Multi-facade and Ubiquitous Web Navigation and Access through Embedded Semantics

Ahmet Soylu[1], Felix Mödritscher[2], and Patrick De Causmaecker[1]

[1] K.U. Leuven, Department of Computer Science, CODeS, iTec, Kortrijk, Belgium
{Ahmet.Soylu,Patrick.DeCausmaecker}@kuleuven-kortrijk.be
[2] Vienna University of Economics and Business,
Department of Information Systems, Vienna, Austria
felix.moedritscher@wu.ac.at

Abstract. Web content contains valuable information on the semantic structure of a site, which can be used to access and navigate the pages through ubiquitous computing environments. Semantic web approaches normally aim at modeling semantic relations and utilizing these models to provide enhanced functionality for humans or machines. In this paper we present an approach which focuses on using embedded semantics in order to achieve enhanced web access and navigation for the ubiquitous environments. Precisely we propose specifying and extracting microformat-based information within the web server and delivering it along the semantic structure of the site. We also describe our first prototype, the Semantic Web Component (SWC), and report on first experiences which evidence benefits in terms of less internet traffic and reducing the delivery of irrelevant information thus increasing the web accessibility as well as the navigability in ubiquitous environments.

Keywords: Ubiquitous Computing, Pervasive Computing, Embedded Semantics, Web of Data, Web Accessibility.

1 Introduction

The main motto of Ubiquitous Computing (UbiComp) [1, 2] deals with employing a variety of computing devices and applications, which are spread around the human environment, to seamlessly facilitate daily life through anytime and anywhere service and information access. These devices and applications need to communicate effectively so their behaviors and states can be synchronized (i.e. device/application interoperability). Furthermore, they need to share and understand available information to be able to deliver information and services relevant to users' context (i.e. data interoperability). The Web provides an appropriate framework respectively following two complementary approaches [3]: *(1) A communication-application space* which aims at enabling various mobile and stationary devices, including sensors and embedded devices, to get connected over the Internet. Consequently these devices can deliver their services to the each other and use available web applications and services while bringing them to the user environment (i.e. Web of Things [4]). *(2) An information space* which focuses on utilizing the Web as an ultimate information

T.-h. Kim et al. (Eds.): FGIT 2010, LNCS 6485, pp. 272–289, 2010.

source [5], envisioning a web environment being one huge virtual, readable and writeable database rather than a document repository (i.e. Web of Data [6]). Consequently, the so-called 'Semantic Web' aims at increasing the utility and usability of the Web by utilizing semantic information on data and services [7].

In this paper, we will focus on the Web as an information space. The devices in UbiComp environments are expected to interact with each other, so that, they form a functional unit, i.e. virtually representing a computer. Hence, this computing network requires processable data to be readily available, however present web environments rather correspond a document repository [6]. The Semantic Web suggests a set of standards to overcome this problem so that machine readable data can be provided through the Web. XML, RDF, and OWL have been widely used for exchanging messages, modeling the application context [8], and describing services etc. Although each of these languages aims at different purposes at different levels (e.g. syntactic or semantic), the main problems can be summarized as follows. *(1) Redundancy of the information*: Web information can be presented in two distinct facades: *(a) human readable facade* and *(b) machine readable facade* of the information. Structurally separating these two facades requires information to be duplicated both in the form of HTML and in the form of RDF, XML etc. thereby causing synchronization and consistency problems. *(2) Loss of simplicity:* The main reason behind the success of the Web is its simplicity; anyone can use a basic text editor to create a web page. Hence, creating an RDF or XML document and uploading that external file dedicated to a machine-readable use remains forbiddingly complex [9], decreasing the accessibility. A complete picture of the Web's full potential should consider its human impact, as people are the most significant components [6].

A response to such considerations is embedded semantics [3] - eRDF, RDFa and microformats - which allow in place annotation of information without coding an external XML or RDF document and without duplicating the information. However such a solution imposes an extra burden, that is, extraction. Embedded information needs to be extracted out from (X)HTML. Although there exists a variety of client side applications, like the Firefox add-on 'Operator' (http://addons.mozilla.org/en-US/firefox/addon/4106) which detects and extracts embedded information, the restricted resources (i.e. limited memory and screen size, limited internet connection bandwidth, limited processing power, etc.) of mobile and embedded devices available through UbiComp environments make extraction of semantic information from web pages a non trivial task; in particular if pages include a high amount of multimedia content as well as textual, informational and structural elements.

The Web is supposed to be the main information source for UbiComp environments, hence it is important to ensure web accessibility through different devices with varying technical capabilities. In order to countervail the aforementioned critical issue, this paper presents our solution proposal called Semantic Web Component (SWC) for the server side (see Figure 1) and a basic query language, namely Web Query Language (WQL). SWC enables users and devices to access and navigate websites along their semantic structure. Thus, (human and non-human) actors can interact with related information only, not being confronted by irrelevant content. It reduces the size of information to be transferred and processed drastically and fosters the visualization of websites on devices with smaller displays, thereby providing increased accessibility and an efficient integration into the UbiComp

environments. Moreover, WQL allows clients to submit basic queries through URL by HTTP GET method in order to retrieve only the content of interest while it is also used as a part of semantic navigation mechanism.

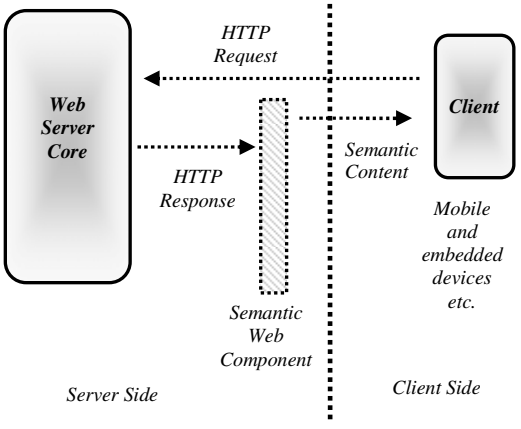

Fig. 1. The Semantic Web Component extracting the semantic content from the response

Since the component is an integral part of the web application server, any website hosted by such a server is covered by the functionalities of SWC. Such an approach is superior to the client side solutions in the sense of its uniformity and ease of employment by both the users and the machines. Furthermore since it imposes less responsibility to the client side, it might foster rapid employment of semantic web technologies and their standardization. The underlying technology utilized in SWC is built upon the embedded semantic technologies eRDF, RDFa and microformats. We also propose in this paper a description language for microformats to overcome its particular drawbacks (e.g. independence and extensibility) which in turn enables this technology to be compliant with our approach. Finally, the practical applicability of SWC is shown through the description of our first prototype implementation based on wide-spread concept of microformats and the description language.

The rest of the paper is structured as follows. In section 2, the role of embedded semantics in web technologies is briefly situated, and the language for microformats is described. The basic approach behind SWC with respect to current literature is explained in section 3. In section 4, the design and architecture of the SWC is presented. Section 5 evaluates our approach and discusses the related work. Finally section 6 concludes the paper and refers to our future work and its driving mantra.

2 Embedded Semantics

According to [10], the World Wide Web (WWW) is intended to be for humans while we believe that Semantic Web approaches rather lead to a more technologized Web, the 'Web for machines'. Although these two facades of the Web coexist, unification in the structural sense is possible. However the Web for humans should not be

compromised for the sake of the Web for machines. Embedded semantics imposes nearly no change in the current web technology and provides a simple and human-centered solution. Such an approach can be summarized by the four layers of information abstraction [2, 11]: *(1) storage layer* (e.g. tuples), (2) *exchange layer* (e.g. XML, JSON, RSS, etc.), (3) *conceptual layer* (e.g. OWL, RDF, etc.), and (4) *representation layer* (e.g. (XHTML, RDFa, eRDF, Microformats). The representation layer is not well studied, and the potential of the embedded semantics remains untouched to a large extent.

Human readable facade of the information through (X)HTML	Machine readable facade of the information through RDF	Uniform representation of the both facades through microformats
Geo: Latitude : 30.386142 , Longitude : 120.092834 	<rdf:RDF xmlns:rdf="…" xmlns:geo="…"> <geo:Point> <geo:lat>30.386142 </geo:lat> <geo:long> 120.092834 </geo:long> </geo:Point> </rdf:RDF>	Geo: 30.386142 , 120.092834

Fig. 2. Human readable facade, machine readable facade, and uniform facade of information

Embedded technologies use the attribute system of (X)HTML to annotate semantic information so that two facades of information are available in a single representation. Furthermore, employing the attribute system of (X)HTML allows developers to associate an external style sheet with the (X)HTML document to give any desired look and feel thereby loosely coupling the presentation and the information. In Figure 2, the first and the second code segments depict the human readable and machine readable facade of the information respectively. The last code segment demonstrates how these two facades can be combined into a single representation by means of embedded semantics. Embedded semantics approaches do not require altering current web technology and the approach itself is as easy as the Web itself since the required hand-on skills are modest. Apart from allowing machines to access machine readable information it also provides better user experience. For instance, users can export or copy some portion of the information from one web page to another one or an application by a single click [12]. In the followings we will elaborate on embedded semantic technologies.

Microformats: This community-driven approach provides a vocabulary and syntax to represent commonly known chunks of information such as events, people etc. Microformats use ‘class’, ‘rel’ and ‘title’ (X)HTML attributes to define domain specific syntaxes. It adopts well-known vocabularies such as vCard for hCard, iCal for hCal etc. Once its vocabulary and syntax are fixed, they should not be changed anymore. While Microformats can encode explicit information to aid machine readability, they do not address implicit knowledge representation, ontological analysis, or logical inference [13].

eRDF: eRDF also uses existing (X)HTML attributes 'class', 'rel' and 'title'. Unlike microformats it is based on the RDF framework that means it does not impose any pre-defined vocabulary. However it is not fully conformant to RDF.

RDFa: RDFa introduces new attributes 'about', 'resource', 'instanceof', 'property' and 'content'. These attributes are not yet supported by the current (X)HTML standard but expected to be included in the future. It is also based on the RDF framework and aims at reflecting the full capability of RDF.

While microformats are limited in flexibility, other techniques such as RDFa and eRDF provide more generic data embedding. Being based on RDF, eRDF and RDFa enable users to mix and use different name spaces. Microformats use a flat name space which is already predefined and cannot be extended or remixed. A microformat requires its own parser while generic parsers can be used with eRDF and RDFa. [14] lists four criteria for embedding semantic information. *(1) Independence and extensibility:* A publisher should not be forced to use a consensus approach, as she knows her requirements better. Web users originate from different communities, and thus follow their own local semantics for data interpretation and representation [12]. *(2) Don't repeat yourself (DRY):* (X)HTML should only include a single copy of the data. Hence modification can be done at one place which avoids consistency and synchronization problems. *(3) Locality:* When a user selects a portion of the rendered (X)HTML within his browser, she should be able to access the corresponding structured data (e.g. with a contextual menu). *(4) Self-containment:* It should be relatively easy to produce a (X)HTML fragment that is entirely self-contained with respect to the structured data it expresses.

Accordingly, an evaluation of technologies [14] is given in Table 1. eRDF has to provide vocabulary related information in the (X)HTML head while microformats either assume clients to be aware of all available syntaxes beforehand or require a profile URI to be provided for extraction. Microformats and eRDF lack self-containment because it is not possible to re-use eRDF or microformat information without requiring vocabulary specific information. On the other hand microformats lack of independence and extensibility since they are based on pre-defined vocabularies and they require a community consensus.

Table 1. An evaluation of embedded semantics technologies based on four main criteria

Criteria/ Technology	Independence and Ex.	DRY	Locality	Self-containment
RDFa	Yes	Yes	Yes	Yes
eRDF	Yes	Yes	Yes	Not fully
Microformats	No	Yes	Yes	Not fully

Although RDFa provides a far better solution in the technical sense, employment rates of these technologies do not seem to be in line with their technical merits. A recent estimate shows that microformats are used in hundreds of millions of web pages (http://microformats.org/blog/2007/06/21/microformatsorg-turns-2/) while deployment of eRDF and RDFa still remains weak. This is mainly because of its simplicity which might be the fifth criteria. On the client site, the user interaction paradigm is switching from passively consuming content (i.e. surfing on the Web) to actively contributing

(i.e. authoring/editing information on the Web) via weblogs, wikis, and user driven contents in general [12]. Therefore, having users as active contributors of the Web (e.g. as content authors, consumers and even as application developers) increases the demand for the simplicity. Particularly, microformats offer an easy mechanism for humans to publish information, and it lowers barriers for publishers by following a publisher-centric solution rather than a parser-centric one (see http://microformats.org/wiki/principles). These merits are due to the basic principles on which the research, design and development of microformats are based.

Since adoption of microformats is wide, we implemented our first prototype of SWC based on microformats. However it is important to note that eRDF and RDFa are compliant with overall idea and to be covered by SWC. However, lack of independence & extensibility and self-containment of microformats are important barriers for the SWC. This is because it is not possible to define custom microformats, and the consumer (i.e. SWC) is expected to have a pre-knowledge of the syntax and vocabulary of available microformats. Methodologically, there are two possibilities to describe microformat-based semantics embedded within the content. (1) Providing a XSL schema and, thus, describing what content chunks should be extracted in which particular way, as also done with GRDDL ('Gleaning Resource Descriptions from Dialect Languages') transformations [15]. (2) Developing an own simplified and generic way to specify embedded semantics. In this paper, we decided to follow the later approach. The first approach imposes dependence to the client by assuming that it supports XSL, and more importantly the client has to accept the extracted information in whatever form it is extracted to by XSL. We want to reduce complexity of the semantic description language and try to avoid describing how to extract the microformat-based semantics. However, a mapping from our data model to XSL can be easily made up. Such a description language provides a further layer of abstraction at the transformation side. Firstly, it allows custom microformats to be defined, thereby alleviating independence and extensibility problem. Secondly it allows clients to understand the exact structure of the available microformat rather than assuming that the client is already aware of all possible microformat syntaxes and vocabularies. Although this approach still does not fully satisfies self-containment, since existence of vocabulary and syntax specific information is required, it is superior to client pre-knowledge approach. Furthermore it allows the client to extract available information in any way to any form without imposing any technological dependence.

Allsopp lists 12 concrete examples of microformat specifications, beginning from elemental ones, like rel-license, rel-tag or VoteLinks, up to compound microformats, such as hCard, hCalendar or hAtom [16]. In practice there exist even more specifications (see www.microformats.org). In order to describe all these embedded semantics, a data model has to consider the following aspects in terms of required or optional attribute fields. *(1) Type:* The first issue to determine is if one wants to use an elemental or a compound microformat. *(2) Identifier:* Second, specifying embedded semantic requires, like all other resource description standards, some kind of identifier, so that applications or humans can differentiate between the semantic elements. *(3) Design pattern:* Third and mostly important, it is necessary to describe which design pattern one wants to address. Microformat design patterns comprise a formalism to 'reuse pieces of code that are generally useful in developing new

microformats' [16]. In other words, design patterns determine which (X)HTML elements and attributes are used to define a certain microformat. Thus, we propose to describe such a design pattern according to these two entities: (a) the element name, and (b) the attribute name. Assuming that a microformat is always based on an attribute, we consider the element as optional and the attribute as required. Furthermore, it should be possible to combine elements according to different attributes. *(4) Label:* A user understandable label which can be used while representing the extracted information, so different parts of the information can be identified by the users. Albeit not mandatory, it has an absolute use for SWC (see section 5). *(5) Matching string:* In order to restrict the (X)HTML attribute of the design pattern, an optional field for string matching is introduced. Values of the specified (X)HTML attributes are evaluated on basis of string equivalents as well as regular expressions. *(6) Scope:* The scope, again, is optional and restricts the scope of the semantics within the web-based content. If given, the embedded semantic is valid within all DOM elements specified by this field. *(7) Selector:* Another optional field, the so-called selector, is necessary to define from which source ((X)HTML element text or attribute) the semantics has to be extracted. If no selector is specified the value of the element is used. Otherwise, an application might retrieve the value of the specified selector which, for instance, could be the title attribute. *(8) Reference:* The optional reference field is of use to refer to another, existing microformat. Such a mechanism is useful for compound microformats, i.e. to include elemental microformats. If referring to another microformat, all other fields except the identifier are ignored. *(9) Optional:* Finally, the optional field indicates that an elemental microformat is optional within a compound one, which means that this element is not required to detect the compound microformat.

```
1  <microformats>
2    <elemental id="xfn_met" label="XFN: People I met in person" pattern="rel" match="friend met" select="text" />
3    <elemental id="vote" label="VoteLinks: Vote for me! " pattern="a:rev" match="vote-*" select="title" />
4    <elemental id="vote_link" label="Click here to vote" pattern="a:rev" match="vote-*" select="href" />
5    <elemental id="all_links" label="All external links" pattern="a:*" match="http://* " select="html/body" />
6    <elemental id="fn" label="hCard: Get full name" pattern="class" match="^fn  | fn | fn$" select="text" />
7    <elemental id="url" label="hCard: Two variants for URLs" pattern="a:relldiv:rel" match="url" select="href" />
8    <compound id="vevent" label="hCalendar: exemplary event" pattern="class" match="vevent">
9      <elemental id="vevurl" label="Event URL" ref="url" />
10     <elemental id="vevsummary"  label="Event summary" pattern="class" match="summary" select="text" />
11     <elemental id="vevstart" label="Start date" pattern="class" match="dtstart" select="title" />
12     <elemental id="vevend" label="End date" pattern="class" match="dtend" select="title" optional="true" />
13   </compound>
14   <elemental id="goal" label="AdeLE's learning goals" pattern="adele" match="to *" scope="/html/body/content"  />
15 </microformats>
```

Fig. 3. An example XML binding for the proposed microformat description language is given

Figure 3 shows a possible description language for microformat-based semantics. In this example, seven elemental microformats (lines 2 to 7 and line 14) and one compound one (line 8 to 13) are specified. The first elemental microformat, namely

'xfn_met' (line 2), stands for a particular type of the commonly-known (X)HTML Friends Network (XFN) specification which can be determined with the rel-attribute having the value 'friend met'. For extracting semantics from such elements, the text-field (the value between opening and closing tag) has to be used via the selector-field. If no selector is given, an information extractor simply uses the value of the specified pattern. Having such a specification of an elemental microformat, any application, even browser plug-ins can detect and extract this kind of semantic information from web-based content if supporting our data model.

3 The Semantic Web Component

In this section the basic idea behind the SWC is introduced. The driving challenges are twofold; *extraction* and *unification*. Firstly, the annotated information needs to be extracted out of the original content. Although the client side approaches are currently common, we will argue for a server side approach. Secondly, the fact that different embedded semantics technologies are available requires unifying the use of these technologies. We advocate this diversity and move unification to the sever side extraction mechanisms rather than opting for a single technology. We start with referring to the related literature with respect to these two points, so that the idea and our understanding can be situated on a concrete grounding. The literature provided in this section is not exhaustive, yet we have selected particular works characterizing our basic challenges.

In [2], the possible benefits of embedded semantics for the UbiComp environments are elaborated. Authors identify information as one of the important elements of their upper context conceptualization and state that embedded semantics is useful for representing different contextual characteristics of the information so that such contextually annotated information can be delivered to the users in an adaptive manner. Furthermore, they describe a web service which extracts and collects embedded information for learning resources from web pages. The harvested information is stored in a semantic database, allowing other clients to query its knowledge base through SPARQL queries. A similar approach has been employed in [17]. In the scope of earth observation [18] uses RDFa for identifying embedded information through a browser extension [19]. The information extracted is either used to populate ontologies with the extracted information or to be stored in the semantic repository. [2, 17, 18, 19] show that there exist different ways of harvesting embedded information. On the one hand, client side tools, such as Operator or Semantic Turkey, are used to extract or distinguish the annotated information from web content. The main drawback of these approaches is that they require a client side mechanism to extract information, so computing resources of the clients are used. Furthermore the whole content has to be downloaded to the target machine which is problematic due to the network load. On the other hand, third party web applications or services, as demonstrated in [2, 17], are utilized. In this case, the semantic search services provided usually duplicate the information by means of storing extracted information which is against one the driving principles of embedded semantics, namely the DRY principle. Furthermore, it imposes a dependency to other third party web applications or services. Clearly such approaches are not feasible for the

UbiComp environments since they are expected to include many small devices having low-bandwidth. Considering unification matter, in [14] proposes a mechanism, namely hGRDDL, to transform microformat embedded (X)HTML into its RDFa equivalent. This mechanism aims at allowing RDFa developers to leverage their existing microformat deployments. They advocate that such a solution can allow RDFa to be a unifying syntax for all the client-side tools. There are two important problems in this approach. First of all, developers need to provide vocabulary and syntax for each microformat to be transformed. Such a problem can be solved by using the description language which we have proposed in the previous section. However we disagree with unification by means of a unified syntax, indeed a technology not only a syntax, since a decision between microformats and RDFa is a tradeoff between simplicity (i.e. usability) and functionality.

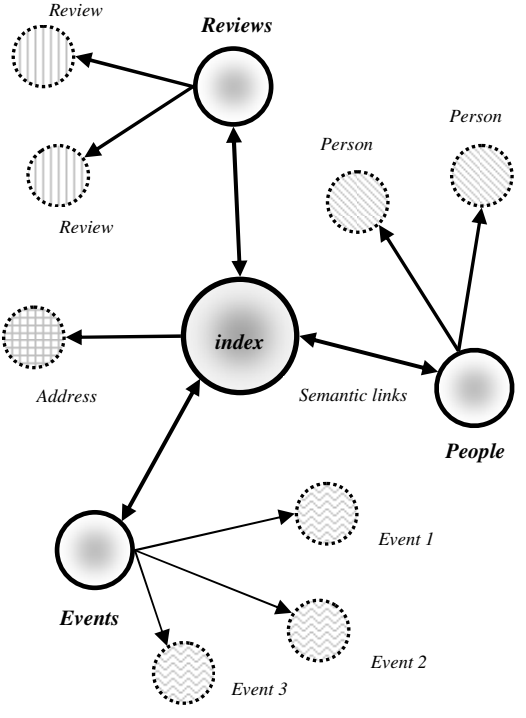

Fig. 4. Semantic information network (map) referring to semantic structure of a web site

Therefore we propose a solution proposal called Semantic Web Component (SWC) for web application servers. Since the Semantic Web is an important construct of the tomorrow's ubiquitous Web, application servers should be able to deliver two facades of the information directly and should allow both humans and machines to interact and navigate through them. SWC resides in the server side and observes the requests and the responses between the client and the server. When a client requests the machine readable facade of the information, instead of returning all the (X)HTML content it filters out only semantically annotated information. One option is that the

information is extracted and represented in a (X)HTML form (i.e. reduced (X)HTML content). All other information, which is not annotated, is simply discarded. Such an approach treats the pages of a website as a set of nodes where each node might contain instances of several types of embedded information. Embedded information can be elemental, including only one single and independent chunk of information, or compound consisting of at least two elemental embedded information. Each node (i.e. page) also has links to other pages having embedded information. We call these links *semantic links*. The approach, we named it *semantic information network* (or map), is visualized in Figure 4. This facade is still the human facade, however it represents reduced content and allows user to navigate through the semantic information network of a website. On the other hand, if the machine asks only for machine readable facade, the component converts extracted information to the XML or RDF.

We summarize advantages of such an approach with respect to the aforementioned works in the followings. *(1) Direct and seamless access* to different facades of the information without imposing any burden to the client side, e.g. no need for data extraction. *(2) Enhanced user experience:* users are usually lost in the abundant information space of the Web where valuable information is hidden in the information sea and presentational and structural elements. Users can simply access the information they do require. *(3) Increased accessibility:* mobile and embedded devices in the UbiComp environments can use both facades of the information. (X)HTML representation of the reduced information will enable them to deliver web information to anyplace while machine readable form of the information will enable devices to process and use the web information. *(4) Higher network efficiency:* the device do not need to retrieve all the (X)HTML content from the server, hence the amount of information travelling in the network decreases. *(5) Centralized solution:* it does not impose use of a common syntax, technology or dependency to any other service; everything is unified at the server side.

We introduce three particular scenarios to demonstrate the use and benefits of such an approach.

Scenario-1: A website of a cinema company provides recommendations for the movies of the season. The site consists of following pages: 'Events', 'People', and 'Reviews'. Each movie is considered as an event in the 'Events' page. 'Reviews' page includes the reviews about the movies and each review is provided by a registered reviewer. The 'People' page contains information about the registered reviewers. The information on these pages is annotated by using and hCal, hReview and hCard microformats respectively. Accordingly the semantic information map of the website is shown in Figure 4. A user wants to see a movie tonight. He does not have much time to surf through the website to find a proper movie. Furthermore, he only has his mobile phone around which has internet access. However his mobile device's connection and screen properties are at a low level. Since the website is hosted by a server which has SWC enabled, the user simply sends a request through his mobile phone. His browser implicitly tells the server that it only requests annotated information. Server returns to the user the list of semantic information available in the index page; people, events, and reviews. The user selects the reviews option to see the movie reviews, and the server returns the list of available instances which are identified with the titles of the movies. The user selects a movie in which he might be interested and reads the reviews. He really likes one of the reviews and wants to see

who wrote it to be sure that he can trust the quality of this information. He navigates back to the first page and repeats same procedure to see the details of the reviewer. The user decides that the movie is worth to go and the reviewer is really appropriate. Then he goes through the events page to see the schedule.

Scenario-2: Based on the previous scenario, the company wants to place small terminals to some particular places through its main hall. These terminals are expected to provide basic information available in their Web page; events (i.e. movies), reviews (i.e. movie reviews) and people (i.e. reviewers). However the budget of the company is limited to buy only low cost devices which only have text-based presentation capabilities, besides this is the only desired functionality of the company. These devices are connected to the cinema's website through normal internet connection. The browser implicitly tells server that it requires semantically annotated information; hence these devices provide the same navigational mechanism as described in the first scenario.

Scenario-3: A recommender system suggests activities to users; therefore it has access to their agendas and profiles. To find appropriate activities, the recommender system harvests the embedded information from various websites in the same way mentioned in the first scenario. It uses ontological reasoning and has a terminological base providing the upper and domain ontologies as well as a knowledge base with the ontology instances. The terminological base is already pre-defined while instances are collected on the fly through harvesting semantic information from different websites. Since the harvested websites are supported by SWC, the recommender agent does not need to be aware of different embedded technologies. The machine readable facade of the information is directly sent from the servers. The recommender system reasons that the user has nothing scheduled on Saturday night and she is keen on horror movies. The agent submits WQL queries to various entertainment sites asking for events scheduled for Saturday night. According the information harvested from the cinema's website, it finds out that there is a new horror movie which is highly ranked on this Saturday night. Therefore this movie is recommended to the user. Obviously there are many other possibilities for recommendations, like a ranked list of the events or the most topical reviews.

4 Design and Implementation

We have set multi-facade and ubiquitous web navigation and access into practice by implementing a prototypic SWC component which includes the microformat descriptors, i.e. the microformat descriptions of the embedded semantics to address, and the semantic extraction functionality. Figure 5 demonstrates the architecture of our component which is realized in the form of Apache modules. Thereby a module is a self-contained plug-in which may implement core functionalities, a general purpose service, a small but vital function or a single purpose application of the Apache Web Server [20].

The SWC component is composed of two modules, namely *mod_semantic* (i.e. the handler, which provides the functions to be performed when URL requests are sent to the server – see http://httpd.apache.org/docs/2.0/handler.html) and *mod_grddl* (i.e. the

filter, which processes the data sent or received by the server – see http://httpd.apache.org/docs/2.0/filter.html). The former module implemented in C listens to the client requests and provides the appropriate functionality. The latter module is responsible for information extraction. The client is expected to send a contextual header value named 'is_semantic' which maps to three distinct modes. The current implementation is based on PHP, for simplicity, thus it works more like a proxy. The modes and the corresponding actions are described in the following.

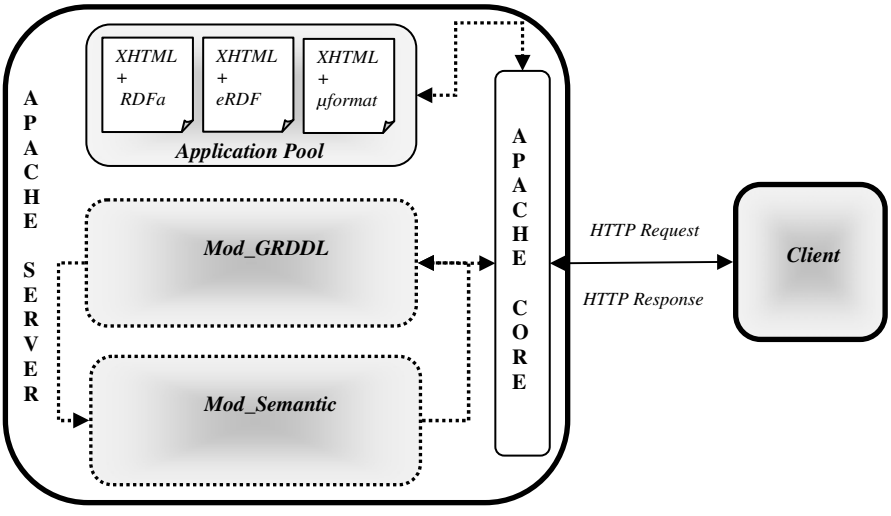

Fig. 5. Overall architecture of the component within the Apache Web Server is depicted

Case-1 (Mixed facade mode): If the value of the contextual header element is equal to '0', the handler module calls the requested resource and delivers it as it is. This case does not change original behavior of the Web server. The returned (X)HTML document also includes embedded information, therefore this mode is called as mixed facade mode.

Case-2 (Machine facade mode): If the value of the contextual header element is '1' then the handler module calls the original resource, and forwards it to the filter module (e.g. after being generated or for dynamic content such as a PHP script). The filter module extracts and delivers embedded information in as RDF or XML and according to the descriptors of the semantics available within the source document.

Case-3 (Human facade mode): If the value of the contextual header element is equal to the '2', the handler module calls the original resource and forwards it to the filter module. The filter module retrieves the descriptor within the source (X)HTML document, extracts the embedded information in the form of (X)HTML and forwards the response. This approach enables user to move inside the semantic (X)HTML structure, i.e. semantic information map previously shown in Figure 4. We also call (X)HTML snippet returned to be the reduced (X)HTML content since it is still represented in (X)HTML but the non-annotated parts of the document are discarded.

In the last mode (i.e. case-3), the content retrieval is iterative. For instance, if a user initiates navigating a website through a page EP (*entry page* or *entry node*: the first accessed page, which does not necessarily need to be the index page) the component returns a set of available types of embedded information and the number of instances available for each type in the EP. The returned message will be a simple (X)HTML response which presents human understandable labels of extracted microformats which are available through the label attributes of the descriptor. If the user selects a type of embedded information, the SWC returns the list of available instances together with a small description, e.g. title of each instance. The user can further select an instance; then the information available for this instance is returned as a response. If the instance is one elemental embedded information or a set of several elemental embedded information the whole content is displayed. However if the embedded information includes another compound embedded information the user needs to further drill down. Up to this point all the navigation within the semantic information was available in the EP. However users can also navigate into the other pages, which is only possible by annotating links which reference to the other pages involving semantic information and omitting the non-annotated links. In order to achieve annotated links, SWC uses the 'nav_link' attribute value which also can be seen as a type of embedded semantics to annotate links between the pages. Rather than fetching the whole content of the page or all the available semantic information through the page, this approach allows user or machine to partially retrieve semantic information. The same iterative procedure can also be applied for the machine facade mode if required.

```
<p class="title">Information available: Index
    <p class="type">
        <a href="www.xxx.com/index.php?WQL=(Id:hCard) ">  People  (8)   </a>
    </p>
    <p class="type">
        <a href="www.xxx.com/index.php?WQL=(Id:hCal) ">   Events (4)   </a>
    </p>
</p>
```

```
<p class="title">Information available: People
    <p class="fn name">
        <a href="www.xxx.com/people.php?WQL=(Order:1) ">  Person name and Surname  </a>
    </p>
    <p class="fn name">
        <a href="www.xxx.com/people.php?WQL=(Order:2) ">  Person name and Surname  </a>
    </p>
</p>
```

Fig. 6. Basic (X)HTML templates for blank and parametric calls are depicted respectively

In order to realize such a navigational procedure, we have introduced a simple *presentational template* and a query language named *Web Query Language* (WQL). The basic presentational template is composed of '<p>' and '<a>' elements. It is used to only deliver the information at the most basic level. The class attribute system is still utilized to annotate the information, so any desired look and feel can be given to

the information retrieved by means of CSS. In Figure 6 an example is depicted. The first part of the figure belongs to the initial call of the entry page. Hence detected types of embedded information and the number of the instances available are listed. The second part of the figure represents a call to a particular type of embedded information in the corresponding page. This time available instances and their descriptive titles are displayed. The navigation is done by using HTTP GET through the URLs available in the basic presentation. Each get method submits a basic WQL query which either includes the unique id (i.e. the type) of the embedded information or the order number of the instance. So a user can navigate into a specific type of embedded information or to a specific instance. When there is no WQL query parameter given in the URL, we name it a *blank call* (e.g. initial call to the entry page). Conversely, if some WQL parameters are defined, the call is named as *parametric call*. The motivation for and the details of WQL are explained in the followings.

The proposed query language arose from the need to identify the type and instances of embedded information to be navigated, e.g. to enable a user to see the instances of people available at the cinema website or to select one specific person. The basic structure of a WQL query is as follows:

[URL]?WQL=([conditions])

The WQL query is provided right after the requested URL with a reserved GET parameter 'WQL'. The conditions have to be specified inside the bracket symbols '(' and ')'. The constructs of the query language are listed and explained in Figure 7.

Symbols	Description
:	Usage: <*elementName*> : <*characterString*> This symbol works as "LIKE" operator in SQL. It is used to describe if a pattern matches given character string. Example WQL query: *WQL=(fn_name:"Tommy")*
;	Usage: <*firstCondition*> ; <*SecondCondition*> This symbol corresponds to AND operator in SQL. Example WQL query: *WQL=(fn_name:"tommy";fn_surname:"Brown")*
,	Usage: <condition_1> , <condition_2> This symbol corresponds to OR operator in SQL. Example WQL query: *WQL=(fn_name:"Tommy",fn_name:"Alice")*
Reserved variables	**Description**
Order	Usage: *Order:* <*positiveInteger*> This reserved variable behaves as a predefined unique key which spans all the instances of different types of embedded information available through a single page. Example WQL query: *WQL=(Order:2)*
Id	Usage: *Id:* <*string*> This reserved variable is used for accessing a specific type of embedded information. Example WQL query: *WQL=(Id:hCard)*

Fig. 7. The basic constructs of the Web Query Language (WQL) is given

WQL realizes the most basic facilities of a SQL like query language. We have introduced three symbols which represent the 'LIKE', 'AND', and 'OR' operators of SQL. When the ':' operator is used, the wildcard character '%' is automatically added

to the both sides of the string, which indicates a string of any length. However when applying it with reserved variables, it works in the same way as the equality operator in order to prevent any ambiguity (e.g. Id:1 vs. Id:12). The element name used with the ':' operator can refer to any of the words in the vocabulary and to any type of the embedded information available. We have also introduced two reserved variables which are 'Id' and 'Order'. The 'Id' variable is used to match with the 'id' attribute in the microformat description language to identify the type of embedded information thereby allowing a particular type of embedded semantics to be retrieved. The 'Order' variable is used to access a specific instance by pointing to a unique number for each instance. This unique value is derived based on the assumption that the order of each instance occurring in a page remains constant, and it is in increasing order depending on the place of the instance. Only reserved variables start with an upper case letter, and the client can provide as many variable-value pairs desired within a WQL query to be matched with the vocabularies of the embedded information. Although the current implementation of the WQL is mainly for navigational purposes, it will be further developed and validated, so it can be used for querying Web pages through URLs at the most basic level as demonstrated in Figure 7. Therefore WQL is intended to be simple. Since WQL is less expressive than SPARQL underlying implementation can be realized through mapping WQL queries to SPARQL queries in order to have a standardized implementation. Search/Retrieval Using URL (SRU - http://www.loc.gov/standards/sru/) and Yahoo! Query Language (YQL - http://developer.yahoo.com/yql/) are similar approaches to WQL. The former focuses on XML and the latter is proprietary since every query is executed through their central API. WQL approach assumes any server to be able to execute WQL queries through SWC. In terms of expressivity, WQL is less expressive than YQL and SRU, however this is because we opt for simplicity at this point. Further extensions to WQL is mainly intended to follow SRU since it is based on a similar simple syntax while YQL follows a more complex SQL/SPARQL like syntax.

5 Evaluation and Discussion

A preliminary evaluation of the component and the description language has been done for a website containing real-world data about a research group. The website includes two types of embedded information, precisely people (i.e. research members) and events (e.g. seminars), comprising 26 instances of people and 15 instances of events. These instances are embedded in 31 pages of the website. The result of a blank request is visualized on the left-hand side of Figure 8, while the result of the request for all people instances is shown on the right-hand side.

The most basic evaluation of our approach can be done from an UbiComp perspective in twofold: *(1) network traffic:* comparison of the amount of information downloaded in the mixed mode and the amount of information downloaded in the human facade mode, *(2) network calls:* comparison of the amount of page requests while user is navigating in mixed mode and the amount of the page requests while user is navigating in the human facade mode. The measurements are based on the fact that all the available semantic information instances need to be retrieved in both facades of the navigation and within one single session for each of the facades.

This case study comparing the human face mode to the mixed mode (full web application) of SWC evidences one important benefit of our approach: On the one hand, the difference between the amount of information transferred during mixed facade navigation and the amount of the information transferred in human-facade mode is drastic. Due to the fact that a page with all its basic presentational markups contains typically around 27 KB of data in average (without considering the multimedia content!), in the first mode a total amount of 849.2 KB data is downloaded. In the second mode, however, the data transferred is reduced to around 110 KB, as a chunk of embedded information has a size of 1-2 KB in average. On the other hand, the number of network calls done in the two sessions increases from 31 calls in the first facade to 51 for the second facade. The difference depends on the structure of the website and structure of the embedded information available. Overall, the increase in the amount of network calls seems admissible since the amount of information downloaded in each call is considerably small. The significant reduction of transferred data clearly favors our component.

Fig. 8. Blank and parametric requests to the research group web site are shown respectively

Due to the navigation along the semantic structure of a website, we see clear advantages for mobile devices and web accessibility. One the one hand, less data is transferred to the web client. On the other hand, irrelevant information is filtered out. The second issue however could be problematic for paid advertisements. In the literature there are several studies addressing web access through mobile devices with limited sources. Amongst others, [21] reports on website personalizers observing the browsing behavior of website visitors and automatically adapting pages to the users. Moreover [22] examine methods to summarize websites for handheld devices. In [23] authors employ ontologies (OWL-S) and web services in order to realize context-aware content adaptation for mobile devices. All these approaches either require authoring efforts, e.g. for creating ontologies, or are based on AI-based techniques which cost a considerable amount of computational processing. Our approach on the

other side builds upon a simple specification of semantics embedded on a website and low processing efforts done by the web server. Anyhow the SWC enables users to access and navigate web content along their semantic structure thus reducing the traffic and providing personalized chunks of information, even for mobile devices.

6 Conclusions and Future Work

In this paper we argued for using embedded semantics (i.e. microformats) for UbiComp. Instead of building upon ontologies or complex mining techniques we proposed a description language for microformat-based information which is used by two Apache web server modules (the SWC) to enable users to access and navigate web content along the semantic structure of a website. In a first evaluation study we evidenced that SWC can reduce internet traffic as well as irrelevant content thus increasing its applicability for UbiComp environments and mobile devices.

The work presented in this paper serves as a proof of the concept, indicating the advantage of the overall approach. Accordingly, our future work involves exhaustive validation of the whole approach with a particular focus on usability. E-learning is one of our immediate application domain since our main research challenge is enabling adaptive ubiquitous learning, requiring us to ensure the accessibility of the web-based learning environments. Finally, we envision going beyond semantic and ubiquitous web navigation and extending our approach with respect to user interactions and user-driven development of web environments.

Acknowledgments. This paper is based on research funded by the Industrial Research Fund (IOF) and conducted within the IOF Knowledge platform 'Harnessing collective intelligence in order to make e-learning environments adaptive' (IOF KP/07/006). Partially, it is also funded by the European Community's 7th Framework Programme (IST-FP7) under grant agreement no 231396 (ROLE project).

References

1. Satyanarayanan, M.: Pervasive computing: vision and challenges. Pers. Commun. IEEE 8, 10–17 (2001)
2. Soylu, A., De Causmaecker, P., Desmet, P.: Context and Adaptivity in Pervasive Computing Environments: Links with Software Engineering and Ontological Engineering. J. Softw. 4, 992–1013 (2009)
3. Soylu, A., De Causmaecker, P., Wild, F.: Ubiquitous Web for Ubiquitous Computing Environments: The Role of Embedded Semantics. J. Mob. Multimed. 6, 26–48 (2010)
4. Dillon, T., Talevski, A., Potdar, V., Chang, E.: Web of Things as a Framework for Ubiquitous Intelligence and Computing. In: Zhang, D., Portmann, M., Tan, A.-H., Indulska, J. (eds.) UIC 2009. LNCS, vol. 5585, pp. 1–10. Springer, Heidelberg (2009)
5. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Sci. Am. 284, 34–43 (2001)
6. Ayers, D.: The Shortest Path to the Future Web. Internet Comput. 10, 76–79 (2006)
7. Mödritscher, F.: Semantic Lifecycles: Modelling, Application, Authoring, Mining, and Evaluation of Meaningful Data. Intl. J. of Knowl. Web Intell. 1, 110–124 (2009)

8. Perttunen, M., Riekki, J., Lassila, O.: Context Representation and Reasoning in Pervasive Computing: a Review. Intl. J. Multimed. Ubiquitous Eng. 4, 1–28 (2009)
9. Khare, R.: Microformats: the next (small) thing on the semantic Web? Internet Comput. 10, 68–75 (2006)
10. Huang, W., Webster, D.: Enabling context-aware agents to understand semantic resources on the www and the semantic web. In: IEEE/WIC/ACM Conf. on Web Intelligence (WI 2004), Beijing (2004)
11. Reichle, R., Wagner, M., Khan, M.U., Geihs, K., Lorenzo, L., Valla, M., Fra, C., Paspallis, N., Papadopoulos, G.A.: A comprehensive context modelling framework for pervasive computing systems. In: 8th IFIP Intl. Conf. on Distributed Applications and Interoperable Systems, Oslo, Norway (2008)
12. Mrissa, M., Al-Jabari, M., Thiran, P.: Using microformats to personalize web experience. In: 7th Intl. Workshop on Web-Oriented Software Technologies IWWOST 2008 (2008)
13. Khare, R., Çelik, T.: Microformats: A pragmatic path to the Semantic Web. In: 15th Intl. World Wide Web Conf., Edinburgh, pp. 865–866 (2006)
14. Adida, B.: hGRDDL: Bridging micorformats and RDFa. J. Web Semant. 6, 61–69 (2008)
15. Connolly, D.: Gleaning Resource Descriptions from Dialects of Languages. W3C (2004), http://www.w3.org/2004/01/rdxh/spec
16. Allsopp, J.: Microformats: Empowering Your Markup for Web 2.0. FriendsofED, Berkeley (2007)
17. Sabucedo, L.A., Rifón, L.A.: A Microformat Based Approach For Crawling And Locating Services In The Egovernment Domain. In: The 24th Intl. Symposium on Computer and Information Sciences, pp. 111–116. IEEE Press, Guzelyurt (2009)
18. Fallucchi, F., Pazienza, M.T., Scarpato, N., Stellato, A., Fusco, L., Guidetti, V.: Semantic Bookmarking and Search in the Earth Observation Domain. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part III. LNCS (LNAI), vol. 5179, pp. 260–268. Springer, Heidelberg (2008)
19. Griesi, D., Pazienza, M.T., Stellato, A.: Semantic Turkey - a Semantic Bookmarking tool (System Description). In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 779–788. Springer, Heidelberg (2007)
20. Kew, N.: Apache Modules Book: Application Development with Apache. Prentice Hall, Englewood Cliffs (2007)
21. Anderson, C.R., Domingos, P., Weld, D.S.: Personalizing Web Sites for Mobile Users. In: The 10th Intl. World Wide Web Conf., pp. 565–575. ACM, Hong Kong (2001)
22. Buyukkokten, O., Garcia-Molina, H., Paepcke, A.: Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices. In: The 10th Intl. World Wide Web Conf., pp. 652–662. ACM, Hong Kong (2001)
23. Forte, M., de Souza, W.L., do Prado, A.F.: Using Ontologies and Web Services for Content Adaptation in Ubiquitous Computing. J. Syst. Softw. 81, 368–381 (2008)