Integrated Computer-Aided Engineering 19 (2012) 93–109
DOI 10.3233/ICA-2012-0393
IOS Press

Ubiquitous web navigation through harvesting embedded semantic data: A mobile scenario

Ahmet Soylu[a], Felix Mödritscher[b] and Patrick De Causmaecker[a,*]
[a]*Department of Computer Science, ITEC-IBBT, CODeS, K.U. Leuven, Kortrijk, Belgium*
[b]*Institute for Information Systems and New Media, Vienna University of Economics and Business, Vienna, Austria*

Abstract. In this paper, we investigate how the Semantic Web can enhance web navigation and accessibility by following a hybrid approach of document-oriented and data-oriented considerations. Precisely, we propose a methodology for specifying, extracting, and presenting semantic data embedded in (X)HTML documents with RDFa in order to enable and improve ubiquitous web navigation and accessibility for end-users. In our context, embedded data does not only contain data type property annotations, but also object properties for interlinking, and embedded domain knowledge for enhanced content navigation through ontology reasoning. We provide a prototype implementation, called Semantic Web Component (SWC) and evaluate our methodology along a concrete scenario for mobile devices and with respect to precision, performance, network traffic, and usability. Evaluation results suggest that our approach decreases network traffic as well as the amount of information presented to a user without requiring significantly more processing time, and that it allows creating a satisfactory navigation experience.

Keywords: Embedded semantics, linked data, ubiquitous computing, mobile web, ontologies

1. Introduction

In recent years, researchers have paid increasing attention and effort to the Semantic Web. Tim Berners-Lee and his colleagues have formulated the vision of the Web as a universal medium for data, information, and knowledge exchange [6]. The so-called 'Semantic Web' aims at increasing the utility and usability of the Web by utilizing semantic information on data and services [24]. Generally, Semantic Web approaches build upon specifications for modeling and expressing web semantics [17], e.g., the Resource Description Framework (RDF), different data interchange formats (RDF/XML, N3, N-Triples, etc.), the Web Ontology Language (OWL), and the forth. However, less attention is paid to embedded semantics (e.g., RDFa, eRDF, microformats, and microdata). Existing approaches, like microformats or various harvesting solutions [2, 28], are normally restricted to pre-defined and widely accepted formats with a specific focus on structure of the data or to specific analysis techniques like text summarization or latent semantic analysis etc. [19].

On the one hand, to a large extent, the work done has focused on challenges regarding machine consumption of semantic data while less attention has been paid to the perspective of human consumption of web semantics, e.g., the enhancement of web navigation. Yet, observable efforts aiming at the provision of user-friendly means for displaying and browsing available semantic data either address expert level users (i.e., developers) or content publishers with data-centric approaches (e.g., the generation or validation of data mashups). Therefore the key challenge how to utilize Semantic Web technologies to improve end-user functionality remains uncovered.

On the other hand, although there exists a variety of client side applications, like the Firefox add-on named "Operator" which detects and extracts embedded information, the restricted resources (i.e., limited memory, screen size, internet bandwidth, processing power, etc.) of mobile and embedded devices available within Ubiquitous Computing (UbiComp) [34] environments make extraction of semantic data from web pages a non trivial task; in particular if pages include a high amount

*Corresponding author: Patrick De Causmaecker, Department of Computer Science, K.U. Leuven Campus Kortrijk, Etienne Sabbelaan 53, 8500 Kortrik, Belgium. Tel.: +32 0 56 246002; Fax: +32 0 56 246052; E-mail: Patrick.DeCausmaecker@kuleuven-kortrik.be.

ISSN 1069-2509/12/$27.50

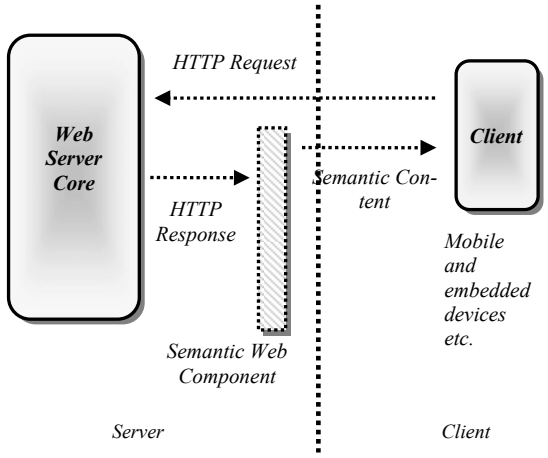

Fig. 1. The semantic web component.

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 accessibility for different devices with varying technical characteristics.

Above and beyond, this paper aims at using embedded semantic data to enable and enhance ubiquitous web navigation and accessibility by following a hybrid approach merging document-oriented and data-oriented considerations, addressing devices with different characteristics and considering usability matters like the cognitive load of users. In order to countervail the aforementioned critical challenge, we examine how embedded semantics can be used to access and navigate web-based content according to its internal structure and semantics described by embedded data. Precisely, we propose a methodology for specifying, extracting, and presenting semantic data embedded in (X)HTML through RDFa for the end-users. In the course of this work, embedded semantic data is not only considered to be structured (i.e., with types and data type properties), but also to be linked/related (i.e., interlinked with object type properties) [8]. We further assume that web-based content embeds domain-specific knowledge (limited with subclass and type relations at the moment) which can be used to improve content navigation through basic ontology reasoning. Consequently, we describe the technical realization of the proposed methodology through a server-sided prototype called Semantic Web Component (SWC), see Fig. 1, and evaluate our approach along a concrete scenario for mobile devices and with respect to precision, performance, network traffic, and usability. Evaluation results suggest that our approach decreases network traffic as well as the amount of information presented to a user with a fair processing time, and that it allows creating a satisfactory navigation experience.

The proposed methodology 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 with irrelevant content. The costly extraction process takes place at the server side. 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.

The rest of paper is structured as follows. Section 2 criticizes the embedded semantics technologies with respect to the requirements of the proposed methodology while Section 3 reports on related work. Section 4 describes the overall approach for Semantic Web navigation. In Section 5, we present and elaborate on the proposed methodology and describe design and implementation of a first prototype. Then, Section 6 evaluates the computational feasibility and usability aspects of the methodology before Section 7 discusses further work and concludes the paper.

2. Embedded semantics

Web content can be presented in two distinct facades: (a) *human readable facade* and (b) *machine readable facade* of 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. 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. Embedded semantics, eRDF, RDFa, microformats, and microdata [1,5,18,20], enables publishing machine readable structured data about non-information sources (i.e., things in the world) from diverse domains such as people, companies, books, products, reviews, genes etc. within (X)HTML documents.

In [1], four criteria are listed for embedding semantic information. (1) *Independence & extensibility*: A publisher should not be forced to use a consensus approach, as she knows her requirements better. (2) *Don't repeat your-self* (*DRY*): (X)HTML should only include a single copy of the data. (3) *Locality*: When a user selects a portion of the rendered (X)HTML within his browser, she should be able to access the correspond-

ing structured data. (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.

Among these four criteria, independence & extensibility and self-containment are important for the overall approach. eRDF has to provide vocabulary related information in (X)HTML head while microformats either assume the client to be aware of all available syntaxes beforehand or a profile URI is 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 independence & extensibility since they are based on pre-defined vocabularies and they require a community consensus.

Apart from encoding explicit information to aid machine readability, support for data interlinking [8] and implicit knowledge representation (i.e., sharing domain knowledge), thus ontological analysis, or logical inference [27] are of importance, particularly, although not a strict requirement, our approach benefits from these merits for enhancing the navigation experience. Therefore implicit knowledge representation can be considered fifth criterion. Microformats do not address implicit knowledge representation, hence logical inference and ontological analysis [21] while eRDF is not fully conformant with the RDF framework. Data (inter-)linking, regarded as a sixth criterion, enables linking different data items either within the local information source, or in a broader sense, across external information sources. This requires data items included in (X)HTML to have their own identifiers (i.e., HTTP URIs), which is missing in microformat approach preventing relationship assertions between data items [8].

An evaluation of embedded semantics technologies is given in Table 1 with respect to the six criteria.

3. Related work

Related work can be discussed under two main dimensions: The first one is the navigation which can be further categorized in terms of data vs. document-oriented approaches and generic vs. domain-specific approaches. The second dimension is data extraction in terms of client-based vs. server-based approaches.

Considering the first dimension, traditional hypertext browsers follow a document-centric approach by allowing users to navigate forward and backward in a document space through HTML links. Similar to the document network of the current web, linked data creates a global data graph through RDF links connecting different data items. Result of this data-centric approach is that a user may begin navigation in one data source and progressively traverse the Web by following RDF rather than HTML links [8].

Consequently, several generic linked data browsers [7], following this data-centric approach, have been developed. Zitgist (http://zitgist.com/), Tabulator (http://www.w3.org/2005/ajar/tab), Marbles (http://marbles.sourceforge.net/), Disco (http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/), Dipper (http://api.talis.com/stores/iand-dev1/items/dipper.html), sigma [31] (http://sig.ma), jSpace (http://www.clarkparsia.com/jspace/), and Exhibit (http://www.simile-widgets.org/exhibit/) are the most notable ones. Although a data centric approach gives users the advantage of discovering new data sources, they are not tailored for the end-users. Users might face a heavy cognitive load, thereby loosing the focus, since navigation might involve other data sources which are not explicitly selected. Normally, a website is organized into a set of information sources (i.e., (X)HTML documents) where each information source stands as a container for non-information sources (i.e., entities). Linking through these documents is established in a user-centric way rather than a data-centric one in order to provide a smooth navigation experience. Therefore a hybrid methodology merging document-centric and data-centric approaches should be more appropriate. Furthermore, generic linked data browsers usually omit supportive metadata required to improve utility of embedded data for human users. For instance, data type properties, object properties etc. are presented to the users with their machine-targeted identifiers. Most of the time these identifiers are not well suited for human consumption (since they are not intended to be), thereby necessitating a standardized metadata layer providing required labeling and generic knowledge.

Apart from generic linked data browsers, several domain-specific viewers for linked data have also been developed. Amongst others [10], proposes a Semantic Web portal for supporting domain users in organizing, browsing and visualizing relevant semantic data. In [4], a template-based approach is followed for improving syndication and use of linked data sources. The domain-specific nature of these approaches allows them to present data in a form suited to its characteristics (e.g., for instance geographical data can be presented in a map). However the need for the domain knowl-

Table 1
An evaluation of embedded semantics technologies is given

Criteria/ technology	Independence & Ex.	DRY	Locality	Self-cont.	Implicit Know. Rep.	Data linking
RDFa	√	√	√	√	√	√
Microdata	√	√	√	√	√	√
eRDF	√	√	√	Not fully	Not fully	√
Microformats	×	√	√	Not fully	×	×

edge restricts these approaches to content producers and mashups for specific presentation environments. The goal of our work is to come up with a generic data browser that can present any type of custom or de-facto standardized embedded data on different devices.

There are several studies, out of linked data community, addressing web access through mobile devices with limited sources. Amongst others [3], reports on website personalizers observing the browsing behaviors of website visitors and automatically adapting pages to the users. Moreover [9] examine methods to summarize websites for handheld devices. In [12], authors employ ontologies (OWL-S) and web services to realize context-aware content adaptation for mobile devices. These approaches either require authoring efforts, e.g., for creating ontologies, or are based on costly AI-based techniques.

Considering the second dimension (i.e., extraction), we highlight work particularly characterizing the basic challenges. In [28], the authors describe a web service extracting and collecting 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. In the scope of earth observation [11], apply RDFa for identifying embedded information through a browser extension [14]. The information extracted is either used to populate ontologies or stored in a semantic repository. In [11,14,28], it is shown 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 is that these approaches require a client side mechanism to extract information hence 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 [28], can be utilized. In this case, semantic search services usually duplicate the information by means of storing extracted information separately, which violates the DRY principle. It also imposes a dependency on other third-party web applications or services. Clearly such approaches are not feasible for the UbiComp environments with various tiny devices.

Regarding embedded semantic technology to use [1], proposes a mechanism for unification, 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. Authors advocate that their proposal can allow RDFa to be a unifying syntax for all client-sided 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. Although the description language that we proposed earlier [29] can solve this problem, we disagree with unification by means of a unified syntax since a decision between microformats and RDFa is a tradeoff between simplicity (i.e., usability) and functionality.

4. Proposed solution approach

The overall approach requires that, at the server-side, requests and the responses between the client and the server be observed. When a client initiates a semantic navigation request for a page of a website, semantically annotated information (i.e., embedded data) is filtered out instead of returning all the (X)HTML content. The extracted information is presented as an (X)HTML document (i.e., reduced content). All non-annotated information is simply discarded. This facade is still the human facade, however it allows users to navigate through the semantic information available in a website by following data links and relevant HTML links. Each (X)HTML page might contain links referring to other pages of the site having embedded information. Such links are also annotated and called semantic links. Such an approach considers a website as a graph, and the pages of it as a set of nodes where each node represents a sub-graph containing instances of several types. Data items (i.e., instances) are composed of data type properties associating them with data literals, and object properties linking them to each other. The embedded data

together with semantic links (i.e., HTML links), associating pages, and object relations (i.e., RDF links), associating data instances, create a semantic information network, as we named it.

The advantage of the current document-oriented web navigation is that each page contains conceptually related information, and enables the user to have an overview of the content, increasing content and context awareness and control [30]. However the problem is that, in each page the user is confronted with ample amounts of information. A purely a data-oriented approach has the advantage of enabling the user to iteratively filter the content in order to accesses information of interest. However applying a purely data-oriented approach to web navigation is problematic since: (1) in data-oriented approaches the navigation is highly sequential, consequently, long data chains constructed through RDF links can easily cause users to lose provenance and get lost, (2) embedded data available in different pages of a website does not necessarily need to be related or linked. In this context, purely data-oriented approaches are more suitable to expert users for specific purposes, like ontology navigation. We follow a hybrid approach merging document-oriented and data-oriented considerations. The hybrid approach gathers the benefits of both approaches: (1) by following semantic links a user can switch focus from one information cluster/sub-graph (i.e., webpage) to another at once, hence navigation experience is not highly sequential, while content and context awareness and control are maintained, and (3) by following data links within a webpage, the users can access information of interest through iteratively filtering the content rather than being confronted with abundant information.

The approach requires every instance, type, data type property, and object property (i.e., relationship) to be annotated with human consumable metadata for presentation purposes. The minimal requirement is assignment of human readable labels. A thumbnail and a short description are optional, but quite enhancing. Although additional domain knowledge can be utilized if available, the approach is domain independent and does not necessarily rely on the existence of such information. RDFa allows our approach to make use of the full potential of the RDF framework (i.e., types, item relationships, class relationships etc.) while the applicability of our approach with microformats is limited to typed instances (i.e., structured data) with a highly linear navigation experience. Any information source which includes embedded information in RDFa, microdata, microformats, or eRDF is supported by our proposed solution, as far as the basic requirement is satisfied (i.e., appropriate labeling). Several improvements for navigation have also been explored, for example, based on the number of available instances of a class or cardinalities of relationships. These are to prevent long request chains (see Section 5). A server-sided mechanism is preferred in order to isolate end-user devices from computational load of the extraction.

Two example scenarios, based on the semantic information network map depicted in Fig. 2, are introduced to demonstrate the applicability and benefits of the proposed approach. A cinema company provides recommendations for the movies of the season through its website. The site includes the pages 'Events' and 'Reviews'. Each movie is considered as an event in the 'Events' page. 'Reviews' page contains the reviews about the movies. Reviews are provided by registered reviewers, and users are subscribed to receive reviews. Additionally, address information for branches are given at the main page. Events, people, reviews, and addresses are particularly important entities, since they construct the essence of its content. Therefore, corresponding instances have been annotated in RDFa and widely known vocabularies such as iCal, vCard, and hReview are employed

Scenario-1: 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. However his mobile device's connection and screen properties are at a low level. Since the website is hosted by a server which is SWC enabled, the user simply sends a request through his mobile phone. His browser implicitly tells the server that it only requests annotated information. If the index page is requested, the server returns the list of semantic links and information available in the index page: 'Address' data type, 'Events Page', and 'Reviews Page'. The user follows the 'Reviews Page', and the server returns the list of available types: 'Reviews', and 'People'. The user selects the 'Reviews' type (see Subsection 5.1 for labeling and class names) and its instances are retrieved. The instances are presented by the titles of the movies and possibly associated with a small thumbnail and a short description if available. The user selects a review about a movie of interest and reads the review. He wants to see who wrote it to be sure that the quality of this information can be relied upon. The user follows the data link to the corresponding reviewer to access the reviewer instance for details. Then he navigates to the 'Events' page to see the schedule related information. This basic scenario is meant to

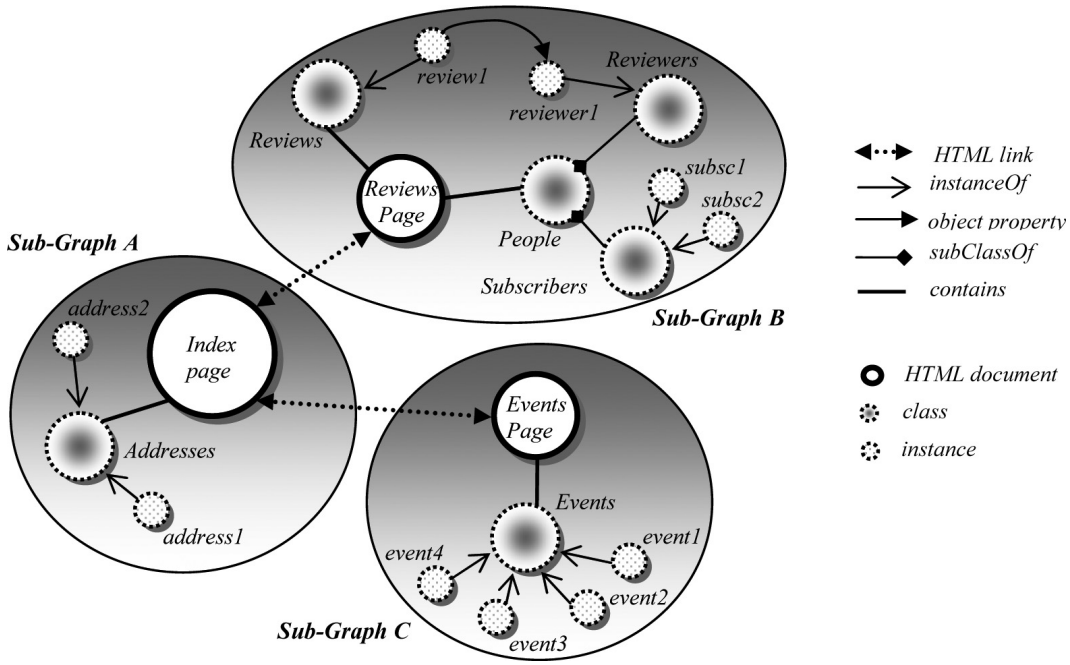

Fig. 2. Semantic information network map referring to semantic structure of a website.

address any kind of client (e.g., mobile, stationary etc.) having access to the Internet and any HTML browser with basic capabilities.

Scenario-2: Another user is visually handicapped, and uses a screen reader to navigate the Web. Information on websites is often abundant. Therefore she has to spend a lot of time to access the information of interest. She accesses a SWC enabled website. Only the semantic information constituting the essence of the site is downloaded. Furthermore, this data is not presented as a whole but presented in a hierarchical manner. On the one hand, the amount of content is reduced hence the total amount of text to be read. On the other hand, since the data is presented in a simple hierarchy, she can access a particular piece of information leaving non-essential and non-interested items unread.

In the following the advantages of the proposed approach are summarized with respect to related work presented in Section 3. (*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 [25] of the Web where valuable information is hidden in the 'information sea' and as part of presentational and structural elements. Users can simply access the desired information. (*3*) *Increased accessibility and ubiquity:* 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 the machine-readable form of the information will enable devices to process and use the web information. (*4*) *Higher network efficiency*: the devices do not need to retrieve all the (X)HTML content from the server, hence the amount of information travelling in the network decreases. (*5*) *Semi-centralized and a generic solution*: the fact that different embedded semantics technologies are available requires unifying the use of these technologies. In this paper, this diversity is advocated and unification is considered to happen through the sever-sided extraction mechanisms rather than by opting for a single technology. (*6*) *Low entry barriers:* Web publishers do not need high investment in development, content-authoring, and hardware, and since the requirements for the end-user devices are basic, the users do not need high-cost devices.

5. Methodology, design, and implementation

A sample HTML page with embedded linked data is used through this section in order to exemplify the proposed methodology. The example HTML page has been taken from the linked data tutorial published online at [16]. Exact details of the embedded data as

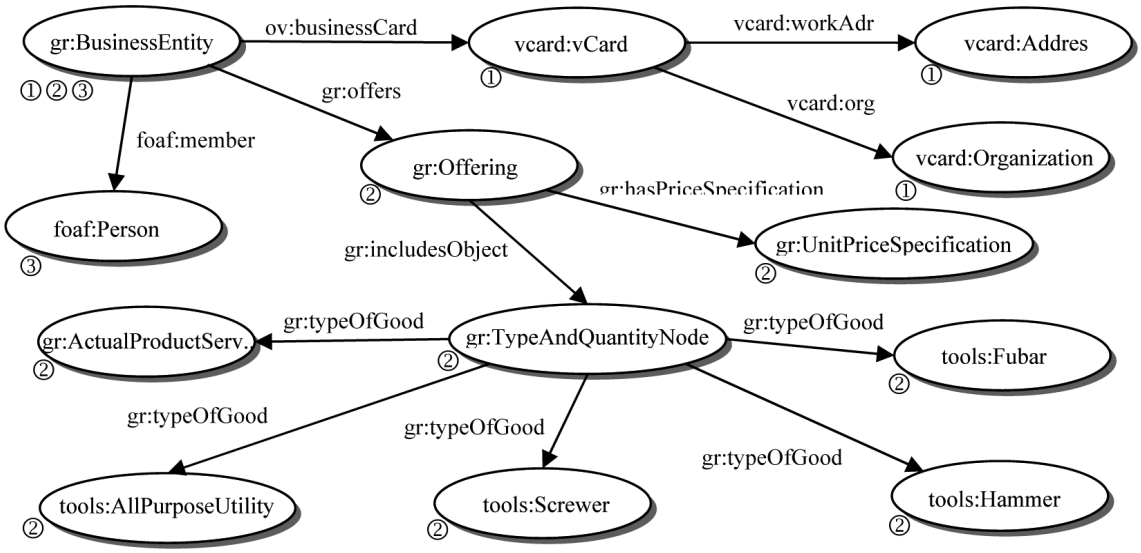

Fig. 3. Partially extracted ontology from the sample website; data type properties are omitted for the sake of brevity.

well as the original sample document can be found in [16]. The original HTML document is divided into sub-pages, namely Home (φ), Products (κ) and People (λ), in order to demonstrate semantic links.

The sample site provides basic information about a company, its products, and team members. Particular name spaces are used: 'gr' (GoodRelations) and 'tools' (a self-defined domain vocabulary) to present company and product related information, 'vcard' to present business card of the company, and 'foaf' to present information about people. The partial ontology of the website is depicted in Fig. 3. It is partial since the domain knowledge available is limited to the information revealed by the extracted instances.

5.1. Document preparation

The embedded semantic data needs to be adapted in three levels: (*1*) *metadata level* describes how instances, classes etc. needs to be annotated with human understandable textual and visual elements; (*2*) *domain knowledge level* describes how additional domain knowledge can enhance the navigation; and (*3*) *navigation level* describes how navigational hierarchy can be constructed and fine tuned.

5.1.1. Metadata level

Embedded information within the (X)HTML document needs to be accompanied with appropriate metadata in order to facilitate user consumption. Table 2 shows the list of required and optional metadata elements.

Embedded data is often presented by using original type, property, and relationship identifiers as it appears in a vocabulary or ontology. In other cases, it is assumed that the presentation environment has knowledge about the domain and provides appropriate visual elements for the humans. On the one hand, original names are not meant for regular end-users and mostly do not convey any meaning for them. On the other hand, assuming availability of domain pre-knowledge is not realistic within the context of generic semantic web browsers. Therefore embedded data needs to be accompanied with basic presentational metadata, primarily textual labels, while short descriptions and thumbnails are essentially enhancing.

Naming convention in a vocabulary or ontology is singular, since a class represents a real world concept. However, during navigation, the user perceives a class as a set of instances of a particular concept. Hence, the following conventions are suggested. We use plural label values for classes, (e.g., People for foaf:Person). While labeling data type and object type properties, rather than using the classical naming used in vocabularies or ontologies, i.e., 'verb' – '*range class name*' such as 'gr:hasPriceSpecification', target class name, or if not appropriate, another noun should be used without attaching a verb (e.g., 'Price' instead of 'hasPrice'). Plural label values should be used for Object type relations, if the cardinality is higher than 1, otherwise singular label values are appropriate (e.g., 'Price' for gr:hasPriceSpecification, and 'Members' for 'foaf:member').

Table 2
Metadata elements associated with the embedded data

Property	Description
Name space: RDF schema, Element: rdfs:label Optional: No	Provides human readable labels for: *types* (i.e., classes), *instances*, *data type properties*, and *object properties* (i.e., relations).
Name space: RDF schema, Element: rdfs:comment Optional: Yes	Provides human readable short descriptions for: *types, instances, data type properties,* and *object properties.*
Name space: Dublin core, Element: dc:description Optional: Yes	Provides thumbnail images for: *types*, *instances*, and *object properties.*

5.1.2. Domain knowledge level

Indeed, every application, maintains a conceptual model of its domain. Such conceptual models might be implicit (i.e., encoded in application code), or explicit but informal (i.e., not machine readable, e.g., documentations). Once a conceptual model is made explicit and formal, it can be used for knowledge share and reasoning. Embedded data within an (X)HTML document does not only serve domain instances, but also formalizes a part of domain knowledge through embedded data instances. Although the revealed knowledge is sufficient for our proposal, additional domain knowledge can be explicitly provided through embedding it along with data instances. In our context, subclass relationships (among others such as domain, range constructs etc.) are particularly beneficial. This is because domain knowledge can enhance the navigation experience, for instance, by classifying data instances with respect to a class hierarchy. Otherwise content navigation might be almost linear (object relations prevent it from being fully linear), and might overload the users with long lists of instances.

Considering the partial ontology presented in Fig. 3, a publisher could specify that 'tools:Hammer', 'tools:Screwer', and 'tools:Fubar' are subclasses of 'tools:AllPurposeUtility', and that 'tools:AllPurposeUtility' is a subclass of 'gr:AllProductServiceInstance'. Then, a major question arises concerning the amount of knowledge to be provided, for instance, after saying that 'instance A' is type of 'tools:Hammer', is it necessary to explicitly state that 'tools:Hammer' and 'tools:Screwer' are subclasses of 'tools:AllPurposeUtility', and 'instance A' is type of 'tools:AllPurposeUtility' and 'gr:AllProductServiceInstance'? In the context of the proposed methodology, a publisher can choose to provide subclass relationships, and leave 'subclass' and 'type' relationships to be inferred through ontology reasoning. That being possible, it compromise the performance since inference takes additional time. Nevertheless, in a typical web page, the amount of domain knowledge and instances are expected to be limited, hence the inference process should not cause long delays. Indeed, another possibility is to extract useful domain knowledge without requiring the publisher to provide it explicitly. This is possible by employing an ontology learning mechanism using clues revealed by the existing data instances. However, this might be costly and not sharply precise and accurate. The possible benefits and drawbacks will be discussed in Section 5.2. We presently do not utilize an ontology learning mechanism.

5.1.3. Navigation level

Our methodology intertwines document and data hierarchies. The former outlines the navigation while the latter determines access paths to content items. Data hierarchy is constructed through object relations and class-subclass relationships, while document hierarchy is constructed through HTML links. The main strategy, when a page is accessed, is to first list all available classes, which do not have a parent class, together with semantic links; this step constructs the front layer of the navigation hierarchy. Then the user is enabled to navigate through by selecting a class, subclass, instance, relation, range class and so on. For instance, consider that there are only two classes, 'class A' and 'class B', where ' class B' is subclass of 'class A' or an instance of 'class A' has an object relation with an instance of 'class B'. In the latter case, with the first request of the page, both classes will be presented to the user. On navigating through the 'class A', the user will also encounter the 'class B'. In the former case, only 'class A' will be presented, and 'class B' will only be accessible through 'class A'.

Neither a fully linear nor a fully hierarchical representation of the content is appropriate for the web navigation. Here, it should be possible to break the hierarchy and move a particular branch to the front, or to remove/hide a particular class from the front. Precisely: (1) a class might be hidden/removed from the front

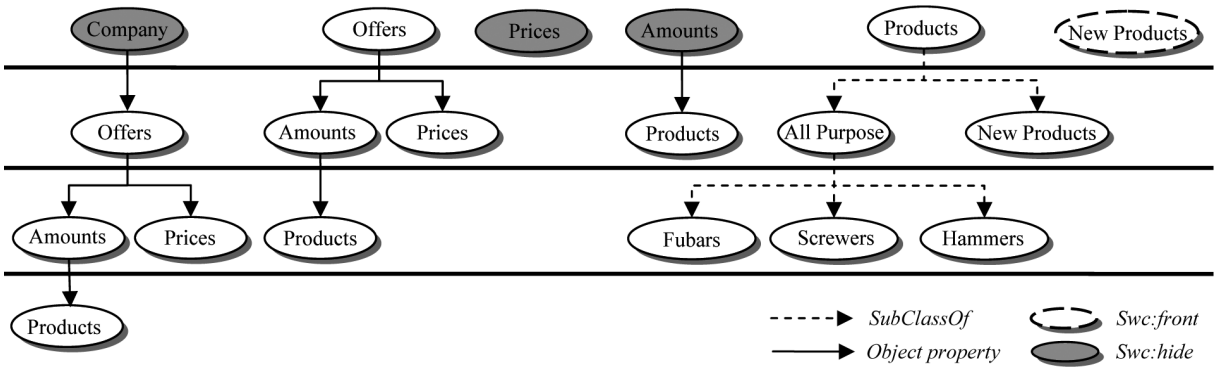

Fig. 4. Navigation hierarchy constructed for the Products page through object relations, class-subclass relations, 'swc:hide', and 'swc:front'.

layer, if the existence of this class strongly depends on another class (e.g., similar to weak entities in relational databases), (2) a class might be moved to front, if it contains instances which need to be immediately available through the front layer. An example is depicted in Fig. 4.

It shows classes available in the navigation hierarchy, at different levels, for the Product page of the sample site. This hierarchy is based on the following labeling: 'Company' for 'gr:BusinessEntity', 'Offers' for 'gr:Offering', 'Prices' for 'gr:UnitPriceSpecification', 'Amounts' for 'gr:TypeAndQuantityNode', 'All Purpose Utilities' for 'tools:AllPurposeUtility', 'Products' for 'tools:AllPurposeUtility', 'Fubars' for 'tools:Fubar', 'Hammer' for 'tools:Hammer', and 'Screwers' for 'tools:Screwer'. Each area between bold horizontal bars represents a navigation layer (the top one being the first layer). A class might appear in different layers due to object properties. In this example, 'Company', 'Offers', 'Prices', 'Products', and 'Amounts' appear in the front layer since they do not have any parent class. However, it is quite logical to remove 'Prices' and 'Amounts' from the front layer of the navigation hierarchy. This is because, in the sample, 'Amounts' are associated with 'Offers' through an object relation (see Fig. 3), hence every instance of 'Amounts' is associated with an instance of 'Offers'. In the context of the sample website, every 'Amounts' instance is only meaningful with an 'Offers' instance which makes direct access to an 'Amounts' instance useless and similarly for 'Prices'. Since the 'Company' class is the primary class of the data hierarchy, it might be appropriate to remove/hide it from navigation hierarchy since it might lead to a fully hierarchical experience. Consider that a new class 'New Products', being the direct subclass of 'Products', is added in order to advertise newly added products. The publisher may want to inform the customers as quickly as possible about new products by pushing this class to the front, where it can be directly accessed from the front layer as well as from its original position in the navigation hierarchy.

Two new data properties support this purpose, 'swc:hide' and 'swc:front', in our application's 'swc' name space. 'swc:hide' and 'swc:front' can only be applied to classes, and their value should be assigned to 'yes'. Another matter, concerning the navigation hierarchy, is interweaving HTML navigation and data navigation as demonstrated in Fig. 2. To support annotation of HTML links targeting sub-pages of the website containing embedded semantic data, a new class has been introduced, 'swc:SemanticLink'. Every link targeting another page containing embedded semantic data, should be annotated as an instance of 'swc:SemanticLink'.

5.2. Extraction, reasoning, and presentation

Extraction and reasoning processes are conducted at the server side while the presentation related process can either take place at the client (through JavaScript calls to the server) or at the server side (through HTML links). In any case, aforementioned processes exist under two layers, namely the extraction & reasoning layer and the presentation layer. Both layers are instance-based, that is a class or an object property etc. are only accepted to exist, if there is at least one instance associated with it.

5.2.1. Extraction & reasoning

At this layer, three core services are provided: (1) 'getClasses' is responsible of retrieval of classes in a particular level of navigation hierarchy, more specifically, (a) all non-hidden classes, having no parent class or those are pushed to the front layer, available at a given

HTML document, (b) classes that are direct subclasses of a particular class, (c) all non-hidden classes having a particular relationship with a particular instance, and (d) all classes that are direct subclasses of a particular class and having a particular relationship with a particular instance. This service requires ontology reasoning to be applied which we restrict to only subclass and type inference (i.e., excluding inference for range, domain, inverse of, sub-property etc.). Subclass and type inference are required to provide a hierarchical access to instances. For example, if one asserts that 'instance X' is type of 'class C', 'class C' is subclass of 'class B', and 'class B' is subclass of 'class A', then followings are inferred: 'class C' is subclass of 'class A', and 'instance X' is type of 'class B' and 'class A'. This implies 'class B' and 'class A' have to be visited before 'class C'. (2) 'getInstances' is responsible of retrieval of instances in a particular level of the navigation hierarchy, more specifically, (a) all the direct instances of a particular class, (b) all the direct instances of a particular class, and are in a particular relationship with a particular instance. (3) 'getInstance' is responsible of retrieval of a particular instance in a particular level of the navigation hierarchy, more specifically, all the properties of a particular instance. For consistency and performance matters, once embedded data is extracted, it is temporally stored during session life-time. Since only 'getClasses' service requires inferred data and others operate on non-inferred data pool, extracted information is stored both as is and with inferred data.

5.2.2. Presentation layer

The navigation starts with loading all classes having no parent class, and without imposing any specific relation with any instance. Once the results are received from the extraction & reasoning layer, after invoking the 'getClasses' service, the class related presentational metadata is listed as a menu item for each class. Specific actions are hooked to each item invoking 'getClasses' (retrieving direct subclasses of selected parent class) and 'getInstances' (retrieving direct instances of the selected class); this is repeated for any subclass selected. While listing instances, if the instance being listed is a 'SemanticLink' type instance, it is hooked with a specific action invoking the 'getClasses' service, and the instance URI becomes the new URL of the navigation provenance. If the instance being listed is not a semantic link type, the instance is hooked with an action invoking the 'getInstance' service in order to retrieve and present all the properties of the selected instance. If a property being presented is not a data type property but an object property, then an action invoking the 'getClasses', 'getIntances', or 'getInstance' service is hooked to it. Deciding which action to be hooked to an object property is subject to some heuristics for enhancing the user experience.

These heuristics are constructed by following an approach similar to a possible ontology learning mechanism based on embedded data. However it is fundamentally different in the sense that findings are mostly pseudo and specific to a single session. For instance, as mentioned in Subsection 5.1, rather than explicitly providing a class hierarchy, it can be constructed through analyzing the existing data. Nevertheless, an (X)HTML document with embedded data represents only a single portion of possible instances, and most of the domain knowledge deduced from this single set cannot be generalized due to the Open World Assumption (OWA) which is quite natural to follow in our context. Since the presentation approach followed by the proposal is situated on existing data rather than a reference domain ontology, some pseudo deductions might lead to useful heuristics which are demonstrated in Fig. 5. It is possible to shorten navigational chains by making use of object property characteristics such as cardinality and range. Normal behavior, after a particular relation is being selected, is to present instances in a class hierarchy (by default assuming that the cardinality is more than 1 with a multiple range). Consider the child-path P2 of P. To access price information of a particular offer, a user has to pass 5 navigation levels. However, since there is only one instance associated with this relation (pseudo cardinality 1), rather than listing the class hierarchy and then the instances (which is only one), the navigation can directly jump to the instance description. Such an approach shortens the navigation path 2 levels, and in cases where the real cardinality is exactly one, it removes a possible confusion by not following a singular relation label with a plural class label. Considering the child-path P1 of P, to access amount information of a particular offer, a user has to pass 5 navigation levels. However, since there is only one class having at least one instance associated with this relation (pseudo range 1), rather than listing the class hierarchy (which is only one), the navigation might directly jump to the list of instances. Such an approach shortens the navigational path 1 level. As demonstrated, pseudo cardinalities and ranges can lead to useful practices and omitting unnecessary navigation levels becomes possible due to the fact that textual description of a relation itself already reveals sufficient information about the coming presentational

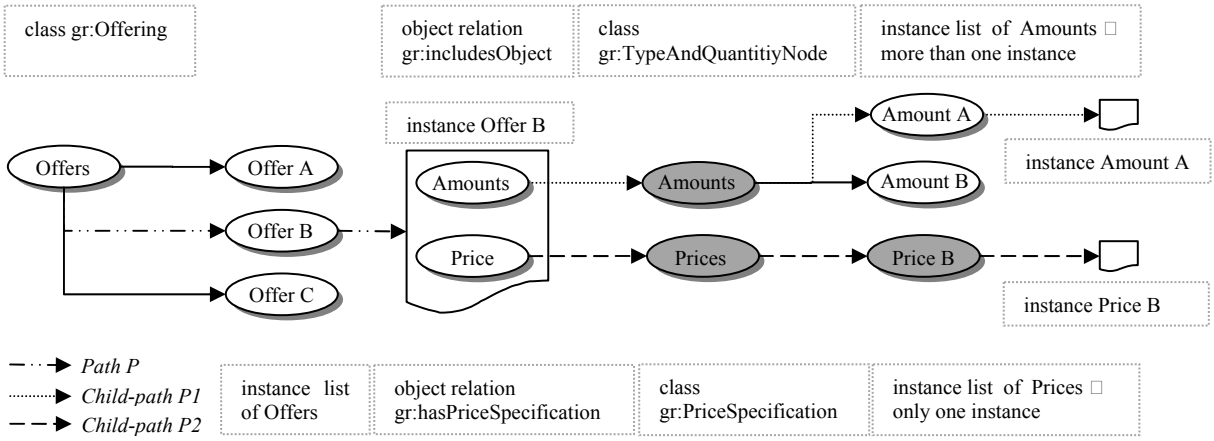

Fig. 5. Navigation paths to instance 'Amount A' and 'Price B' can be shortened by omitting intermediate steps in the class hierarchy.

level. Moving back to the ontology learning discussion, we do not attempt to learn a class hierarchy, because, differences between deduced pseudo class hierarchies and original ones may not be accepted by the users or confuse them.

5.3. Architecture

The proposed architecture (see Fig. 6) for SWC consists of three modules for a HTTP server: (*1*) *Mod Semantic* is responsible for extracting contextual information from the request header. It detects the device type or extracts an explicit semantic navigation request from the request header encoded with a specified parameter. This parameter can be further adjusted for directly accessing machine readable information (e.g., RDF). The module reads the requested (X)HTML document from the application pool and forwards it to 'Mod GRDDL', if semantic navigation request is active, otherwise to the client.

(*2*) *Mod GRDDL*: This module is responsible for extracting embedded semantic data from the (X)HTML. It stores the data, once extracted, to the session store temporarily, in RDF form, during the client's session life time. If inference over extracted data is demanded, it applies ontological reasoning and stores a new data-set separately. File identifiers are created through hashing source URL, session id of the client, and 'inferred'/'noninferred' parameters for quick and error free access.

(*3*) *Mod SWC:* This module is responsible for preparing and maintaining the state of the presentation. It detects the state of navigation (i.e., the active navigation level) and extracts the requested navigation level. It queries the session store, with SPARQL, for the corresponding URL and the parameters. On directly requesting machine readable information, it returns RDF data to Mod Semantic, otherwise it generates the requested presentation level in (X)HTML.

A listener is associated with Mod Semantic in order to direct incoming client requests to the SWC. Once Mod SWC delivers back the final output, Mod Semantic forwards it to the client. There might be other modules processing the content, for instance a script interpreter for dynamic content. In this case Mod Semantic needs to be placed in appropriate order within the module queue.

6. Evaluation

SWC has been implemented to prove its applicability. A first and simpler variant was restricted to microformat [29]. The current version supports RDFa, eRDF and microformats (as long as the requirements are satisfied) and is based on PHP. It behaves like a proxy. This design decision has been made due to the simplicity of the implementation and in order to provide a demonstrator for the community. The demonstrator is available through the following link http://www.ahmetsoylu.com/pubshare/icae2011/.

Two versions are available currently: The first version is based on PHP and JavaScript for providing enhanced user experience through a dynamic presentation layer. The extraction & reasoning layer is implemented as a server-sided component and the presentation layer is developed as a client-sided component. This prototype caches the visited navigation layers at the client. Hence, it does not require executing server calls more than once. However this first prototype is only suitable

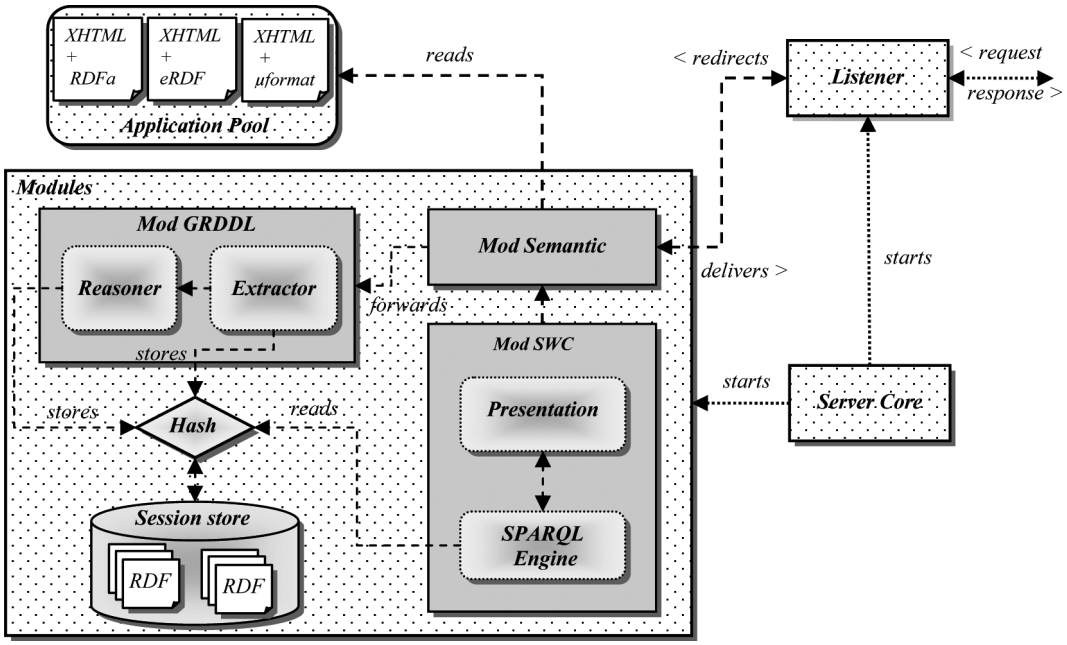

Fig. 6. SWC architecture consists of three modules for HTTP servers: Mod Semantic, Mod GRDDL, and Mod SWC.

for devices capable of executing JavaScript and having sufficient amount of browser cache. Although today most of the mobile devices have these capabilities, a second version has been developed as a fully server-sided component. An example navigation session of a user seeking a particular offer is depicted in Fig. 7.

6.1. Performance

Two performance tests for SWC have been conducted in order to validate the computational feasibility of the proposed approach. The tests measure the performance for data extraction, inference, and presentation processes. The first test is applied to sample (X)HTML documents containing increased amount of embedded linked data instances. The time spent for extraction, for inference, and the total amount of time spent for delivery of the first request and for the second request have been traced for each (X)HTML document. The results are shown in Table 3.

The number of triples extracted increase from 89 (roughly 10 instances) to 1324 (roughly 150 instances). Normally, total number of instances available in a single typical (X)HTML page is not expected to exceed 300 triples. Measurements done for higher numbers have been conducted to demonstrate the feasibility and the limits for larger data-sets.

Results show that the most expensive operation is the extraction taking roughly 5.27 seconds for 1324 triples. The total amount of time spent for delivering the requested content (including extraction, inference, and presentational process) is about 5.61 seconds where a minor amount of time is spent for presentational processing (which includes SPARQL querying), and a comparatively higher amount of time is spent for inference even though no triples are inferred. These results confirm that the approach is computationally feasible for even larger data sets because the extraction process is only executed for the first request of a page. Thereafter, the semantics embedded within the page is extracted and stored for subsequent requests during the session. The total amount of time spent for 2nd request (1324 triples) lasts only 0.173 seconds, which is a reasonable value compared to the 5 seconds required for the initial request. Although the time spent for inference is minor, defining an optional parameter for the document header indicating whether inference is required can eliminate it.

The second test is applied to the sample HTML documents containing increased amount of embedded linked data instances and additional domain knowledge causing an increasing number of new triples to be inferred. Regarding inference, only subclass inferences for T-box (stores terminological knowledge, e.g., classes, properties etc.) and type inheritance for A-box (stores assertional knowledge, e.g., instances) are enabled. The results are shown in Table 4.

Table 3
Performance results for SWC with no inferred triples (all time measurements in seconds)

# of extracted triples	# of inferred triples	# number of triples	t for extraction	t for inference	total t for 1st request	t for loading 2nd request	total t for 2nd request
89	0	89	0,282981157	0,026779891	0,318247082	0,017015219	0,026117086
154	0	154	0,479631901	0,040988922	0,529893875	0,023073912	0,032717943
284	0	284	0,865058184	0,068399906	0,944193125	0,036037922	0,046891928
414	0	414	1,258276939	0,095240116	1,366838932	0,046136141	0,059608221
544	0	544	1,662160158	0,123139143	1,801177025	0,059186935	0,074002981
674	0	674	2,096745014	0,154187918	2,269737959	0,074954987	0,092892171
804	0	804	2,820359945	0,180546999	3,022419931	0,08604002	0,107284784
934	0	934	3,498962879	0,212009907	3,736311913	0,098967075	0,123646021
1064	0	1064	4,067336082	0,240711927	4,338186979	0,111680031	0,140800953
1194	0	1194	4,684145927	0,273259878	4,992095947	0,142024041	0,175936937
1324	0	1324	5,270559788	0,302443027	5,613559961	0,135274172	0,173180103

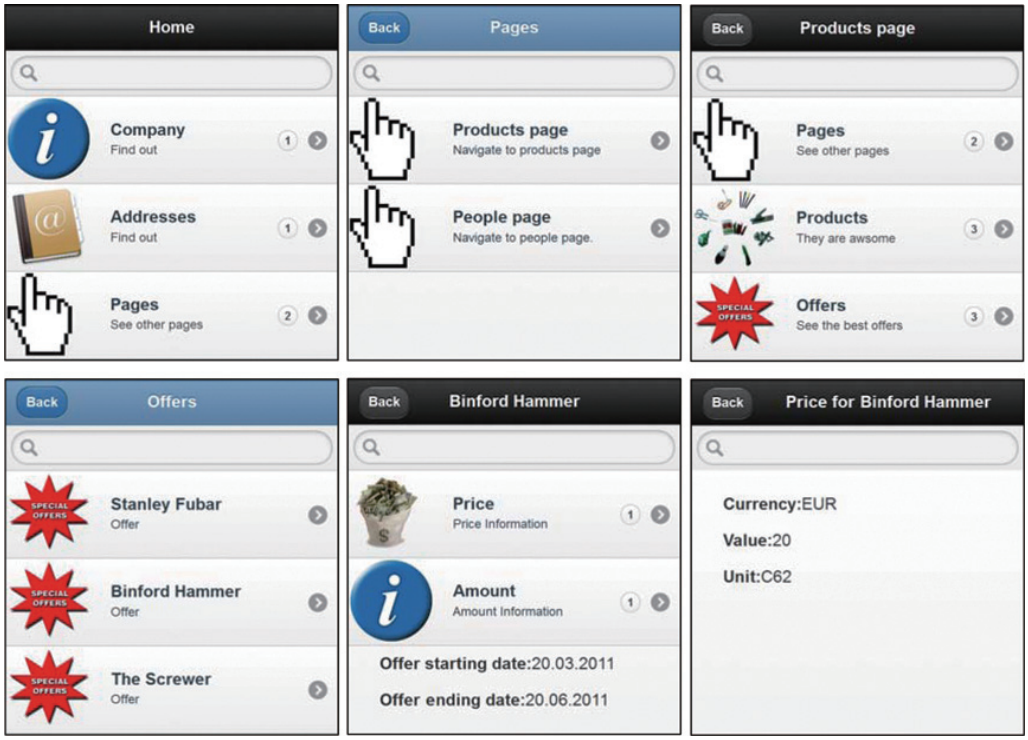

Fig. 7. An example navigation session of a user seeking a particular offer.

The time spent for inferring 600 triples from 1247 triples is around 0.56 seconds. The inference process, indeed, is known to be expensive. However, since T-box and A-box sizes are comparatively small, the time spent for inference is decent. T-box and A-box sizes, hence the number of inferences to be done, are not expected to be high for a typical (X)HTML page. Even for comparatively larger sizes, the results suggest that the proposal is feasible. If necessary, the number of extracted triples can be reduced by only using 'rdfs:label' and omitting 'rdfs:comment' and 'dc:description' elements anyway; this would reduce efforts by two triples per instance, class, and object property.

Overall, since the embedded data is divided into pages, and each HTML document contains a decent number of instances and domain knowledge, the proposed approach is found to be computationally feasible. The proposal is not tested with more data-sets and considerably bigger T-boxes, as the proposed approach does not aim at realizing a browser for knowledge bases with full-fledged inference support. The semantic reasoner used in the implementation and evaluation is not optimized for performance. Mature reasoners, opti-

Table 4
Performance results for SWC with inferred triples

# of extracted triples	# of inferred triples	# number of triples	t for extraction (second)	t for inference	total t for 1st request	t for loading 2nd request	total t for 2nd request
87	5	92	0,267782927	0,028562069	0,305315971	0,016274929	0,024868965
147	10	157	0,448071003	0,043763876	0,501824141	0,020592928	0,030339956
267	20	287	0,813581944	0,072698116	0,898980141	0,039797068	0,052239181
387	30	417	1,197718143	0,102658033	1,316694975	0,047405005	0,063531876
507	40	547	1,569974899	0,135476112	1,726269007	0,067197084	0,088369846
627	50	677	1,927372932	0,161001921	2,113218069	0,073454857	0,098347902
747	60	807	2,263870001	0,191908121	2,486411095	0,085950136	0,117304087
867	70	937	3,082850933	0,222520113	3,342474937	0,101141214	0,138206005
987	80	1067	3,769922018	0,273550034	4,090603113	0,116384983	0,162179947
1107	90	1197	4,216381073	0,290177107	4,561030149	0,130821943	0,183983088
1227	100	1327	4,744309902	0,338299036	5,146225929	0,142494917	0,205542088
1231	200	1431	4,816390038	0,373064995	5,278498888	0,152925968	0,241765022
1239	400	1639	4,829391003	0,453548908	5,422011852	0,178221941	0,324625969
1247	600	1847	4,820935011	0,561543941	5,575149775	0,206342936	0,415158987

mized for best performance, are expected to provide even better performance.

6.2. Network efficiency

To evaluate network efficiency, precision and number of requests are used as criteria. In our context, precision (P) [23] is the fraction of the size of retrieved data that are relevant to the user's information need (i.e., target instance), and number of requests refers to the total number of HTTP calls required to access the target instance. Precision is calculated by

$$P = \frac{t}{\sum_{i=1}^{n} p_i}$$

where n denotes number of network requests, p_i denotes the size of the returned data for a request, and t denotes size of the target instance. More specifically, for normal navigation, p_i refers to the size of an (X)HTML page required to be visited to access target information, where for SWC, it refers to the size of a presentational layer required to be visited to access target information. An example trace has been shown in Table 5 for four different target instances.

Although the sample site is quite small and does not include many irrelevant content elements the precision reaches to 72% where it is only 16% for normal web navigation. The lowest precision is measured with SWC is 6% where it is 0.6% for normal web navigation. The lowest precision, although it is still 10 times higher than the normal navigation, results from the size of the target instance. The example instance, a product, only contains its name (one word, i.e., a few Bytes). A typical web page is expected to have a bigger size, and a higher navigational depth. A typical embedded data instance is expected to have 6–7 properties resulting in 1–2 KB of target data size. In this respect, for a full-fledged website, precision is expected to be much lower for normal navigation and much higher for SWC. Regarding network calls, naturally, SWC requires more network calls than normal navigation, since it iteratively proceeds to the target instance. However the increase in amount of network calls seems admissible since the amount of information downloaded in each call is considerably small.

The example provided in Table 5, being extremely optimistic for normal navigation, implies higher network efficiency. The significant reduction in transferred data size clearly favors our solution approach.

6.3. Usability

A preliminary usability evaluation has been conducted to see (1) whether our semantic approach can create a satisfactory navigation experience comparable/superior to normal navigation, (2) to find directions for developing more heuristics, and (3) to detect any major usability problems. The usability analysis is targeted to find problems inherent to the methodology itself, regardless of problems originating from the target websites (i.e., content organization).

As suggested in [33], 5–6 users are normally sufficient to cover major usability problems while 15–16 users provide the highest benefit. This part of the evaluation is iterative and expected to provide a basis for the future work. Hence we opted to conduct a preliminary study first with 6 test users. The profiles of the test-users are given in Table 6.

A think-aloud test was conducted. The test-users were asked to conduct four tasks through the sample

Table 5
Precision and number of required requests are traced for a set of target instances

target page	target instance	target instance size (KB)	retrieved data size with SWC	precision with SWC	# of requests	retrieved data size with normal navigation	precision with normal navigation	# of requests with normal navigation
home	organization	1.81	2.49	0.72	3	11.13	0.160	1
home	address	0.79	1.48	0.53	3	11.13	0.070	1
people	a member	1.26	3.05	0.41	5	18.68	0.060	2
products	a product	0.16	2.54	0.06	7	24.60	0.006	2

Table 6
Profiles of the test users

User	#years using Internet	frequency of Internet use	# years using mobile Internet	level of expertise	frequency of mobile Internet use	age group
1	12	daily	0	Regular user	never	25–30
2	12	daily	1	Regular user	often	20–25
3	14	daily	3	Developer	occasionally	25–30
4	15	daily	2	Computer Sci.	sometimes	30–35
5	15	daily	0	Regular user	never	25–30
6	5	daily	1	Regular user	often	30–35

website with SWC: (1) find a particular person, (2) find company's contact details, (3) find a particular product, and (4) find a particular price information.

Regarding the navigation experience, three users managed to complete given tasks without any observable significant confusion in their first try. 'User 3', 'User 6' and 'User 5' could not complete the first task in their first try (due to content organization) but completed all other tasks in their first try. Users stated that they were quite satisfied with their navigation experience and they did not have any confusion or uncertainty, but had some critiques on the organization of the content. At the end of the tasks, the original pages have been shown to the users, and asked what differences they see. Users, mainly, delivered their experience with normal web navigation, stressed difficulty of finding information, and commented that it is easier to access information with our approach. They listed several websites, which they may want to access with such a mechanism.

Regarding possible suggestions for new heuristics, it has been observed that the users were quite reactant to unexpected situations, for instance, when they did see a class (e.g., menu item) containing only one instance or subclass, they immediately commented on unnecessary navigation levels. Even though the number of instances or subclasses might change (e.g., a new product or product type appears), users were apparently instance-oriented. This might require us to apply a new heuristic, by omitting classes in navigation hierarchy with only one subclass or one instance. For task 3, 'User 2', 'User 4' and 'User 5' tried to access price information trough products and complained when they could not access it. This might require us to employ inverse properties and enable inference support for this construct.

Regarding usability problems, users reported two minor issues. 'User 4', 'User 5' and 'User 6' complained that pages are grouped under a category, and commented that they should be directly accessible. 'User 2' commented that pages should be accessible at every level of navigation. This might require us enabling semantic link instances to be directly visible without being classified under any parent class. However class based access should be maintained and particularly for websites with high number of links. Whenever users are asked to find an instance at another page, it is observed that users quickly moved back to pages option, and navigated to the correct page. Users were asked whether they know where they are, and successfully replied with the active page. 'User 2' commented that he knows being at 'Products page', but it might be hard to remember if navigation becomes deeper requiring active page information to be maintained. Users also commented that it would be good if they could search within the content. Providing a search mechanism, exploiting the structure and semantics, is definitely crucial.

Regarding the publisher side of the usability, SWC does not remove the usability problems inherent to content organization, and usefulness of the embedded data (i.e., what to structure and annotate). Publishers need to employ some measures to test usability of the content organization. Two notable measures, that we recommend, are observed: precision and perceived recall. A precision function is defined in Subsection 6.2 for network efficiency. Indeed it can also be considered as

a measure for user cognitive load; since the rate of the presented information has an impact on the cognitive load [26]. However, the precision function defined in Subsection 6.2 is the expected precision assuming that the users follow the correct paths and do not get lost. Observed precision accounts the unexpected navigational levels that a user visited. Expected and observed precision can be combined to an efficiency measure to identify usability problems regarding the content organization. The efficiency measure is defined by

$$E = \frac{\frac{t}{\sum_{i=1}^{n} p_i}}{\frac{t}{\sum_{j=1}^{m} p_j}}$$

where n denotes number of expected requests, m denotes number of observed requests, p_i denotes the size of returned data for an expected request, p_j denotes the size of the returned data for an observed request, and t denotes the size of the target instance.

Regarding perceived recall (R) [23], it is the fraction of relevant data size that is retrieved. Recall can be calculated by asking users which information they deem relevant, and comparing it with scope of embedded instances. Perceived recall can allow publishers to fine-tune their decision on what is relevant and essential for the users visiting their website.

7. Conclusion

In this paper, a methodology has been proposed for enabling end-users and devices in UbiComp environments to navigate websites along their semantic structure and domain knowledge. Proposed methodology follows a hybrid approach of document-oriented and data-oriented considerations. Embedded data specification, extraction, and presentation mechanisms have been defined. Several heuristics have been introduced to make use of domain knowledge, with ontology reasoning, for generating user-friendly navigation experiences. A prototype, named SWC, and its architecture have been described. Several metrics have been applied or introduced for evaluation of the proposed approach and for an efficient semantic content-organization. The approach has been evaluated along a concrete scenario and with respect to precision, performance, network traffic, and usability. The evaluation results suggest that the proposed approach decreases network traffic as well as the amount of information presented to the users without requiring significantly more processing time, and that it allows creating a satisfactory navigation experience.

The future work, firstly, involves investigation of new heuristics for enhancing navigation experience. Secondly, from the interaction point of view, annotating interactional elements (i.e., HTML forms) will lead us to full realization of our approach; techniques for reformation of annotated interactional elements will be investigated.

Publishers need to be supported with appropriate tools in order to automate the annotation process. Such tools might employ database schemas [15,35], domain ontologies (possibly interweaved with database schema [32]), or the website itself by detecting data items [13], even relations, within the (X)HTML documents. Finally, we emphasize that the proposed approach can be used for various devices; it has been evaluated for a mobile scenario for simplicity. However, in UbiComp environments, support for different modalities (e.g., haptic interfaces etc.) are important, automated approaches are required for adapting the navigation experience with respect to modality of the end-user interface (e.g. [22], for IPTV).

Acknowledgements

This research is conducted within the project 'Harnessing collective intelligence in order to make e-learning environments adaptive' (IOF KP/07/006). Partially, it is also funded by the EC's IST-FP7 under grant agreement no 231396 (ROLE project).

References

[1] B. Adida, hGRDDL: Bridging micorformats and RDFa, *Journal of Web Semantics* **6**(1) (2008), 61–69.
[2] J. Allsopp, *Microformats: Empowering Your Markup for Web 2.0*, FriendsofED, Berkeley, 2007.
[3] C.R. Anderson, P. Domingos and D.S. Weld, Personalizing Web Sites for Mobile Users, in: *WWW 2001*, ACM, 2001, pp. 565–575.
[4] S. Auer, R. Doehring and S. Dietzold, LESS – Template-Based Syndication and Presentation of Linked Data, in: *LNCS: The Semantic Web: Research and Applications* (Vol. 6089), Springer, 2010, pp. 211–224.
[5] D. Ayers, The Shortest Path to the Future Web, *Internet Computing* **10**(6) (2006), 76–79.
[6] T. Berners-Lee, J. Hendler and O. Lassila, The semantic web, *Scientific American* **284**(5) (2001), 34–43.
[7] C. Bizer, The Emerging Web of Linked Data, *IEEE Intelligent Systems* **24**(5) (2009), 87–92.

[8] C. Bizer, T. Heath and T. Berners-Lee, Linked Data: The Story So Far, *International Journal on Semantic Web and Information Systems* **5**(3) (2009), 1–22.
[9] O. Buyukkokten, H. Garcia-Molina and A. Paepcke, Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices, in: *WWW 2001*, ACM, 2001, pp. 652–662.
[10] Y. Ding, Y. Sun, B. Chen et al., Semantic Web Portal: A Platform for Better Browsing and Visualizing Semantic Data, in: *LNCS: Active Media Technology* (Vol. 6335), Springer, 2010, pp. 448–460.
[11] F. Fallucchi, M.T. Pazienza, N. Scarpato, A. Stellato, L. Fusco and V. Guidetti, Semantic Bookmarking and Search in the Earth Observation Domain, in: *LNCS: Knowledge-Based Intelligent Information and Engineering Systems* (Vol. 5179), Springer, 2008, pp. 260–268.
[12] M. Forte, W.L. de Souza and A.F. do Prado, Using Ontologies and Web Services for Content Adaptation in Ubiquitous Computing, *Journal of Systems and Software* **81**(3) (2008), 368–381.
[13] X. Gao, L.P.B. Vuong and M. Zhang, Detecting Data Records in Semi-Structured Web Sites Based on Text Token Clustering, *Integrated Computer-Aided Engineering* **15**(4) (2008), 297–311.
[14] D. Griesi, M.T. Pazienza and A. Stellato, Semantic Turkey – a Semantic Bookmarking tool (System Description), in: *LNCS: The Semantic Web: Research and Applications* (Vol. 4519), Springer, 2007, pp. 779–788.
[15] B. Harrington, R. Brazile and K. Swigger, A Practical Method for Browsing a Relational Database using a Standard Search Engine, *Integrated Computer-Aided Engineering* **16**(3) (2009), 211–223.
[16] M. Hausenblas and R. Cyganiak, Publishing and consuming Linked Data embedded in HTML, http://www.w3.org/2001/sw/interest/ldh/, Retr. Oct. 2011.
[17] I. Herman, W3C Semantic Web Activity, W3C, http://www.w3.org/2001/sw, Retr. Oct. 2011.
[18] I. Hickson, HTML Microdata, http://dev.w3.org/html5/md-LC/, Retr. Oct. 2011.
[19] K.S. Jones, Automatic summarising: The state of the art, *Information Processing & Management* **43**(6) (2007), 1449–1481.
[20] R. Khare, Microformats: the next (small) thing on the semantic Web? *Internet Computing* **10** (2006), 68–75.
[21] R. Khare and T. Çelik, Microformats: A pragmatic path to the Semantic Web, in: *WWW 2006*, ACM, 2006, pp. 865–866.
[22] M.H. Lee, The design of a heuristic algorithm for IPTV web page navigation by using remote controller, *IEEE Transactions on Consumer Electronics* **56**(3) (2010), 1175–1781.
[23] C.D. Manning, P. Raghavan and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.
[24] F. Mödritscher, Semantic Lifecycles: Modelling, Application, Authoring, Mining, and Evaluation of Meaningful Data, *International Journal of Knowledge and Web Intelligence* **1**(1/2) (2009), 110–124.
[25] M.S. Pera and Y.K. Ng, Utilizing Phrase-Similarity Measures for Detecting and Clustering Informative RSS News Articles, *Integrated Computer-Aided Engineering* **15**(4) (2008), 331–350.
[26] M. Sicilia and S. Ruiz, The effects of the amount of information on cognitive responses in online purchasing tasks, *Electronic Commerce Research and Applications* **9**(2) (2009), 183–191.
[27] A. Soylu, P. De Causmaecker and P. Desmet, Context and Adaptivity in Pervasive Computing Environments: Links with Software Engineering and Ontological Engineering, *Journal of Software* **4**(9) (2009), 992–1013.
[28] A. Soylu, P. De Causmaecker and F. Wild, Ubiquitous Web for Ubiquitous Computing Environments: The Role of Embedded Semantics, *Journal of Mobile Multimedia* **6**(1) (2010), 26–48.
[29] A. Soylu, F. Mödritscher and P. De Causmaecker, Multi-facade and Ubiquitous Web Navigation and Access through Embedded Semantics, in: *LNCS: Future Generation Information Technology* (Vol. 6485), Springer, 2010, pp. 272–289.
[30] S. Spiekermann, *User Control in Ubiquitous Computing: Design Alternatives and User Acceptance*, Shaker Verlag, Aachen, 2008.
[31] G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru and S. Decker, Sig.ma: Live views on the Web of Data, *Journal of Web Semantics* **8**(4), 355–364.
[32] M.C. Valiente, A systematic review of research on integration of ontologies with the model-driven approach, *International Journal of Metadata, Semantics and Ontologies* **5**(2) (2010), 134–150.
[33] R.A. Virzi, Refining the Test Phase of Usability Evaluation: How Many Subjects Is Enough? *Human Factors* **34**(4) (1992), 457–468.
[34] M. Weiser, The computer for the 21st century, *Scientific American* **265**(3) (1991), 66–75.
[35] F. Zhang, Z.M. Ma and L. Yan, Construction of Ontology from Object-oriented Database Model, *Integrated Computer-Aided Engineering* **18**(4) (2011), 327–347.