therefore, if HTTP cookies are not available, different solutions have to be considered:

- an intermediate (proxy) could supply this functionality, as available in the WAP protocol. Additionally, since WAP 2.0 WSP¹⁵ is available. WSP provides HTTP/1.1 functionality and incorporates new features, such as long-lived sessions. suspend/resume.
- URL rewriting: session keys are embedded in URLs

We have considered both possibilities for our implementation, but we haven't implemented a mechanism to meet this problem. This should be part of a future release of the mobile module for the eSarine web shop. Nevertheless, we have included URL-rewriting in our WALL-implementation of the web shop.

¹³<http://www.w3.org/TR/cselection/>

¹⁴Definition according to [39]

¹⁵Wireless Session Protocol, see [18]

Chapter 5

Adaptations for a Mobile Web

The idea of the mobile web is to make the world wide web accessible for all kinds of mobile devices. This means that the internet can be accessed everywhere and at anytime. Experts believe that the future of web lies in the mobile devices. In Thailand, for example, there are now 23.3 million mobile-phone users countrywide ¹ and in the UK 8 billion webpages have been viewed from mobile phones in 2003 [25]. Unfortunately today's websites are only optimized for the use with desktop clients. When browsing the internet with a mobile device, users are very often deceived by the poor navigational facilities of the pages. Navigating on a traditional desktop-oriented website with a mobile phone generally requires a lot of time as horizontal and vertical scrolling is required, and large images considerably slow down the loading process. Therefore, if some day the mobile web should become a reality, adaptations of traditional websites are urgently required.

In this section we are going to give an overview of the problems that are being encountered when browsing websites with mobile devices and we are going to discuss solutions that can be adapted for already existing web pages.

5.1 Principles of Mobile Browsing

Studies have shown that on desktop computers, a large number of users abandon a page that takes more than 10 seconds to load. This means for general web design that web pages should contain less than 9 seconds worth of size. As connection speeds have been increasing rapidly over the past couple of years, 9 seconds can contain a lot more information, presuming that the site can afford to make slower-connection users wait longer. Unfortunately, for mobile phones these desktop-optimized pages are almost impossible to view comfortably.

Mobile devices are different from desktop clients in several aspects:

- the have less memory
- small display

¹ Reference: The Sunday Nation, January 23, 2005, Bangkok

- slow connection speeds

To meet these requirements, web designers have to perform a considerable number of changes on their websites. This means that websites designed for mobile phones should be as lightweight as possible, but contain as much information as necessary. A good overview about web design for mobile devices is given in [7].

5.1.1 Mobile Web Design

Designing a web page for device independent rendering has become a major issue in the past years. The wide range of devices that are currently in use make it difficult, if not almost impossible, to provide device independent content. Different standards and browser capability issues as described in chapter 4 have to be addressed by web developers. The most mechanism to solve issues related to device independence are WALL and WURFL (see section 4.3) that allow to address many of the current problems. For the purpose of this thesis we have implemented two different prototypes for our mobile rendering system, the first one is based on WALL, and the second one uses a CSS 2.0² approach with support for different media types. The two implementations address different device types, whereas WALL (see section 4.3) offers an customization to most devices, CSS 2.0 is only supported by very few browsers such as Opera (see section 5.1.2) and Openwave. But as we believe that in future, most mobile devices will implement the CSS 2.0 standard, we decided to include this technique in our implementation. In this subsection we are going to give a short overview of mobile web design principles as described in [7]:

Scrolling is Better than Fetching

Unfortunately, many developers wrongly interpret the notion that websites should be as light-weight as possible and conclude that each page should contain as little information as possible. But this is not true, in the contrary, each page should actually contain as much relevant information as possible, within its capacities of course. As an example, consider a menu with nine items, a comfortable limit for a list on a scroll-and-select device. Scrolling through a nine item list is no more difficult than scrolling through a list with four or five items.

Avoiding Ambiguous References

References should be clearly labeled in order to avoid confusion, especially if a link leads to an external site. Especially references labeled with search should be clarified as users might tend to misinterpret its meaning. Therefore, the label should mention what is actually being searched (the WWW or the website) in order to avoid potential confusion. For example, if you are linking to a general web search engine, "Web Search" could help users avoid a potential misinterpretation. Or for e-Sarine the label "e-Sarine Search" would be better than just "Search".

Another very helpful addition, according to [7], that facilitates browsing with mobile devices is to put

²<http://www.w3.org/TR/1998/REC-CSS2-19980512/>

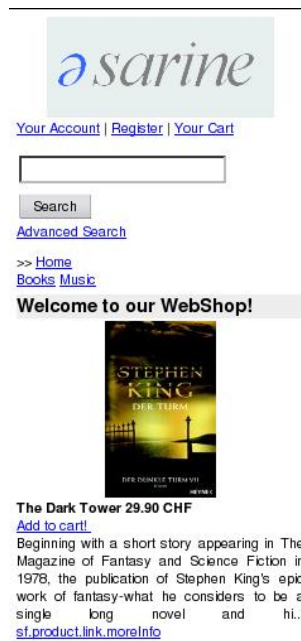


Figure 5.1: Viewing the eSarine web shop with Opera small screen rendering.

navigation links at the bottom of screen. This way, users don't not have to scroll through them to see the content and they also don't have to scroll back to the beginning of the page to find them. Of course, this should not be taken as a general rule. The entry page³ should of course highlight navigational links to other pages on the site and put them on top of the page, whereas information pages requested by the user should provide navigational links at the bottom.

5.1.2 Opera

Today's mobile phones are normally shipped with a pre-installed browser. Different browsers are currently available for mobile phones, although two browsers seem to be the most widely used: Openwave⁴ and Opera⁵. Unfortunately, it is very difficult to provide a optimized solution for each different browser as different each browser implements the W3C standards more or less rigorously.

Opera, the browser that is included in some mobile phones as well as PDA's implements a special mechanism that offers small screen rendering. That means, an Opera browser will automatically reformat traditional websites to fit inside the screen with, thus eliminating horizontal scrolling. Another interesting feature of Opera small screen rendering is how it handles large images. In fact, image sizes are adapted to fit into screen with. And Opera control the image download process by only downloading the reduced image, not the full size one which makes browsing more efficient.

After considering the advantages and the drawbacks of the small screen rendering technique, we decided not to make direct use of it. One major disadvantage of Opera is that it doesn't allow the combination

³or homepage

⁴<http://www.openwave.com>

⁵www.opera.com

of small screen rendering with an additional CSS stylesheet. In order to optimize our eSarine web shop for use with mobile devices, we therefore decided to write our own stylesheet, thus not making use of the small screen rendering technology.

5.2 Graphical Adaptations

In this thesis, we propose two different approaches to the graphical interface of our webshop. The first one is based on the use of separate CSS stylesheets for desktop and mobile users, making use of a media type that is called "handheld" and that can be used in the HTML link-tag when defining the stylesheet:

```
<link rel="stylesheet"
      type = "text/css"
      media="screen"
      href="screen.css"
>
<link rel="stylesheet"
      type = "text/css"
      media="handheld"
      href="mobile.css"
>
```

By separating different media types, the adaptations can be implemented by using CSS stylesheets. With this method it is absolutely essential that document content and structure are strictly separated.

The second method that we we have implemented for the adaptation of the webshop for use with mobile devices is based on the WALL/WURFL technology as described in chapter 4.3. Both methods turned out to work well, although CSS 2.0 is not widely supported by mobile phones, especially if they implement the WAP 1.x standard. WALL has the great advantage of adapting the presentation automatically to the respective devices, taking into account the device rendering format (CHTML, XHTML, WML, or XHTML MP). The CSS mechanism, in contrast, is much more restrictive, as it doesn't change the format of the website (which is always XHTML) and the website will therefore only be partly displayed if a device supports XHTML MP or CHTML. A more detailed overview of the implementation of the presentation layer for the eSarine webshop is given in chapter 8.

Chapter 6

Personalization

Personalization is tailoring specifically to one individual. It is the adaptation of a consumer product, electronic or written medium, based on personal details or characteristics provided by a user or consumer.

Wikipedia¹

On the World Wide Web, personalization means adapting web pages based on the preferences of an individual visitor. Personalization implies that the changes are based on implicit data, such as device type or page views. Instead, a different term, customization, is often used when the web site only takes into account explicit data such as user ratings or user-defined preferences. Therefore, personalization as used in the context of this thesis, is referred to as the process of adapting a web site to meet the needs of each respective user.

Different approaches to personalization have been studied in the past. Some of these techniques will be presented in the next section. For a general overview of different web personalization strategies, see [16]. Our approach to personalization consists of adapting the web site's markup to mobile phones and using Semantic Web technologies to store and retrieve user preferences in the context of the e-Sarine web shop. We will show how the Semantic Web mechanisms can be used for web personalization, thus facilitating the process considerably.

An ideal personalization mechanism should know in advance what a user wants. Obviously, it is not possible to predict the user's behaviour, but there are several methods that allow to improve the user acceptance of a web site. Even though personalization is widely used in the domain of web site personalization for desktop-oriented web sites, we argue that personalization is even more important when it comes to producing web sites for mobile device clients. On a mobile device, the space available for presenting information is very restricted. This means that the web site should allow the user to easily locate the information they want. If browsing requires too much time for searching and loading, the user will most likely refrain from visiting again. Therefore, the mark-up of the web site as well as the content have to be provided in a way that makes browsing attractive, or at least as straightforward as possible. In this chapter we are going to present an architecture that attempts to combine two different aspects of

¹<http://en.wikipedia.org/wiki/Personalization>, retrieved in October 2004

personalization: adaptation to a specific device, as discussed in the last chapter 5, and personalization of the content.

6.1 Semantic Web

The main idea of the Semantic Web [8] is that machines should be able to understand, process and reason about resources in order to improve the support for human interaction with the World Wide Web. In this context, the question of personalizing the process of interaction with web content logically arises. Research in the domain of adaptive hyper media has been trying to understand how personalization and adaptation strategies can be applied in hypertext systems. The idea of combining Semantic Web technology and adaptive hyper media techniques [12] could therefore lead to powerful architectures that can exploit the strengths of both research areas. Semantic Web techniques would allow to organize the content of a web site² in a machine-readable way, thus facilitating the classification and clustering of the content that would be explicitly defined by the semantics. In addition, the idea of adding semantics to resources could also be used for the creation of user-specific semantics³, representing a user's interests and preferences. This approach has been adopted by the GraniteNights project, an implementation of an evening scheduler in the city of Aberdeen, that has been developed by Grimnes et al. at the University of Aberdeen by [20]. Instead of using traditional user profiles, the profiles are stored as RDF documents based on the same ontology which has been employed throughout the web site's content. This allows to express semantic information in the user profile with the advantage of simplifying the matching between the given semantics of a web site and the user profiles, as simple, graph-based matching techniques can be adopted for this purpose.

The approach to personalization that we are going to present in this chapter is largely based on the ideas described by Grimnes et al. [20]. We are going to use Semantic Web methods to implement the personalization module, using RDF as the format of choice for user profiles.

6.2 Content Personalization Techniques

Although explicit personalization techniques, i.e. customization, are widely used they don't lead to satisfactory results in most cases. This is mainly due to the fact that user interests might change over time and during specific periods. For example, a user who is generally not interested in sports might still follow the 100 meter sprint at the Olympics. Therefore, the system should be flexible enough to adapt in these situations. This is where implicit personalization comes in. It allows the automatic adaptation of a user's interests without requiring the user's active participation. The adaptation strategy is based on the idea of positive and negative feedback, i.e. the more time a user spends on a web page, the more interest he might show in its content. Furthermore, the transition from non-personalized to personalized content should be smooth without presenting random content to a user.

²which actually is a hypertext system, i.e. a number of resources connected by links

³Semantics here refer to the hierarchy of interests of a user

6.2.1 Personalization Strategies: An Overview

A widely used approach to personalization is the so-called 'usage-based' personalization method. The core idea of this approach consists of a pattern discovery algorithm that compares the profiles of all the subscribed users in order to derive browsing patterns and subsequently provide each user with information that could be interesting to them. This approach has been adopted by several web sites, including the recommender system of Amazon.com⁴. The idea that lies behind this approach is straightforward: users with similar browsing behavior are almost certainly interested in similar topics.

The main disadvantage of user-based personalization is quite obvious: a large user-base has to be available before the personalization mechanism can work successfully. A conflicting situation occurs if a new item is added to the information pool. It could not be recommended to a potentially interested person as it has not yet been discovered by a user. One solution to the 'new item problem', as it is referred to in the literature, could consist in performing a text-based comparison in a preprocessing phase before the item is being added in which the text is compared to other already existing entries and therefore a kind of 'classification' can be derived.

A general approach to personalization will normally consist of the following modules [16]:

- User profiling: The process of gathering information specific to each visitor. This can be done explicitly or implicitly. The gained information is used to customize the content and the structure of a Web site to the clients specific interests.
- Log analysis and Web usage mining: The analysis of logs is performed to discover interesting usage patterns. The extraction of user-related information concerning the browsing behavior is part of the user profiling process.
- Content management: The process of classifying the website's content in semantic categories to make information retrieval easier. Ideally this categorization should be organized in a machine-readable way.
- Web site publishing: Presentation of the content to the end-user. Different technologies are available for publishing web content.

Usage-based personalization methods would therefore try to classify the web site's content into "user-interest clusters", possibly combined with an additional classification mechanism for new items. Although this solution has been very popular in the past, it lacks generality and does not work well for web sites with a small user base. In order to improve this method, we propose to add additional metadata to web resources, which allows us to address both problems at once, the 'new item problem' and the problem of requiring a large user base. We believe that a combination of usage-based approaches and Semantic Web technologies would provide a solid base for a personalization architecture. The Semantic Web propagates the use of metadata to describe web resources and the relations among them, it is therefore an ideal candidate to meet classification requirements of web shop items and all other resources that are related to it. The fact that the metadata is provided in a machine-readable format facilitates

⁴<http://www.amazon.com>

automated processing considerably.

In our implementation we use RDF as the format of choice for the user profiles and provide an ontology in OWL. This allows us to implement a simple matching mechanism that compares the profiles to the ontology. Each user profile therefore represents its own view of the web sites' content. Additionally, as mentioned before, we would not have to deal with categorization and inter-user pattern comparison as the user profile as well as the semantics of the content are given by a graph⁵. It is not necessary to provide any other means for the classification of the content as this is already given by the combination of an ontology and the RDF metadata. The web site publishing will consequently be done dynamically based on the user profile a markup to meet the requirements of different device profiles.

6.3 Web Data

An important aspect of web personalization is the collection of web data which are the foundation of any personalization mechanism. Eirinaki et al. [16] describe four different categories of web data :

- *Content data* can be simple text, images, or structured data, such as information retrieved from databases.
- *Structure data* can be data entities used within a Web page, such as HTML or XML tags, or other structural information like hyperlinks or metadata.
- *Usage data* are data which are collected in order to gain insights into the usage of a website, such as a surfer's IP address, time and date of access, pages accessed, and any attributes that can be included in a Web access log. Depending on the profiling technique, i.e. implicit versus explicit profiling, this data will be used to build up or adapt the user profiles. If implicit profiling is used, it is important to use a mechanism to identify the repective users.
- *User profile data* contain information about users interests and preferences and might also include demographic information (name, age, country, marital status, education, interests etc.). This information can either be acquired through registration forms or questionnaires, or can be inferred by analyzing Web usage logs.

Web usage data can be collected at the server-side, client-side, on proxy servers, or obtained from databases. Most of the data comes from the server log files. Every time a user requests a website, the web server adds an entry to the log file. These records are stored in a format known as the Common Log File format (CLF), which is W3C standard. The most useful fields of a CLF record are the IP address of the host computer requesting a page, the HTTP request method, and the time of the transaction:

```
127.0.0.1 - - [28/Jul/2005:17:29:02 +0700] "GET  
http://localhost/webshop/ HTTP/1.1" 404 280
```

⁵a weighted graph in the case of user profiles

Although server log files contain many useful information information, and data is stored at a very detailed level, but their size may be extremely large. Another problem with server log files is the information loss that occurs through caching. To improve performance many web browsers cache pages on the users computer for fast retrieval. As a result when a user returns to a previously requested page, the cached page is displayed, leaving no trace in the server log file.

6.3.1 User Identification

Server log data contains information about all users visiting a website. To associate the data with a particular user, a user identification mechanism is necessary. The simplest form of user identification is registration. It has the advantage of being able to collect rich demographic information through questionnaires. However, due to privacy concerns, some users might decide not to browse web pages that require registration, or users may provide wrong personal information.

A different method for user identification is based on log file analysis. Log-based user identification is performed by partitioning the server log into a set of requests that have been made by a single IP address. Accurate server log partitioning might fail in cases of changing IP addresses, or missing referencea due to local or proxy server caching.

Another technique for user identification is based on software agents in the browsers, which send back data. Given that users are very concerned about their privacy, these agents are generally very likely to be rejected by users.

The most reliable mechanisms for automatic user identification are based on cookies. Whenever a browser contacts a website, it will automatically return all cookies associated with this website. Cookie-based user identification is only reliable if the user launches each URL request from the same browser.

6.3.2 Sessions

A user session consists of all requests of a user during a single visit to a website. Since a user may visit a website more than once, a server log may contain multiple sessions for a given user. Automatic session identification can be performed by partitioning log entries belonging to a single user into sequences of entries corresponding to different visits of the same user. Sessions are generally identified by session id's which are conveyed through cookies and are included in each subsequent request of a user.

6.3.3 Clickstream

Clickstream analysis is a special type of web usage mining and constitutes an good way to understanding users behavior. The concept of clickstream usually refers to a visitors path through a website. It contains the sequence of actions entered as mouse clicks, keystrokes, and server responses as the visitor navigates through a website. Clickstream data can be obtained from several sources such as web server log files, commerce server database, or from client-side tracking applications. Most efforts in web usage analysis are focused on discovering users access patterns. Understanding users navigation through a website can help provide customized content and structure tailored to their individual needs. The web



Figure 6.1: The eSarine recommender system in action: Three so-called quick-links to the users' favorite categories are provided on the mobile web pages.

usage mining mechanism that has been implemented for this thesis is based on a simplified clickstream mechanism that keeps track of the items and categories viewed by a user. This data is subsequently used to produce the interest factor in the RDF profiles:

```
<rdf:Description rdf:about="http://category.esarine.net/category#10">
  <j.0:categoryWeight>3</user:categoryWeight>
  <user:lastAccessed>14.08.2005</user:lastAccessed>
  <user:categoryId>10</user:categoryId>
  <user:interestFactor>3.7437418</user:interestFactor>
</rdf:Description>
```

Every time when the user accesses the resource <http://category.esarine.net/category#10>, the categoryWeight is updated (categoryWeight = categoryWeight + 1) and the interestFactor is computed according to the following equation:

$$\text{interestFactor} = [\text{weight}] / [\text{days since last accessed}]$$

Of course, this equation is very simple, and other factors could be taken into account like for example the viewing time of an item or category. A recommender system could subsequently be based on this interestFactors in order to recommend items that belong to categories that were of special interest for a user. A different approach could consist in providing so-called quick-links (direct links) to favorite categories.

6.4 Personalization and the Semantic Web

Recently, the emergence of the Semantic Web as a vision for the next generation of Web technologies has provided the context for revisiting personalization from an analytical (knowledge-based) rather than an empirical view.

Edwards et al. in [15]

The Semantic Web is a vision in which web resources will be enriched with machine-readable metadata, using a single markup format. At a first thought this idea seems very interesting in the context

of new emerging technologies and standards. But a question that naturally arises when thinking about the Semantic Web and its implication is: What are metadata? Which information should be machine-readable? What is relevant and what not?

We are not going to be able to answer all the questions mentioned above as the Semantic Web is still mainly a vision with very few implementations available. But let us get back to the first question about metadata. Metadata⁶ is information about a resource that is not part of the resource's content. In the context of the semantic web metadata are exploited to capture the semantics of the source. When speaking about semantics, we generally refer to meaning, or the actual content, of a resource. Obviously the meaning of a text as such cannot be rendered machine readable. But when different resources are available, the relationships among resources could be conveyed in a machine-readable format. Think of UML⁷-based representations of Java Class Hierarchies for example, or of table-relations in relational data base systems. This kind of information is machine-readable as a matter of fact.

Once metadata is machine-readable, who will actually read this data? In fact, in the vision of the Semantic Web, software agents will be roaming around the web, collecting metadata information for specific purposes and these agents are supposed to be able to "reason" about metadata. Before we continue, let us consider a simple example of metadata. This thesis, for example, is a structured text consisting of the following elements:

- a title: Adaptation of a Webshop for Mobile Devices
- a number of sections and subsections, each with another title
- a number of figures to illustrate the content of the text
- the representation (or mark-up) of the thesis

In Semantic Web terms, this thesis is a resource with the properties "title", "sections", "figures", and "markup". These properties again point to resources. A "section" for example, is a collection of "subsections", and these "subsection" may point to "subsubsections". Mathematically, these statements can be modelled with:

- Resources: everything is a resource⁸
- Properties: directed relations between resources
- Statements: two resources connected by a property

This is where machine-readability comes in. In order for statements to be machine-readable, they have to be conveyed in a standardized format, which is RDF⁹. RDF is nothing more than a general model

⁶A definition of metadata that is very often used is: "Metadata is data about data".

⁷Unified Modelling Language

⁸This view might be confusing, but in fact everything that we can think about, every object, every abstract idea is a resource.

⁹Resource Description Framework

for statements of the kind mentioned above. Different mark-ups for RDF exist, as RDF is just a framework, but the most widely used is the RDF/XML standard which is the official W3C¹⁰ standard for RDF.

```
<rdf:Description rdf:about="http://category.esarine.net/category#1">
  <user:interestFactor>1.1329781</user:interestFactor>
  <user:categoryWeight>1</user:categoryWeight>
  <user:lastAccessed>10.08.2005</user:lastAccessed>
  <user:categoryId>1</user:categoryId>
</rdf:Description>
```

This RDF/XML fragment has been extracted from the user profile as we are using it in the e-Sarine webshop. It represents the description of the resource <http://category.esarine.net/category#1> which has the properties `interestFactor`, `categoryWeight`, `categoryId`, and `lastAccessed`. The XML representation allows us to parse the description with existing XML-parsers such as DOM or SAX.

6.4.1 Introducing Ontologies

Ontology is the study of what there is, an inventory of what exists.¹¹

An ontology is typically a hierarchical data structure containing all the relevant entities and their relationships and rules within a given domain. Let us consider the following ontology class expressed in OWL¹², an XML-based markup language for publishing and sharing data using ontologies on the Internet:

```
<owl:Class rdf:about="http://category.esarine.net/category#7">
  <eSarine:categoryName>Hard Rock</eSarine:categoryName>
  <eSarine:categoryId>7</eSarine:categoryId>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://category.esarine.net/category#4"/>
  </rdfs:subClassOf>
</owl:Class>
```

Each OWL-class has a URI¹³ and a set of optional properties like `<eSarine:categoryName>`, `<eSarine:categoryId>`, and `<rdfs:subClassOf>`. Although this example is really simple, it permits to show where the actual power of OWL lies: in the setup of a hierarchy. Our class, 'Hard Rock' is a subclass of <http://category.esarine.net/category#4>, the 'Music' class:

```
<owl:Class rdf:about="http://category.esarine.net/category#4">
  <eSarine:categoryName>Music</eSarine:categoryName>
  <eSarine:categoryId>4</eSarine:categoryId>
</owl:Class>
```

The hierarchy of the items of a simple webshop could therefore be organized as follows (the indention shows the hierarchy):

¹⁰www.w3c.org

¹¹www.artsci.wustl.edu/~philos/MindDict/0.html

¹²Web Ontology Language

¹³Unique Resource Identifier

```
Class http://category.esarine.net/category#4
  Class http://category.esarine.net/category#5
    Class http://category.esarine.net/category#8
    Class http://category.esarine.net/category#13
  Class http://category.esarine.net/category#7
    Class http://category.esarine.net/category#12
    Class http://category.esarine.net/category#11
  Class http://category.esarine.net/category#6
    Class http://category.esarine.net/category#9
    Class http://category.esarine.net/category#10
Class http://category.esarine.net/category#1
  Class http://category.esarine.net/category#3
  Class http://category.esarine.net/category#2
```

Of course, OWL offers much more than just hierarchies such as property restrictions and data types. For an overview of OWL see [1]. Notice that for the purpose of our architecture the expression of class hierarchies is sufficient to build a semantically aware personalization mechanism.

The combination of an ontology of the webshop products and the RDF user profiles, allows us to analyze user interests in the context of an object hierarchy.

6.5 User Profiles

Personalization generally requires information about the user's preferences, interests, goals, and expectations. The collection of this information that uniquely describes a specific user is called a user profile. Adomavicius and Tuzhilin [3] differentiate between two main components of a user profile: behavioural and factual. The term factual component refers to demographic and transactional information such as age, educational level, address, etc. The behavioural component contains information about the on-line activities of the customer. It is usually stored in different formats such as logic-based descriptions, classification rules, or attribute-value pairs.

We have chosen RDF as the format in which the user profiles are being stored, thus providing a way of describing the user's individual interests and their semantics. This implies that the user profiles are actually a weighted graph representation of a semantic domain which consequently allow a more appropriate matching of user profiles and web shop semantics. The problem can be solved by a simple graph-matching algorithm.

An important aspect of user profiling is the question where the profiles are being stored. It would be quite interesting to save the profiles on the client side, i.e. the user's mobile devices or desktop computers in order to avoid many upcoming privacy issues. However, at the time of writing, most mobile devices only have a limited memory capacity and users would not want large profiles to be stored on their devices. We have therefore decided to store the user profiles on the server-side. But we are aware of the privacy issues that might arise in this context. Nevertheless, we believe that the personalization mechanism is much more efficient with the user profiles directly accessible to the programmes that reside on the server.

6.5.1 RDF User Profiles

In this thesis we are presenting an architecture based on user profiles that are stored in RDF/XML format. User profiles can be stored in any kind of format as long as they remain machine-readable (and human-readable in some cases). But we argue that by using RDF for our profiles, we can improve the quality of the information that is provided by the profiles.

As we have already seen before, RDF is a mechanism to describe statements, i.e. resources and their properties. Before considering RDF/XML as the format of choice for user profiles, we should first discuss what we want our user profile to do, and which information we want to store. Generally, user profiles are used to store user-relevant information in order to:

- identify users
- personalize contents

Generally user profiles retain information such as name, password, preferences, and other user-related information. Traditionally, this information is stored in a data base. Relational data bases are very well suited to retain this kind of information. But in general, they are poorly suited for storing dynamic information such as user preferences, depending on their nature and size. When keeping user interest values in a data base, most systems implement some kind of text- or file-based¹⁴ mechanism. This is where standards come in: instead of reinventing a new format and the appropriate methods for parsing and processing these format for each application, a standardized, such as XML format could be used which would facilitate the processing considerably.

But RDF/XML is more than just a format. It enables us to integrate the different requirements we have stated above. We can take advantage of the XML-format, use the structure to convey semantical information such as the interest of a user in a specific category or item, and combine RDF with the ontology of our webshop items. Therefore, RDF turns out to be an ideal candidate for user profiles. Edwards et al. [15] propose a similar approach to RDF, arguing that RDF can be viewed as a simple 1st order logical system. In their research, Edwards et al. produce interesting results, using an RDF-based search model instead of complex machine learning algorithms.

6.6 Explicit versus implicit profiling

Whenever user profiles are introduced to a web application, the question arises from which source the profiling data should be collected. Generally, two different approaches to data acquisition are mentioned in the literature: implicit and explicit profiling mechanisms. Explicit profiling requires the user's active participation and allows the users to actively control their personal profile content. Users may be requested to fill out a form for registration which they can edit whenever they want in order to change their personal information and to add or remove preferences. With this method the clients can

¹⁴or BLOB-based

inform a website directly about their personal preferences. With explicit profiling the user can control the information that is available to a website provider which eliminates privacy issues that can occur with implicit profiling.

Implicit profiling, in contrast, does not require the user's input and is therefore performed in the background. As an example, Amazon¹⁵ keeps track of the customer's purchasing history and recommends items based on this history. But implicit profiling might go much further. It usually implies monitoring user behaviour to find, analyze, and identify browsing patterns. In many cases, the process of user tracking is done without the users' consent, meaning that the user is not directly aware of it and mostly not being informed. Implicit data is collected for different reasons:

- user profiling and personalization
- statistical evaluations
- third party interests
- personalized advertisements

Implicit data can be collected on the client or on the server side. Server-side data include automatically generated data in server access logs. Client-side data include cookies, mouse or keyboard tracking. Implicit profiling does not require that the user provides personal information. Although implicit profiling seems less intrusive, it raises many privacy issues. As internet users become increasingly concerned about what information is being gathered when they visit a website, it is very important for a website to inform the user if an implicit profiling mechanism is running in the background. It is also necessary to explain what the gathered information is being used for and with whom it is going to be shared. Ideally, the user should be able to switch off the monitoring process if he does not wish to be tracked¹⁶. In most cases, it is therefore advisable to either inform the user about the tracking process or to replace implicit profiling with explicit profiling methods.

6.6.1 P3P: Platform for Privacy Preferences

P3P¹⁷ is a W3C recommendation that presents an infrastructure for the privacy of data interchange. The standard enables website owners to describe their privacy practices in a standardized format which user agents can automatically retrieve and interpret. This simplifies the process of reading and accessing the website's privacy policy for the users. P3P enables machine-readable privacy policies that can be automatically retrieved by web browsers¹⁸ and other user agent tools. Some of these tools are even capable of comparing each policy against the user's privacy preferences and help the user in deciding when to exchange data with web sites. However, there is no way to provide a method for ensuring that sites actually act according to their policies.

¹⁵<http://www.amazon.com>

¹⁶Many servers use cookies to identify users, therefore the user might delete cookies or disable sessions if he doesn't want to be traced back. But generally the providers should assume responsibility for user privacy.

¹⁷Platform for Privacy Preference, see <http://www.w3.org/P3P/>

¹⁸P3P is supported since Internet Explorer 6 and Netscape Navigator 7

Chapter 7

Mobile Payment

Despite all the extensive work accomplished for the development of electronic payment, mobile payment still remains a relatively young field. Mobile payment is believed to be a very interesting service which has been predicted some years ago by a number of research institutions [4] and researchers [14] to become a killer application for medium-sized and small businesses. They believed that it will be capable of generating a huge revenues and reflate the mobile business industry. Therefore, financial institutions, mobile operators, start-ups and technology providers have conducted a variety of standardization and commercial efforts [26]. Nowadays, however, the situation does not seem as good as predicted and the reality of the market shows that numerous issues, such as market uncertainties, business models, and technical uncertainties, need to be solved before a large scale realization can be considered.

Considerable research efforts have been undertaken in order to better comprehend these key aspects of mobile payment. In [31] a multi-perspective Analysis of the mobile payment environment is proposed. Ondrus et al. [31] conclude that the success of mobile payment cannot be predicted by only considering one single dimension. "The immaturity of the market and the consequent unresolved technical, strategic and demand issues make the adoption of mobile payment mechanisms highly uncertain", as they say in their paper. Although many mobile payment solutions are readily available, most are unsuccessful given the fact that they fail to provide the right value proposition to customers. Therefore, the wishes and needs of the customers have to be considered before implementing such a solution. Some research has been conducted in order to discover the most important adoption factors for consumers [42].

7.1 Defining Mobile Payment

Mobile payment is the process of two parties exchanging financial value using a mobile device in return for goods or services. A mobile device defines a wireless communication device, including mobile phones, PDA's, wireless tablets, and mobile computers.

The Mobile Payment Forum in [17]

Different definitions for mobile payment have been proposed (see [23] for an overview). While some authors view mobile payment as a subset of electronic payments, others, such as the Mobile Payment Forum¹ argue that mobile payment should be regarded as a different, independent payment scheme and that "mobile commerce should not be considered as a simple extension of e-commerce on a mobile device" [17]. The Mobile Payment Forum believes that only solutions that are based on characteristics which are unique to the mobile environment will be successful. The Mobile Payment Forum is a global, cross-industry organization that aims at defining technological standards and functional declarations for the mobile payment market.

7.2 Existing Payment Solutions

Different market segments can have different needs, such as simplified payments and reduced transaction cost. This might be interesting for consumers who want to pay small purchases immediately. Other customers might have a demand for more security in large transactions. Consumers in this group would consider mobile payment transactions as a replacement for credit cards.

Paybox² and Mobipay³ propose phone-based payment schemes as a convenient way to pay at e-shops and mobile merchants. However, the convenience and usefulness of these services have to compete with alternatives such as cash or credit cards. Thus, it is not surprising that these solutions failed to attract many subscribers so far.

There seems to be a demand for mobile payment services in low-cost transactions, such as transportation costs and cinema tickets. Mobile payments schemes such as E-ZPass⁴ and Octopus [37] were quite successful by providing a solution to specific customer needs. E-ZPass is a North American wireless toll collection system that considerably improved the convenience of paying tolls without queuing. Octopus, a card-based payment service was originally invented for public transportation in Hong Kong, and provides a convenient way to pay for transportation fares and items in grocery stores.

7.3 Micropayments and Macropayments

Another issue that regularly comes up in the mobile commerce discussion is the amount of money that is being transferred in a transaction. Pico- and micropayments⁵ are payment schemes for very small charges, and only attractive if the volume of transactions is high, given that the profit margin is usually very small. In the context of mobile payments, small-charge-transactions are mainly used for buying ring-tones or images as well as paying for small purchases from vending machines⁶. Macropayments⁷, in contrast, are conducted for purchases with a higher margin and are therefore more profitable. However, there are stricter security requirements in order to minimize the risk of fraud for financial transactions.

¹<http://www.mobilepaymentforum.org>

²<http://www.paybox.net>

³<http://www.mobipay.com>

⁴ <http://www.mta.nyc.ny.us/bandt/ezintro.htm>

⁵Picopayments are payments in a range of less than 2 swiss francs, micropayments less than 15 swiss francs

⁶Like the Swisscom mobile payment service for example.

⁷For purchases above 15 swiss francs

For everyday use, mobile payment schemes seem more interesting for payment of small amounts as the customer confidence level for macropayments over mobile devices is still relatively low.

7.4 Mobile Payment Security

Security is the main factor that has slowed down the m-commerce development in the past. Security not only includes the actual security measures provided by the payment services, but also the confidence of potential customers. Customers concerns are mainly relating to security, privacy, and ease. Winning the customer confidence is therefore crucial for the development of a successful mobile payment scheme. Or, as the Mobile Payment Forum expresses it in [17]: "The challenge for the mobile payment industry is to convince the conservative majority of customers to embrace mobile payments".

7.4.1 New Generation Mobile Networks

The new generation mobile networks, or 3G networks as they are called in the literature, will allow considerably increased data traffic. With the deployment of these technologies such as EDGE, UMTS, and CDMA it becomes possible to develop mobile payment solutions independent of the underlying wireless technology, provided that internet and security standards are consistently implemented and supported.

7.4.2 WTLS: The WAP Security Layer

The WAP Transaction Layer Security, WTLS is a secure protocol layer similar to the Secure Session Layer (SSL) and Transaction Layer Security (TLS) protocols. It is session-based and allows for client and server to recalculate encryption key information from a given sequence number. There are three levels of WTLS secure sessions ⁸:

- Level 1: anonymous encryption, neither client nor server are authenticated.
- Level 2: support for server certificates in which clients authenticate the server.
- Level 3: support for client certificates and server authentication The WTLS certificate is specified to reduce information transfer.

However, WTLS has several major drawbacks⁹. WTLS allows only for weak encryption algorithms, and with some WAP clients WTLS encryption can be manually disabled. Newer versions of the WAP Security Layer may address these problems.

⁸Information about WTLS is scarce, a good introduction is given on: <http://www.advisor.com/Articles.nsf/aid/MIKAP001>, last consulted in June 2005

⁹Markku-Juhani Saarinen describes in "Attacks against the WAP WTLS Protocol" how easily the WTLS protocol can be attacked, see <http://www.freeprotocols.org/harmOfWap/wtls.pdf>, last consulted in June 2005

7.5 Need for Standards

Standards aim at facilitating the introduction of new technologies by providing a set of interfaces and restrictions that all products implementing these technologies should share. For mobile payment market standards would bring huge benefits if the following considerations are being respected [23]:

- the design of commons interface instead of many incompatible ones
- centralization and sharing of infrastructure
- implementation of special expertise at a common level

At present, too many different solutions are available, making it increasingly difficult for merchants and customers to come to a choice. With standardized technologies, the promotion and realization for the adoption of mobile commerce services would be facilitated considerably. Experts emphasize that standardization efforts should aim towards coordination and interoperability among industries and technologies.

7.6 Mobile Payment Business Models

Although many leading companies already offer mobile payment for digital content downloads, the real breakthrough of mobile payment has not yet taken place. The main players of this process are expected to be network operators, financial organizations, and third-party billing providers. At the time of writing, most payments in mobile transactions are transfers in the range of micropayments. Nevertheless, experts believe that by the end of 2007 mobile payments will amount to a significant average return per user.

Digital content such as ringtones, wallpapers, and logos are commonly payed by SMS or MMS, i.e. by charging the customers' phone bills. Amazon.com started offering mobile payment solutions in 2000 when they introduced HDML¹⁰ to their webshop. The customer is provided with a server-based wallet that enables him to purchase goods online. This solution might work for a big online market player like Amazon, but has the drawback that the merchant, Amazon, hosts the sensitive payment data, and not a trusted third-party or a credit-card company.

7.7 Simpay, Paybox, and Paypal

Simpay¹¹ is a mobile payment company that was set up by Orange, Vodafone, T-Mobile, and Telefonica Moviles with the aim of implementing a solution for micropayments. The idea was to realize a Europe-wide payment standard. Unfortunately, the project failed in June 2005¹². Nevertheless, Paybox announced that they will carry on in order to become a European leader in mobile payment solutions. Paybox has been founded in 1999 and was the first to offer of a worldwide mobile payment scheme.

¹⁰Handheld Device Markup Language, a predecessor of WAP.

¹¹see <http://www.thefeaturearchives.com/47374.html>, last consulted in June 2005

¹²see the Kaywa mobile payment weblog: <http://mobile.kaywa.com/mobile-payment/simpay.html>, last consulted in June 2005

Paybox is today now widely accepted in Austria. However, it is only open to Austrian Mobile customers and all members of a transaction need an account. For a transaction, the user has to respond a call that prompts for a PIN. Paybox charges bank accounts only and does not offer credit-card payments.

Another mobile payment solution is offered by Paypal¹³, an e-payment company that has been developed for payments over the web. Paypal offers users an electronic server-side wallet in which payment information such as credit card number or bank account information are saved on the Paypal server. Instead of using the credit card number to issue a payment, customers use their Paypal account information to send money to merchants or peers by email.

A completely different approach is a wallet-based solution in which sensitive information is stored on the clients' mobile device. Nokia¹⁴ for example offers a Nokia Wallet technology. It is based on the Electronic Commerce Modeling Language (ECML) standard and is already widely used in Scandinavian banks. The Nokia Wallet mainly implements an encrypted storage place on the mobile phone.

7.8 A Mobile Payment Scheme for e-Sarine

As stated above, most implementations available for the mobile payment sector are still very immature. It is very difficult to predict the direction of the future development in this field. Of course, a webshop such as e-Sarine could implement a similar method to that of Amazon.com, with a server-based wallet that can be used for payments from desktop computers and mobile phones. But most customers today are not willing to give away sensitive information to merchants, especially if they are not very well established.

Paybox, although very successful in Austria, is only occasionally used in Switzerland. And the Nokia Mobile Wallet is restricted to Nokia devices only. We therefore propose e-Sarine users to use Paypal for purchases from mobile phones. Paypal can be easily accessed from a mobile phone if the user already has an ac

¹³<http://www.paypal.com>

¹⁴<http://www.nokia.com/>

Chapter 8

Implementation Details

In this section we are going to present some details of our implementation. It is important to note that many different implementations would have been possible and that we were forced to make our choices regarding the technologies involved and the algorithms we have proposed. In this chapter we are not going to speak about why we have decided to use one method or another as this has been done in previous chapters. We are rather going to show how the technologies and methodologies were integrated for with the existing webshop framework. The architecture of the eSarine implementation is based on the technologies as presented in figure 8.5. The architecture of the presentation layer is organized as a set of three Tiles skins, each of which is tailored towards the rendering to a different kind of device. For our implementation we distinguish between three different device types: desktop computers, XHTML-enabled mobile devices such as PDA's and new-generation mobile phones, and mobile devices that support CHTML, XHTML-MP, or WML. For the implementation of the CHTML, XHTML-MP, and WML support, we have used the WALL tag-library.

The business logic is implemented in Jakarta Struts that allows to make the bridge between the presentation layer and the actual business logic that is implemented as Java Beans and Struts Actions. The webshop data is stored in a PostgreSQL database that is accessed through the Struts business logic. The personalization module is implemented on top of the JENA API, a Java-based Semantic Web framework that provided all the necessary methods for the creation of the RDF user profiles and the webshop ontology.

8.1 Presentation Layer

The presentation layer of the eSarine webshop consists of three different Tiles-style skins, each with an own set of JSP files to display the content of the webshop. As eSarine implements a very strict separation of presentation and content (the content of the website is stored in a database), it was not necessary to change much of the Java code and to reproduce the content in order to change the layout of the website. In the introduction we have considered different methods for the implementation of the presentation layer. Let us just recapitulate these propositions in order to explain our choice of using Tiles



Figure 8.1: An overview of the technologies involved in the development of the eSarine webshop.

skins:

1. Implementation of a separate skin for mobile browsers with an additional personalization layer.
2. Implementation with dynamic generation of CSS style sheets.
3. Implementation of a declarative mechanism via XML which allows the dynamic generation of web pages, depending on the properties of the browser and possibly the user preferences.

In fact, we have produced a combination of points one and three. Point two would have been very interesting, especially when taking into account that it could have been realized without much effort. However, as we have seen in chapters 4 and 5 of this thesis, CSS 2.0 and media type support is not widely available for mobile devices which would restrict the graphical adaptation of the website to devices that are actually capable of displaying XHTML 1.0. But what we want is to adapt the website for use with any kind of web-enabled device which therefore excludes an implementation based on CSS 2.0 from our propositions.

In point one of our list we propose to implement a separate Tiles-style skin for use with mobile devices and this is exactly what we have done. Unfortunately, the switching of one skin to another cannot be implemented automatically, i.e. when a user requests the website from a mobile phone we choose the appropriate skin based on the request header information and build up the response according to the chosen skin. The eSarine webshop has implemented Tiles skins for a different purpose: to provide a (desktop) user with a personalized look-and-feel (colors, fonts, etc). Therefore, a favorite skin is attributed to every user and stored in the database. As we were careful not to interfere with the given webshop architecture, we decided to implement a different approach. The user has to request a different website according to the device capabilities, that is, a user with an XHTML-enabled phone could choose to either

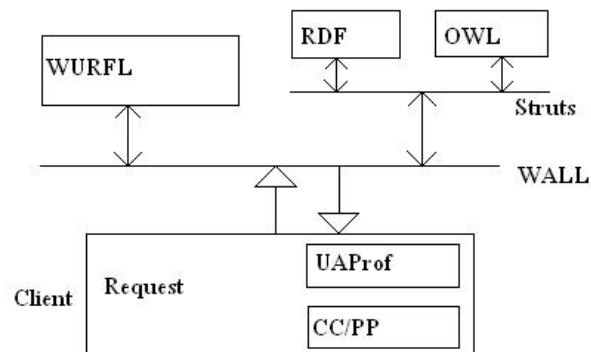


Figure 8.2: The structure of a request: A client sends a link to the CC/PP or UAProf profiles in the header, the WALL implementation subsequently looks up the profile in the WURFL database, retrieves the RDF user profile and the OWL ontology in order to create a personalized response.

use the WALL-based¹ or the CSS-based² version of the mobile website, whereas a WML-enabled phone could only use the WALL version. This mechanism allows us to implement the change of the skin based on the request rather than the implicit information given in the headers and the webshop architecture remains intact.

We have also mentioned above that we actually propose a combination of skins and a declarative XML-based method. The XML-files that are involved in the whole personalization and presentation architecture are visualized in figure 8.2. The personalization mechanism depends on the RDF of the user profiles and the OWL webshop ontology, the mobile customization method or personalization layer is based on the CC/PP or UAProf as well as the WURFL database which is also in XML format. The WALL tag library plays the role of a connector between the device capability profiles³ and the WURFL database. Thus, WALL retrieves the device information from the request and searches the WURFL database for the device profiles in order to convert the requested webpage into the appropriate format for display on the respective devices.

8.1.1 Adapting the Markup

The adaptation of the markup for mobile devices is based on the definition of two new skins that we add to the Tiles definition file: tiles-defs.xml.

```
<definition name=".skin.mobile"
```

¹url<http://www.esarine.ch/mobile>, note this is not a proper link, just an example, but the extension '/mobile' is as implemented.

²<http://www.esarine.ch/wml>

³UAProf and CC/PP are device capability profiles, see chapter 4 for details.



Figure 8.3: The mobile XHTML skin on the Openwave 7 browser.

```

path="/mobile/pages/storefront/mobile/general/layout.jsp">
  <put name="header"
    value="/mobile/pages/storefront/mobile/general/header.jsp" />
  <put name="search"
    value="/mobile/pages/storefront/mobile/general/search.jsp" />
  <put name="menu"
    value="/mobile/pages/storefront/mobile/general/menu.jsp" />
  <put name="categories"
    value="/mobile/pages/storefront/mobile/general/categories.jsp" />
  <put name="path"
    value="/mobile/pages/storefront/mobile/general/path.jsp" />
  <put name="footer"
    value="/mobile/pages/storefront/mobile/general/footer.jsp" />
  <put name="logo"
    value="/webshop/mobile/pages/storefront/mobile/general/logo.gif" />
  <put name="css"
    value="/webshop/mobile/pages/storefront/mobile/general/layout.css" />
  ...

```

The first skin, `skin.mobile` is an adaptation of the already existing webshop in XHTML with a specific media-type based CSS file. In fact, the normal webshop base-skin should do. But in order eliminate unnecessary markup and white-spaces, we have decided to add this optimized mobile version to the webshop. Given that we have not yet implemented a mechanism to automatically redirect to the skin that best suits a device specification, it is necessary to access the webshop through a different URL in order to get to this optimized XHTML skin. This mobile version is accessed through: `pathtowebshop/mobile`. Of course we have also added the same mobile CSS⁴ to the eSarine base-skin by using media-types. But this skin does not provide the optimized version of the mobile pages, but a mobile-enabled webshop that is based on the same layout as the desktop version.

The second skin, `skin.wall` is a WML implementation of the webshop that allows the JSP pages to produce WML instead of HTML or XHTML. The implementation is based on WALL (see chapter 4.3 for details).

⁴The mobile CSS is available in Appendix E.

```
<definition name=".skin.wall"
path="/wml/pages/storefront/wall/general/layout.jsp">
  <put name="header"
    value="/wml/pages/storefront/wall/general/header.jsp" />
  <put name="search"
    value="/wml/pages/storefront/wall/general/search.jsp" />
  <put name="menu"
    value="/wml/pages/storefront/wall/general/menu.jsp" />
  <put name="categories"
    value="/wml/pages/storefront/wall/general/categories.jsp" />
  <put name="path"
    value="/wml/pages/storefront/wall/general/path.jsp" />
  <put name="footer"
    value="/wml/pages/storefront/wall/general/footer.jsp" />
  <put name="logo"
    value="/webshop/wml/pages/storefront/wall/general/logo.gif" />
  <!-- no need for a CSS !-->
```

To change the Tiles skin when a request is made to the server, it is necessary to implement the appropriate actions and to add them to the struts-config.xml as follows:

```
<action
  path="/sf/WelcomeMobile"
  type="ordering.actions.WelcomeMobileAction"
  scope="session"
  roles="0">
  <forward name="success" path=".welcome" />
</action> <action
  path="/sf/WelcomeWML"
  type="ordering.actions.WelcomeWMLAction"
  scope="session"
  roles="0">
  <forward name="success" path=".welcome" />
</action>
```

The implementation of the different welcome actions only differ in a single detail: setting the skin, i.e. to .skin.wall for the WelcomeWallAction and .skin.mobile for the WelcomeMobileAction.

Using Tiles skins considerably facilitated our implementation, as we could base our architecture on an existing method for defining a new skin and adding a supplementary welcome action to set the skin, everything else was done by Struts and Tiles. The only disadvantage of using Tiles and Struts was that Struts does not interoperate well with the WALL tag library. This is why in some cases we were restricted to use standard Struts-tags which means that some parts of a web page might not be displayed properly on some devices. But Luca Passani who has invented WALL promised full support for Struts in the next release of WALL. The main problem of using WALL and Struts is that WALL does not integrate Struts actions. In many cases it was possible to rewrite the URL so that the Struts actions were implicitly called through their URLs rather than through the standard Struts calling method. Using URL rewriting we could avoid almost all the calls to Struts actions with very few exceptions. We therefore believe that our implementation should work on almost any device that can be found in the WURFL database.



Figure 8.4: WML version of the wall-based skin implementation. Note the presence of the search function compared to Figure 4.4.

8.1.2 Adding the WURFL Database

In order to be able to use the WALL tag library, it is necessary to install the WURFL database access service by initializing the WURFL servlet. This is done with the addition of the servlet to the applications' web.xml file:

```
<!--WURFL Initialization --> <servlet>
  <servlet-name>init</servlet-name>
  <servlet-class>net.sourceforge.wurfl.wurflapi.WurflServletInit</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

By loading it on startup, we don't need to take care of WURFL once the server is started.

8.1.3 Struts Actions with WALL

As mentioned above, one of the major problems that was encountered during the implementation phase of the thesis was the incompatibility of the WALL tag library and Struts actions. Calling a Struts action from a JSP is normally done as follows:

```
<html:link page="/sf/ViewProduct.do" paramName="product"
paramProperty="id" paramId="productId" >
```

Generally speaking, calling a Struts action is nothing more than producing an HTML link to that action that is associated with a given parameter (here productId). Unfortunately, WALL does not allow this as links in WALL cannot take parameters. Therefore, we had to use URL rewriting, thus adding the parameter to the URL as follows:

```
<bean:define id="productId" name="product" property="id" />
<wall:a href="/webshop/sf/ViewProduct.do?productId=${productId}">
  <bean:write name="product" property="name" filter="false"/>
</wall:a>
```

The statement above defines a parameter `productId` that is the Id of a product object that must be available at runtime. The link consequently refers to the `ViewProduct` action, adding the `productId` to the URL. This allows to implicitly call the `ViewProduct` action and can be used to replace most of the HTML links to Struts actions, given that the required arguments are available.

8.2 Personalization Module

The personalization module implements the adaption of the eSarine web shop web site to user preferences and interests. Although several solutions for personalization have been proposed in the past⁵, we have decided to implement a different approach to personalization that is based on Semantic Web technologies. Although the idea to store meta-data in an RDF⁶ file is not new, but the Semantic Web is not yet widely used for personalization purposes. For a semantic description to be complete, an ontology has to be provided as a base. But not only should the items be labeled with a semantic meaning in a machine-readable format, also user profiles should be stored in the same format. We therefore propose to use RDF for the profiles.

The creation of the user profiles will be done implicitly using a simple web usage mining algorithm based on click-stream analysis. As the semantic information of the webshop contents is given explicitly with the ontology, no additional classification is necessary.

The architecture of the personalization module consists of two inter-dependent parts: the semantic modelling part and the semantic interpretation part. The semantic modelling part is responsible for the generation of valid user profiles in RDF format and for automatic extraction of a valid OWL ontology from the categories in the database. The semantic interpretation part subsequently combines the RDF user profiles and the ontology of the webshop in order to produce a personalized web page of eSarine, tailored to the specific interests of the user.

8.2.1 Gathering User Data

Whenever a user selects a category or an item in the eSarine webshop, the information about the users' selection is subsequently transferred to the personalization module. The information that is stored about a users interests are based on the click-stream that the user produces while browsing. That is, a user that is interested in books will first click on the "Books"-link of the webpage which will set the counter for the book-category to one. On this page, the user might be interested in children's books which will set the counter of children's books to one and the counter of the book-category to two. What we have explained here is a general model for weighting trees as done in several domains of computer sciences. At the root, the weight is always 100 per cent.

Depending on the click-stream of a user, we are increasing the weight of the corresponding entries in our RDF user profiles. This corresponds to the tag `< user : interestWeight >` in Appendix B where a full

⁵see chapter 2 for details.

⁶Resource Description Framework, see <http://www.w3c.org>.

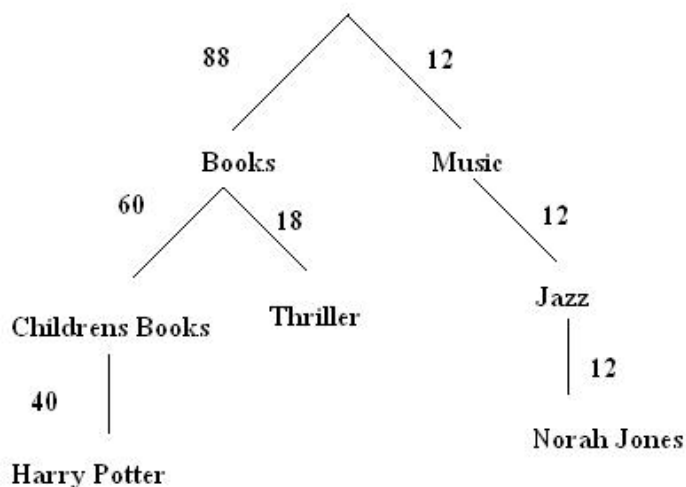


Figure 8.5: Graphical representation of a user profile. The numbers are interests values in percent. That means, this example user is mainly interested in children’s books and jazz. An ideal recommender system would therefore recommend items from these two categories.

example of a user profile is provided.

8.2.2 The Webshop Ontology

The eSarine webshop ontology is based on the entries in the webshop database. The ontology is updated whenever an operation is performed on one of the categories. The OWL output is generated with the help of the JENA API⁷. Jena is a framework for building semantic web applications. The organization of the webshop ontology is very simple and does not yet exploit the whole power of an OWL-based ontology. We have only defined the subclass relationships among the categories without implying any kind of restrictions, dependencies, or other ontology-style properties. However, it is important to note that such an extension of the ontology could be easily included in the existing architecture without changing the overall structure of the implementation. The ontology of the eSarine webshop has already been introduced in chapter 6.4.1 and the full OWL-file can be found in Appendix A.

8.2.3 Creating RDF user profiles

In general, user profiles can be stored in any kind of format as long as they remain machine-readable. But we have argued in chapter 6.5.1 that by using RDF we can considerably improve the quality of the information that is presented in the profiles.

⁷<http://jena.sourceforge.net>

As we have already seen before, RDF is a mechanism to describe statements, i.e. resources and their properties. One of the reasons why RDF is attractive for our application is that with XML we can use a standardized format with a large available set of operations to transform, parse, and understand the contents. However, as we have seen before, RDF/XML is much more than just another format. It allows us to integrate different requirements that have been defined for user profiles in chapter 6. To summarize: With RDF we can take advantage of the XML-format, use the RDF structure to convey semantical information such as the interest of a user in a specific category or item, and combine the RDF profiles with the ontology of our webshop categories. Hence, RDF is an ideal candidate for user profiles. The RDF user profiles of our implementation are organized as follows:

```
<rdf:Description rdf:about="http://category.esarine.net/category#10">
  <user:categoryWeight>3</user:categoryWeight>
  <user:lastAccessed>14.08.2005</user:lastAccessed>
  <user:categoryId>10</user:categoryId>
  <user:interestFactor>3.7437418</user:interestFactor>
</rdf:Description>
```

The excerpt above originates from the user profile in Appendix B. It represents a single category (resource: <http://category.esarine.net/category#10>) that has been visited by the user. It contains the category weight which is a counter that shows how many time the user has accessed the category in previous visits. But the most important information that can be extracted from this profile is the interestFactor that is calculated as:

```
interestFactor = [weight]/[days since last accessed].
```

The interest factor determines to what extent a category is of interest for a user and serves as a base for our recommender system that provides direct links to the users' favorite categories.

The generation of the user profiles with Jena is very simple. To illustrate how Jena works, let us have a look at some code examples:

```
//create an empty RDF model:
model = ModelFactory.createDefaultModel();

//read model from InputStream in:
model.read(in)

//get a specific category resource with categoryURI:
Resource categoryResource = model.getResource(categoryURI);

//retrieve the weight property of the
Statement weightStmt =
categoryResource.getProperty(PROFILE.categoryWeight);

//increment weight:
categoryResource.addProperty(PROFILE.categoryWeight,
weightStmt.getInt() + 1);
```

This code snippet illustrates the mode of functioning of the Jena API. First, create a default model into which we load the contents of the input file (in RDF/XML format). The method `getResource` allows us to retrieve the category resource from which we can fetch the weight property of the category resource. The `interestFactor` is later updated as follows:

```
Statement dateStmt = element.getProperty(PROFILE.lastAccessed);
String dateString = dateStmt.getString();
int days = GeneralMethods.dateDifferenceToNow(dateString);
Statement weightStmt = element.getProperty(PROFILE.categoryWeight);
int weight = weightStmt.getInt();

//add interest factor
if(days > 0){
    Statement rmStmt = element.getProperty(PROFILE.interestFactor);
    float fact = weight/days;
    rmStmt.remove();
    element.addProperty(PROFILE.interestFactor, fact);
}
```

We retrieve the `lastAccessed` property and calculate the date difference to the given moment in order to calculate the `interestFactor` with the weight and the time difference.

In the last section we have seen how the category ontology is created from the given database entries. In this section we have presented implementation details of the user profiles that are stored in RDF/XML format. We have repeated several times that the format of the profiles and the ontology is crucial for the success of our approach. In the next section we will present the semantic inference reasoner, a class that provides the missing link to connect the user profiles with the category. We are going to show where the connection between the two units is made, where the control system of the recommender system is located, and how these mechanism are implemented for the eSarine webshop framework.

8.2.4 The Semantic Inference Reasoner

The semantic inference reasoner class is the heart of the semantic interpretation part, it reads the two documents, the RDF user profile and the OWL of the webshop as input and produces a list of products that is to be recommended to the user:

```
public Vector getFavorites(){

    TreeMap tree = new TreeMap();

    for( ResIterator iter = infModel.listSubjectsWithProperty(PROFILE.interestFactor);
        iter.hasNext();){

        Resource element = (Resource)iter.next();
        float interestFactor = element.getProperty(PROFILE.interestFactor).getFloat();
        tree.put(new Float(element.getProperty(PROFILE.interestFactor).getFloat()),
            new Integer(element.getProperty(PROFILE.categoryId).getInt()));
    }
}
```

```
}

Vector favorites = new Vector();
for(int a = PersonalizationStrings.personalizedListSize; a > 0 ; a--){
    if(tree.size() >= 1){
        Float t = (Float) tree.lastKey();
        favorites.add(tree.get(t));
        tree.remove(t);
    }
    else{
        a = 0;
        //exit the loop
    }
}
return favorites;
}
```

The `getFavorites` method that finally returns a vector of favorites to the welcome page produces an ordered vector starting from the category with the highest interest factor. The `TreeMap` tree is built based on the ordered values of the `interestFactor` tag in the RDF/XML user profile. As one can see the implementation of the recommender system is still very simply and has rather been implemented in order to show how such a system could be built than to actually build a real recommender system. It takes the categories with the highest interest factors as input and subsequently produces a list of favorite categories. This list is then sent to the welcome action that finally chooses the categories to be recommended. The simplicity of the implementation shows how much we can gain by using semantic web technologies rather than traditional recommender mechanisms: Due to the semantic information that is available for each category, we can determine the user preferences from the browsing behavior.

8.3 Conclusion

We would have liked to present a mobile payment implementation in this thesis, but as discussed in chapter 7, such a solution can only be realized in a local context with the support of mobile providers or financial institutions. We have also seen that the mobile payment sector is still in its beginnings with only very few concrete developments in Switzerland. Given the absence of standards and support we have therefore decided not to implement such a mechanism, but to rather encourage users of our webshop system to use Paypal for mobile payments in our webshop. The support of paypal is already ensured in our system.

Most of the solutions presented in this chapter are based on a compromise with the goal of minimizing changes in the existing webshop architecture. Of course, in some cases the architecture of the webshop was quite restrictive, especially when considering the choice of technologies. The combination of Struts and Tiles is very attractive for such a webshop system, but makes the adaptation to mobile device needs more complex. WALL for example does not work well with both, Struts and Tiles. And, although we

would have liked our personalization module to be more flexible, i.e. by allowing webshop providers to choose if they want to add personalization or not, we were forced to change parts of the implementation in order to add the personalization module. Therefore we had to change some of the webshop Java Beans, especially the UserBean in order to provide a personalization functionality. Nevertheless, we believe that the power of our implementation lies in its simplicity and the combination of the chosen technologies.

Chapter 9

Future Work

The implementation proposed in this thesis is only a prototype that has been built in order to test different specifications, implementations, and architectures. The mobile web is still a relatively young phenomenon that will change rapidly in future. Therefore, we have tried to discuss different aspects of the field by using the eSarine webshop as a platform for testing our ideas and providing a concrete example to show how such an implementation could be realized.

9.1 Device Independence

Implementing device independent websites is a very challenging task as new devices appear on the market regularly, implementing a different version of the given specifications. With WALL/WURFL these problems can be met from a developers point of view, as the implementation of the WALL tag library takes care of device discrepancies, but this requires that the WURFL database as well as the implementation of WALL are constantly up to date, thus keeping pace with the changing device specifications.

For todays web programmers, the best solution to meet these problems, the best advice is to consistently separate content from presentation. This allows to quickly adapt the presentation layer to meet new device specifications. Although future devices might be able to display XHTML pages, and they will feature support for CSS media types, certain adaptation of the website might still be necessary. A completely different scenario for future devices predicts that such adaptations will become needless as the device browsers themselves will take care of the website alteration to meet the requirements for rendering to a specific device. Even today, Opera follows this approach by offering a small screen rendering mechanism. This would imply that in future, developing websites to meet device capabilities would not be necessary as the devices themselves may take care of these aspects. Nevertheless, the strict partition of websites into content and presentation is essential in order to prevent conflicting device capabilities that make the rendering for specific devices impossible.

9.2 Personalization

Personalization is said to become more and more important with the continuing growth of the world wide web. Relevant information may become increasingly difficult to locate for users so that they may feel impelled to use personalization agents that will help them find the information they want. Today Google for example offers a personalized search service that arranges search results according to user preferences. Unfortunately, without semantic information about website contents, it is very difficult for agents to decide which information is relevant to a user. But with the emergence of the semantic web, semantically enhanced web pages will help personalization agents to classify websites and to decide whether the proposed information is interesting for the user or not. In order for the vision of the semantic web to become true, it is necessary for applications to be built according to the standards so that information agents are able to process the semantic information. For the purpose of our implementation we have tried to use standards where ever standards were available, such as RDF or OWL. We would have liked to use shared ontologies to promote interoperability with other applications, but we have not been able to find an ontology that met our requirements. Nevertheless, our implementation is built in a way that enables developers to easily replace the given ontology with a different as long as the format remains the same, i.e. OWL.

9.2.1 Personalization Refinement

Our implemented personalization mechanism could be enhanced in several aspects. We believe that the overall architecture is stable enough to remain unchanged, but the different parts that make up the personalization module of our application could easily be replaced with different implementations.

Personalization Algorithm

The algorithm that we have used for the purpose of this thesis is fairly simple and does not take into account all the different aspects of personalization. For example, it could be interesting to consider viewing period or viewing time, thus, increasing the interest factor if the page has been viewed several times within short intervals or for a very long time. Other improvements could be considered like the combination of item and category weights in order to produce different recommendations. Personalization could also be used in a broader sense by ordering search results and category listings according to the user preferences. Of course, this would require a refinement of the personalization mechanism which is not yet built according to these specifications.

Combining CC/PP and RDF Profiles

Given that CC/PP, UAProf, and the eSarine RDF user profiles are all expressed in a single format, RDF, it could be an interesting approach to enhance the communication and interaction between profiles. Such an architecture could be based on a simple agent mechanism in which agents would interact in order to solve a common personalization task. A similar agent-based implementation for personalization

has been implemented by the GraniteNight project [20] where it seemed to be quite successful. Unfortunately, introducing agents to the eSarine webshop system would require a certain number of adaptations of the architecture in order to be able to accommodate software agents. Nevertheless, introducing agents to eSarine would be conform to the vision of the semantic web. eSarine could therefore offer interface which third-party agents could implement in order to export eSarine as a web service.

A third profile that could be introduced to eSarine is P3P to describe eSarine privacy specifications which again could be retrieved by third-party agents. The implementation of a mechanism that would allow the combination of different profiles and specifications could considerably improve the personalization mechanism. However, such an implementation would also lead to a far more complicated architecture and personalization algorithm which would have to take into account much more information to produce a personalized web shop experience.

Using RDF Queries

For the GraniteNight project, Grimnes et al [20] have proposed an implementation of RDFQL, an RDF query language. A similar approach could be used for the eSarine webshop to simplify the matching of ontology and RDF user profiles. Additionally, a RDF query language would provide a good abstraction for the implementation of an agent-based architecture as described above.

9.3 Mobile Payment

We would have liked to propose a mobile payment solution for the eSarine webshop, but the mobile payment situation today is very inconsistent. Therefore, we argue that for a mobile payment solution to be successful, it is important that it is implemented according to the local specifications of the country in which the webshop system is actually being used. Obviously, the world wide web is global, but the mobile payment market is not. Different mobile network operators offer different payment solutions, security specifications, and billing infrastructure. In this thesis, we have proposed to use Paypal which is a global payment mechanism. Maybe other solutions will be proposed in future, but at the moment the future of mobile payment is very difficult to predict. Two different developments seem possible: local solutions that are based on cooperations with network operators, and global solutions like Paypal that can be used world wide, independent of locations and currencies.

Chapter 10

Conclusion

In this thesis we have presented a way to adapt an existing webshop system for use with mobile devices. We showed which aspects of the webshop have to be altered in order to make the website attractive for mobile device users. The development of a "mobile" version of the eSarine website has been performed on three levels. In a first stage we tried to adapt the existing presentation layer to meet different device capabilities and specifications. We showed that it is crucial for the success of these transformations to strictly separate content and presentation. The eSarine webshop has already followed this principle from the beginning of its development. The JSP pages as well as the CSS stylesheet were only concerned with the layout and markup of the webshop. The entire text that is displayed on the website is either stored in Struts Actions, Java Beans, or the connected database. This fact facilitated the implementation of a different presentation considerably. For the purpose of this thesis, we proposed two different implementations of the presentation layer for mobile devices: a mechanism based on CSS media types, and an implementation that takes advantage of the WALL tag library to render a website to mobile devices. Both approaches seemed to work well. Unfortunately the CSS 2.0 standard is not yet widely supported by mobile device browsers, but we hope that this support will be available in future. We also showed that with the emergence of CSS 3.0 new extensions of Media Types will be available which will in turn facilitate mobile web development in many aspects. At the time of writing, the CSS Media Types were not yet widely available which made us choose the WALL tag library as an alternative. We believe that the interoperability and device independent development offered by WALL is a very interesting abstraction mechanism for mobile websites. Although CSS allows us to make certain adaptations to device capabilities, WALL goes much further by not just changing certain aspects or details of a website, but by actually building a completely new markup that is tailored to the respective device. As for now, it is difficult to predict which method will be more successful for the future. This mainly depends on the device manufacturers and on the approach they follow to implement W3C standards such as XHTML and CSS. If manufacturers will continue supporting different standards, the approach offered by WALL is definitely going to be more successful as it offers far more flexibility and a good level of abstraction for mobile web developers.

The second part of the thesis was concerned with personalization of the eSarine webshop content. We

implemented a mechanism that allows to semantically describe the content of the webshop by building an ontology for the eSarine categories. We showed that the way eSarine presents the data is already conform to an OWL model. This means that all we actually had to do is to describe a consistent way for describing the structure of the data in OWL/XML format in order for the semantic inference reasoner to be able to understand the information correctly. Furthermore, we presented a method to build RDF user profiles that capture the different interests of a user and to describe them in a machine-readable format. We chose to use semantic web technologies for the personalization module which turned out to be a good decision as the interpretation of user interests is largely facilitated by the fact that the semantic web has been built for exactly this purpose. We are aware of the fact that our implementation is very straightforward and that a more complex approach might have led to different results. Nevertheless we believe that we were able to demonstrate the power of using semantic web technologies for personalization and that even our simple approach led to very interesting results, especially compared to the time and effort that have been invested in more elaborate non-semantic solutions. We also argue that we gain a lot of computing time by using this approach as it does not require any clustering of user groups with similar interests or the implicit classification of webshop items.

The third and last part of the thesis covered the broad subject of mobile payment. In the beginning we had actually planned to provide a mobile payment solution for eSarine, but then, after reading the literature we came to the conclusion that proposing an implementation at the present situation of the state-of-the-art would be very difficult. Therefore, we refrained from implementing our own solution, but we presented a detailed overview of different approaches that were available at the time of writing. We also proposed to use Paypal that is available also for mobile payment and which is already supported by eSarine.

Bibliography

- [1] OWL Web Ontology Language Overview. Retrieved Mai 12, 2005 from <http://www.w3.org/TR/owl-features>.
- [2] Wireless Watch Japan. Retrieved August 12, 2005 from <http://www.wirelesswatch.jp>.
- [3] G. Adomavicius and A. Tuzhilin. User profiling in personalization applications through rule discovery and validation. In *Proceedings of KDD-99*, pages 377–381, 1999.
- [4] B. Adrian. Overview of the mobile payments market 2002 through 2007. *Gartner*, November 2002.
- [5] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A new method for similarity indexing for market data. In *Proceedings of the 1999 ACM SIGMOD Conference, Philadelphia*, June 1999.
- [6] C. R. Anderson, P. Domingos, and D. S. Weld. Personalizing Web Sites for Mobile Users. In *Proceedings of the 10th Conference on the World Wide Web (WWW10)*, 2001.
- [7] B. Ballard. *UI Design Guidelines for Mobile Web Development*. Little Springs Design, 2004.
- [8] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.
- [9] D. Billsus, C. Brunk, C. Gladish, and M. Pazzani. Adaptive Interfaces for Ubiquitous Web Access. *Comm. ACM 45*, May 2002.
- [10] D. Billsus and M.J. Pazzani. Learning collaborative information filters. In *Proceedings of the International Conference on Machine Learning, Madison, WI*, 1998.
- [11] S. Braynov. Personalization and Customization Technologies. In *The Internet Encyclopedia*. John Wiley and Sons, 2003.
- [12] P. Brusilovsky and W. Nejdl. Adaptive Hypermedia and Adaptive Web. In *Practical Handbook of Internet Computing*. M. P. Singh, 2002.
- [13] M. Deshpande and G. Karypis. RItem-based top-n recommendation algorithms. *ACM Transactions on Information Systems 22(1)*, pages 1 – 34, 2004.
- [14] M. S. Ding and J. F. Hampe. Reconsidering the challenges of mpayments: A roadmap to plotting the potential of the future mcommerce market. In *16th Bled eCommerce Conference, Bled, Slovenia*, 2003.

- [15] P. Edwards, G. A. Grimnes, , and A. Preece. An Empirical Investigation of Learning from the Semantic Web. In *Proceedings of ECML/PKDD-2002 Semantic Web Mining Workshop*, 2002.
- [16] M. Eirinaki and M. Vazirgiannis. Web Mining for Web Personalization. *ACM Transactions on Internet Technology*, Vol. 3, No. 1, pages 1 – 27, February 2003.
- [17] Mobile Payment Forum. White Paper. Retrieved June 1, 2005 from http://www.mobilepaymentforum.org/pdfs/mpf_whitepaper.pdf, 2002.
- [18] The WAP Forum. Wireless Application Protocol WAP 2.0: Technical White Paper. Retrieved August, 2005 from http://www.wapforum.org/what/WAPWhite_Paper1.pdf.
- [19] D. Frauchiger, A. Meier, H. Stormer, and N. Werro. Zur Entwicklung des Struts-basierten Webshops eSarine. *HMD - Praxis der Wirtschaftsinformatik*, Jhrg 41, HMD Nr. 238, August 2004.
- [20] G. A. Grimnes, S. Chalmers, P. Edwards, and A. Preece. GraniteNights - A Multi-Agent Visit Scheduler Utilising Semantic Web Technology. In *Proceedings of the Seventh International Workshop on Cooperative Information Agents*. Springer Helsinki, Finland, 2003.
- [21] G. A. Grimnes, P. Edwards, and A. Preece. Learning from Semantic Flora and Fauna. In *Proceedings of the AAAI Workshop on Semantic Web Personalization*, 2004.
- [22] K. Hendry. Web engineering for mobile devices. Master's thesis, University of Helsinki, 2005.
- [23] Andreas Huber. Mobile Payment a Comparison Between Europe and the U.S. Technical report, University of Zurich, 2004.
- [24] T. Husted, C. Dumoulin, G. Franciscus, and D. Winterfeldt. *Struts in Action*. Manning Greenwich, 2003.
- [25] R. Jones. Creating Web Content for Mobile Phone Browsers. Retrieved October 29, 2004, from http://www.oreillynet.com/pub/a/wireless/2004/02/06/mobile_browsing/.
- [26] S. Karnouskos. Mobile payment: A journey through existing procedures and standardization initiatives. *IEEE Communications Surveys and Tutorials* 6(4), page 4466, 2004.
- [27] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. R. Grouplens. Applying collaborative filtering to usenet news. *Communications of the ACM* 40(3), 1997.
- [28] P. Kontinurmi. User Profiles and Their Management. Technical report, Helsinki University of Technology, Software Business and Engineering Institute, 2001.
- [29] B. Mobasher, Xin Jin, and Yanzan Zhou. Semantically Enhanced Collaborative Filtering on the Web. citeseer.ist.psu.edu/694625.html, 2004.
- [30] M. OConner and J. Herlocker. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems, Berkeley, CA*, August 1999.

- [31] Jan Ondrus, Giovanni Camponovo, and Yves Pigneur. A Proposal for a Multi-perspective Analysis of the Mobile Payment Environment. In *Proceedings of the 4th International Conference on Mobile Business (ICMB'05)*, 2005.
- [32] L. Passani and A. Trasatti. WURFL and WALL. Retrieved Mai 5, 2005 from <http://wurfl.sourceforge.net/>.
- [33] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommender algorithms for e-commerce. In *Proceedings of the 2nd ACM E-Commerce Conference (EC00)*, Minneapolis, MN, October 2000.
- [34] B. Siggelkow. *Jakarta Struts Cookbook*. O'Reilly, 2005.
- [35] Axel Spriestersbach, Thomas Ziegert, Guido Grassel, Michael Wasmund, and Gabriel Derrler. Flexible pagination and layouting for device independent authoring. In *WWW2003 Emerging Applications for Wireless and Mobile access Workshop*, 2003.
- [36] H. Stormer. Personalized Websites for Mobile Devices using dynamic Cascading Style Sheets. In *Proceedings of the 2nd International Conference on Advances in Mobile Multimedia (MoMM)*. Bali, September 2004.
- [37] John Ure. Mobile Commerce in Hong Kong. Technical report, Center of Telecom management, Marshall School of Business, University of Southern California, 2003.
- [38] W3C. Media Queries, W3C Candidate Recommendation 8 July 2002. Retrieved in August, 2005 from <http://www.w3.org/TR/css3-mediaqueries/>.
- [39] Device Independence Working Group W3C. Authoring Techniques for Device Independence. Retrieved Mai 5, 2005 from <http://www.w3.org/TR/2004/NOTE-di-atdi-20040218/>.
- [40] SYMM Working Group W3C. Synchronized Multimedia Integration Language (SMIL 2.0). Retrieved August, 2005 from <http://www.w3.org/TR/2005/REC-SMIL2-20050107/>.
- [41] N. Werro, H. Stormer, D. Frauchiger, and A. Meier. eSarine - A Struts-based Webshop for Small and Medium-sized Enterprises. In *Proceedings of the EMISA Conference - Information Systems in E-Business and E-Government*. Luxembourg, October 2004.
- [42] A. Zmijewska, E. Lawrence, and R. Steele. Towards understanding of factors influencing user acceptance of mobile payment systems. In *Proceedings of the IADIS WWW/Internet*, Madrid, 2004.

Appendix A

The Webshop Ontology

```

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:esarine="http://esarine.unifr.ch/2005/owl/1.0#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">

  <owl:Class rdf:about="http://category.esarine.net/category#-1">
    <esarine:categoryName>No Parent</esarine:categoryName>
    <esarine:categoryId>-1</esarine:categoryId>
  </owl:Class>

  <owl:Class rdf:about="http://category.esarine.net/category#1">
    <esarine:categoryName>Books</esarine:categoryName>
    <esarine:categoryId>1</esarine:categoryId>
  </owl:Class>

  <owl:Class rdf:about="http://category.esarine.net/category#4">
    <esarine:categoryName>Music</esarine:categoryName>
    <esarine:categoryId>4</esarine:categoryId>
  </owl:Class>

  <owl:Class rdf:about="http://category.esarine.net/category#7">
    <esarine:categoryName>Hard Rock</esarine:categoryName>
    <esarine:categoryId>7</esarine:categoryId>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://category.esarine.net/category#4"/>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:about="http://category.esarine.net/category#3">
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://category.esarine.net/category#1"/>
    </rdfs:subClassOf>
  </owl:Class>

```

```
<esarine:categoryName>Children's Books</esarine:categoryName>
<esarine:categoryId>3</esarine:categoryId>
</owl:Class>

<owl:Class rdf:about="http://category.esarine.net/category#6">
  <esarine:categoryId>6</esarine:categoryId>
  <esarine:categoryName>Pop</esarine:categoryName>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://category.esarine.net/category#4"/>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="http://category.esarine.net/category#11">
  <esarine:categoryId>11</esarine:categoryId>
  <rdfs:subClassOf rdf:resource="http://category.esarine.net/category#7"/>
  <esarine:categoryName>Iron Maiden</esarine:categoryName>
</owl:Class>

<owl:Class rdf:about="http://category.esarine.net/category#2">
  <esarine:categoryName>Thriller</esarine:categoryName>
  <esarine:categoryId>2</esarine:categoryId>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://category.esarine.net/category#1"/>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="http://category.esarine.net/category#12">
  <esarine:categoryName>Metallica</esarine:categoryName>
  <rdfs:subClassOf rdf:resource="http://category.esarine.net/category#7"/>
  <esarine:categoryId>12</esarine:categoryId>
</owl:Class>

<owl:Class rdf:about="http://category.esarine.net/category#9">
  <rdfs:subClassOf rdf:resource="http://category.esarine.net/category#6"/>
  <esarine:categoryName>Nickelback</esarine:categoryName>
  <esarine:categoryId>9</esarine:categoryId>
</owl:Class>

<owl:Class rdf:about="http://category.esarine.net/category#10">
  <rdfs:subClassOf rdf:resource="http://category.esarine.net/category#6"/>
  <esarine:categoryId>10</esarine:categoryId>
  <esarine:categoryName>Robbie Williams</esarine:categoryName>
</owl:Class>

<owl:Class rdf:about="http://category.esarine.net/category#5">
  <esarine:categoryId>5</esarine:categoryId>
  <esarine:categoryName>Jazz</esarine:categoryName>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://category.esarine.net/category#4"/>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="http://category.esarine.net/category#13">
```

```
<esarine:categoryName>Bobby McFerrin</esarine:categoryName>
<rdfs:subClassOf rdf:resource="http://category.esarine.net/category#5"/>
<esarine:categoryId>13</esarine:categoryId>
</owl:Class>

<owl:Class rdf:about="http://category.esarine.net/category#8">
  <rdfs:subClassOf rdf:resource="http://category.esarine.net/category#5"/>
  <esarine:categoryName>Norah Jones</esarine:categoryName>
  <esarine:categoryId>8</esarine:categoryId>
</owl:Class>

</rdf:RDF>
```

Appendix B

A RDF user profile

```

<rdf:RDF
  xmlns:user="http://esarine.unifr.org/2005/user_profile-rdf/1.0#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" >

  <rdf:Description rdf:about="http://user.esarine.net/user#1000">
    <user:userId>1000</user:userId>
  </rdf:Description>

  <rdf:Description rdf:about="http://category.esarine.net/category#10">
    <user:categoryWeight>3</user:categoryWeight>
    <user:lastAccessed>14.08.2005</user:lastAccessed>
    <user:categoryId>10</user:categoryId>
    <user:interestFactor>3.7437418</user:interestFactor>
  </rdf:Description>

  <rdf:Description rdf:about="http://category.esarine.net/category#1">
    <user:categoryWeight>6</user:categoryWeight>
    <user:interestFactor>7.4874835</user:interestFactor>
    <user:lastAccessed>14.08.2005</user:lastAccessed>
    <user:categoryId>1</user:categoryId>
  </rdf:Description>

  <rdf:Description rdf:about="http://category.esarine.net/category#3">
    <user:lastAccessed>14.08.2005</user:lastAccessed>
    <user:categoryWeight>2</user:categoryWeight>
    <user:interestFactor>2.495828</user:interestFactor>
    <user:categoryId>3</user:categoryId>
  </rdf:Description>

  <rdf:Description rdf:about="http://category.esarine.net/category#6">
    <user:interestFactor>3.7437415</user:interestFactor>
    <user:categoryId>6</user:categoryId>
    <user:categoryWeight>3</user:categoryWeight>
    <user:lastAccessed>14.08.2005</user:lastAccessed>
  </rdf:Description>

```

```
<rdf:Description rdf:about="http://category.esarine.net/category#2">
  <user:lastAccessed>14.08.2005</user:lastAccessed>
  <user:categoryId>2</user:categoryId>
  <user:interestFactor>1.2479138</user:interestFactor>
  <user:categoryWeight>1</user:categoryWeight>
</rdf:Description>

<rdf:Description rdf:about="http://category.esarine.net/category#4">
  <user:interestFactor>3.7437415</user:interestFactor>
  <user:lastAccessed>14.08.2005</user:lastAccessed>
  <user:categoryId>4</user:categoryId>
  <user:categoryWeight>3</user:categoryWeight>
</rdf:Description>

</rdf:RDF>
```

Appendix C

XMLSchema for User Profiles

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://unifr.esarine.ch/xsd/user_profiles/"
  targetNamespace="http://unifr.esarine.ch/xsd/user_profiles/"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <xs:annotation>
    <xs:documentation xml:lang="en">
      Simple XML Schema, 2005-02-09
      by Muriel Bowie
      This schema defines terms for user profile namespace used
      for the E-sarine webshop environment.
    </xs:documentation>
  </xs:annotation>

  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/03/xml.xsd">
  </xs:import>

  <xs:element name="userId" type="xs:decimal"/>

  <xs:element name="itemName" type="xs:string"/>
  <xs:element name="itemId" type="xs:decimal"/>
  <xs:element name="itemCategoryId" type="xs:decimal"/>
  <xs:element name="itemWeight" type="xs:decimal"/>
  <xs:element name="lastAccessed" type="xs:date"/>

  <xs:element name="categoryId" type="xs:decimal"/>
  <xs:element name="categoryName" type="xs:string"/>
  <xs:element name="categoryWeight" type="xs:decimal"/>

  <xs:complexType name="elementType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
```

```
        <xs:attribute ref="xml:lang" use="optional"/>
    </xs:extension>
</xs:simpleContent>
</xs:complexType>

<xs:element name="userProfile">
<xs:complexType>
  <xs:sequence>
    <xs:element ref="userId"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded" >
      <xs:element ref="item" maxOccurs="unbounded"/>
      <xs:element ref="category" maxOccurs="unbounded"/>
    </choice>
  </xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="item">
<xs:complexType>
  <xs:sequence>
    <xs:element ref="itemName"/>
    <xs:element ref="itemId"/>
    <xs:element ref="itemCategory" minOccurs="1" maxOccurs="unbounded"/>
    <xs:element ref="itemWeight"/>
    <xs:element ref="lastAccessed"/>
  </xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="category">
<xs:complexType>
  <xs:sequence>
    <xs:element ref="categoryId"/>
    <xs:element ref="categoryName"/>
    <xs:element ref="categoryWeight"/>
    <xs:element ref="lastAccessed"/>
  </xs:sequence>
</xs:complexType>
</xs:element>

</xs:schema>
```

Appendix D

SMIL Examples

The following SMIL examples are based on [39].

D.1 Text Positioning

```
<smil xmlns="http://www.w3.org/2001/SMIL20/">
  <head>
    <layout>
      <root-layout width="320" height="480" />
      <region id="a" top="5" bottom="100" />
      <region id="b" top="200" bottom="280" />
    </layout>
  </head>
  <body>
    <text region="a" src="text.html"/>
    <text region="b" src="additional_text.html"/>
  </body>
</smil>
```

D.2 The Switch Element

```
<smil:switch>
  <html:img
    src="bigColorImages/logo.gif"
    title="Company logo"
    alt="logo"
    smil:systemScreenSize="768X1024" />
  <html:img
    src="smallMonoImages/logo.gif"
    title="Company logo"
    alt="logo"
    smil:systemScreenSize="160X160" />
  <html:img
    src="defaultImages/logo.gif"
    title="Company logo"
```

```
    alt="logo" />  
</smil:switch>
```

Appendix E

The eSarine Mobile CSS

```
body {
max-width:100%;
padding-right: 0px; padding-left: 0px; font-size: 11px; font-family:
'vera sans',helvetica,sans-serif; }

table, tbody, tr, td, th {
width:100%;
max-width: 100%;
padding-right: 0px; padding-left: 0px; display: block; float:none;
border:none; font-size: 11px; font-family: 'vera
sans',helvetica,sans-serif; text-align: justify; }

p{ font-size: 11px; font-family: 'vera sans',helvetica,sans-serif;
text-align: justify; }

.category{ display:block; float:none;
width:100%;
}

img{ display:none; }

.menu{ align:top; }

.body{ display:block; }

form { padding-right: 0px; padding-left: 0px; padding-bottom: 0px;
margin: 0px; padding-top: 0px }

a:link {
background: none transparent scroll repeat 0% 0%;
color: #0049aa; text-decoration: underline }
```

```
A:active {
background: none transparent scroll repeat 0% 0%;
color: #0049aa; text-decoration: underline }

a:visited { color: #23488c; text-decoration: underline }

em{ font-style: normal; font-weight: bold }

b { font-weight: bold }

input { font-size: 11px; margin: 0px }

abbr { border-right: medium none; border-top: medium none;
border-left: medium none; border-bottom: medium none }

acronym { border-right: medium none; border-top: medium none;
border-left: medium none; border-bottom: medium none }

ol { padding-right: 0px; padding-left: 0px; padding-bottom: 0px;
margin: 0px 0px 8px 5px; padding-top: 0px }

ul { padding-right: 0px; padding-left: 0px; padding-bottom: 0px;
margin: 0px 0px 8px 5px; padding-top: 0px }

dl { padding-right: 0px; padding-left: 0px; padding-bottom: 0px;
margin: 0px; padding-top: 0px }

li { padding-right: 0px; padding-left: 0px; list-style-position:
inside; font-size: 7px; padding-bottom: 0px; margin: 0px;
padding-top: 0px }

textarea {
width: 100%
}

unknown{
width: 96%
}

dd { padding-right: 5px; padding-left: 5px; padding-bottom: 0px;
margin: 5px 0px; padding-top: 0px }

dt { margin-top: 10px; background: #eee; padding-right: 5px;
padding-left: 5px; padding-bottom: 0px; margin: 5px 0px;
padding-top: 0px }

h1 { clear: both; padding-right: 5px; padding-left: 5px; font-size:
12px; background: #951515; padding-bottom: 1px; margin: 0px; color:
#fff; padding-top: 1px }

h2 {
clear: both;
```

```
border-right: #333 0px solid;
padding-right: 5px;
border-top: #333 1px solid;
padding-left: 0px;
font-size: 12px;
background: #eee;
padding-bottom: 1px;
margin: 0px;
border-left: #333 0px solid;
padding-top: 1px;
border-bottom: #333 1px solid
}

h3 {
padding-right: 0px;
padding-left: 0px;
font-size: 12px;
padding-bottom: 0px;
margin: 0px;
padding-top: 0px
}

h4 { padding-right: 0px; padding-left: 0px; font-size: 12px;
padding-bottom: 0px; margin: 0px; padding-top: 0px }

h5 { padding-right: 0px; padding-left: 0px; font-size: 12px;
padding-bottom: 0px; margin: 0px; padding-top: 0px }

h6 { padding-right: 0px; padding-left: 0px; font-size: 12px;
padding-bottom: 0px; margin: 0px ; padding-top: 0px }

select { font-size: 7px; margin-bottom:0px;
max-width: 100%
}
```

Appendix F

WALL: Example JSP-Code

The following WALL example is based on [32]:

```
<%@ taglib uri="/WEB-INF/tld/wall.tld" prefix="wall" %>
<wall:document><wall:xmlpidtd />
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<wall:load_capabilities /> <wall:head>
  <wall:title>Cool Portal</wall:title>

  <c:set var="gridsize" value="2" />
  <c:if test="{capabilities.resolution_width >= 160}">
    <c:set var="gridsize" value="3" />
  </c:if>
  <wall:cool_menu_css colnum="{gridsize}"/>
</wall:head>

<wall:body>
  <wall:cool_menu colnum="{gridsize}">
  <wall:cell>
    <wall:img src="pix/movies.gif" alt="Cinema" >
    <wall:alternate_img src="pix/movies_big.gif"
      test="{capabilities.resolution_width >= 160}" />
    <wall:alternate_img nopicture="true"
      test="{not capabilities.gif}" />
    <wall:alternate_img opwv_icon="camcorder"
      test="{capabilities.resolution_width <= 90}" />
    <wall:alternate_img eu_imode_icon="#58999;"
      test="{capabilities.resolution_width <= 90}" />
    <wall:alternate_img ja_imode_icon="#63704;"
      test="{capabilities.resolution_width <= 90}" />
    </wall:img>
    <wall:a href="http://url1" title="">Cinema</wall:a>
  </wall:cell>

  <wall:cell>
    <wall:img src="pix/dice.gif" alt="Games" >
    <wall:alternate_img src="pix/dice_big.gif"
      test="{capabilities.resolution_width >= 160}" />
```

```
<wall:alternate_img nopicture="true"
    test="{not capabilities.gif}" />
<wall:alternate_img opwv_icon="dice"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img eu_imode_icon="#59024;"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img ja_imode_icon="#63729;"
    test="{capabilities.resolution_width <= 90}" />
</wall:img>
<wall:a href="http://url1" title="">Games</wall:a>
</wall:cell>

<wall:cell>
<wall:img src="pix/holiday.gif" alt="Travel" >
<wall:alternate_img src="pix/holiday_big.gif"
    test="{capabilities.resolution_width >= 160}" />
<wall:alternate_img nopicture="true"
    test="{not capabilities.gif}" />
<wall:alternate_img opwv_icon="plane"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img eu_imode_icon="#58978;"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img ja_imode_icon="#63683;"
    test="{capabilities.resolution_width <= 90}" />
</wall:img>
<wall:a href="http://url1" title="">Travel</wall:a>
</wall:cell>

<wall:cell>
<wall:img src="pix/heart.gif" alt="Lovelif" >
<wall:alternate_img src="pix/heart_big.gif"
    test="{capabilities.resolution_width >= 160}" />
<wall:alternate_img nopicture="true"
    test="{not capabilities.gif}" />
<wall:alternate_img opwv_icon="heart"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img eu_imode_icon="#59116;"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img ja_imode_icon="#63889;"
    test="{capabilities.resolution_width <= 90}" />
</wall:img>
<wall:a href="http://url1" title="">Lovelif</wall:a>
</wall:cell>

<wall:cell>
<wall:img src="pix/clouds.gif" alt="Weather" >
<wall:alternate_img src="pix/clouds_big.gif"
    test="{capabilities.resolution_width >= 160}" />
<wall:alternate_img nopicture="true"
```

```
        test="{not capabilities.gif}" />
<wall:alternate_img opwv_icon="partcloudy"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img eu_imode_icon="#58942;"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img ja_imode_icon="#63647;"
    test="{capabilities.resolution_width <= 90}" />
</wall:img>
<wall:a href="http://url1" title="">Weather</wall:a>
</wall:cell>

<wall:cell>
<wall:img src="pix/headphones.gif" alt="Music" >
<wall:alternate_img src="pix/headphones_big.gif"
    test="{capabilities.resolution_width >= 160}" />
<wall:alternate_img nopicture="true"
    test="{not capabilities.gif}" />
<wall:alternate_img opwv_icon="speaker"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img eu_imode_icon="#59126;"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img ja_imode_icon="#63899;"
    test="{capabilities.resolution_width <= 90}" />
</wall:img>
<wall:a href="http://url1" title="">Music</wall:a>
</wall:cell>

<wall:cell>
<wall:img src="pix/soccer.gif" alt="Sport" >
<wall:alternate_img src="pix/soccer_big.gif"
    test="{capabilities.resolution_width >= 160}" />
<wall:alternate_img nopicture="true"
    test="{not capabilities.gif}" />
<wall:alternate_img opwv_icon="football"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img eu_imode_icon="#58966;"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img ja_imode_icon="#63671;"
    test="{capabilities.resolution_width <= 90}" />
</wall:img>
<wall:a href="http://url1" title="">Sport</wall:a>
</wall:cell>

<wall:cell>
<wall:img src="pix/magnifier.gif" alt="Search" >
<wall:alternate_img src="pix/magnifier_big.gif"
    test="{capabilities.resolution_width >= 160}" />
<wall:alternate_img nopicture="true"
    test="{not capabilities.gif}" />
<wall:alternate_img opwv_icon="magnifyglass"
    test="{capabilities.resolution_width <= 90}" />
```

```
<wall:alternate_img eu_imode_icon="#59100;"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img ja_imode_icon="#63873;"
    test="{capabilities.resolution_width <= 90}" />
</wall:img>
<wall:a href="http://url1" title="">Search</wall:a>
</wall:cell>

<wall:cell>
<wall:img src="pix/more1.gif" alt="More" >
<wall:alternate_img src="pix/more1_big.gif"
    test="{capabilities.resolution_width >= 160}" />
<wall:alternate_img nopicture="true"
    test="{not capabilities.gif}" />
<wall:alternate_img opwv_icon="rightarrow2"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img eu_imode_icon="#59030;"
    test="{capabilities.resolution_width <= 90}" />
<wall:alternate_img ja_imode_icon="#63735;"
    test="{capabilities.resolution_width <= 90}" />
</wall:img>
<wall:a href="http://url1" title="">More</wall:a>
</wall:cell>

</wall:cool_menu>

</wall:body> </wall:document>
```

Appendix G

Mobile Screenshots

Here we are presenting some screenshots of our application in order to illustrate the adaptations of the webshop that have been proposed for mobile devices.



Figure G.1: Viewing short product descriptions on the eSarine entry page in WML format.



Figure G.2: Scrolling down on the same WML page as in Figure G.1.



Figure G.3: Shopping cart on Openwave 7 in XHTML.



Figure G.4: Shopping cart implemented with the WALL tag library on Openwave 7.



Figure G.5: Login page in XHTML.



Figure G.6: Login page in WML format.