Assisting Mobile Web Users: Client-Side Injection of Context-Sensitive Cues into Websites

Sven Casteleyn
Vrije Universiteit Brussel
Pleinlaan 2, 1000 Brussel, Belgium
+32 2 629 3754
Sven.Casteleyn@vub.ac.be

William Van Woensel
Vrije Universiteit Brussel
Pleinlaan 2, 1000 Brussel, Belgium
+32 2 629 3754
William.Van.Woensel@vub.ac.be

Olga De Troyer
Vrije Universiteit Brussel
Pleinlaan 2, 1000 Brussel, Belgium
+32 2 629 3504
Olga.Detroyer@vub.ac.be

ABSTRACT

In a mobile setting, the user often browses the Web to consult information related to his current context and environment: e.g., reviews of nearby restaurants, or tourist information on visited monuments. On the other hand, the limitations of mobile devices (e.g., limited screen) and the peculiarities of mobile Web usage (e.g., walking around, driving a car) make it cumbersome to extensively browse a Web page for such useful information. In this paper, we present a client-side approach that aims to assist the mobile user in his browsing session, by correlating the Web page's content with the mobile user's context, and subsequently emphasizing and enriching relevant content with so-called context-sensitive cues. To achieve this, we utilize the SCOUT framework for mobile applications to model and access the user's context, and RDFa annotations present on existing Web pages to identify Web page elements suitable to enrich with context-sensitive cues. The cues themselves are injected using existing adaptation techniques, borrowed from the field of Adaptive Hypermedia.

Categories and Subject Descriptors

D.2.0 [**Software Engineering**]: General; H.1.1 [**Information Systems**]: Models and Principles – *Systems and Information Theory*; H.4 [**Information Systems**]: Information Systems Applications; H.5.4 [**Information Systems**]: Information Interfaces and Presentation – *Hypertext/Hypermedia*

General Terms

Theory, Design

Keywords

Mobile Application, Context-awareness, Context-sensitive Adaptation, Client-side, Semantic Web

1. INTRODUCTION

In recent years, we have seen a drastic evolution of the Mobile Web. Several enabling technologies are reaching maturity, leading to an increased market segment of Mobile Web users[1] who no longer solely need to rely on specialized, down-scaled versions of Web applications. In particular, new generation mobile devices, such as Apple's multi-touch iPhone or Research in Motion's (RIM) Blackberry, provide more advanced and user-friendly input capabilities, and sport Web browsers capable of displaying most Web applications without the need for specialized mobile versions. Furthermore, wireless networks are becoming increasingly available, and up to par data transmission speeds (e.g., 3G networks, available in most developed countries and allowing up to 14 Mbit/s download speeds) are now gradually becoming accessible to the general public. Additionally, mobile devices increasingly possess positioning and sensing capabilities (e.g., GPS; RFID and NFC readers[2]; Quick Response (QR) codes[3], supported on any Web enabled camera phone), which provide possibilities to extensively map the user's environment.

On the other hand, Mobile Web usage is significantly different compared to browsing the Web on a desktop. Mobile users often have particular browsing purposes (e.g., compare shop prices, finding reviews of nearby restaurants), and cannot spend the same amount of time browsing and searching for information (e.g., they are walking around, driving a car). Furthermore, despite hard- and software advancements, mobile users still suffer from the limitations of their mobile device as compared to their desktop counterparts (e.g., smaller screen size, reduced input possibilities), making it cumbersome to view all information offered on a regular Web page.

In this paper, we present a client-side approach, called COIN (COntext INjection), which makes visited Websites context-sensitive on-the-fly by using so-called context-sensitive cues, in order to assist the mobile user in his browsing activity. Context-sensitive cues highlight content on a Web page that is related to the context of the user (e.g., monuments he has passed, buildings currently nearby), and/or enrich such content with additional information obtained from the user's context (e.g., description, time spent nearby), to achieve a certain user-specific goal. For example, on a tourist Website, a context-sensitive cue could consist of highlighting places visited within the last twenty-four hours, and indicating the time at which the user encountered this place, in order to emphasize information relevant to the user's tour through the city. Such cues may also be personalized (e.g., only indicating vegetarian restaurants if the user is a vegetarian), which further increases their benefit.

[1] According to a study of NetMarketshare, mobile browser market share doubled in the last eight months; see http://marketshare.hitslink.com/report.aspx?qprid=61

[2] E.g., Nokia 5140 supports an RFID Reader Shell, while Nokia's 6131 and 6212 models contain a built-in NFC reader.

[3] http://www.denso-wave.com/qrcode/qrstandard-e.html

In a nutshell, our approach consists of the following three steps: 1) identify candidate Web page elements to enhance, by extracting the semantics of the requested Web page's content using semantic annotations (e.g., RDFa[4], microformats[5]) embedded in the content, 2) match the extracted metadata with information available on the user and his (current and past) context, and 3) inject context-sensitive cues into the original Web page, using adaptation techniques borrowed from the field of Adaptive Hypermedia.

Our approach heavily relies on Semantic Web technologies. First, we use the SCOUT framework for mobile applications [1], which uses RDF(S) to model the user's environment and is based on modern mobile phone's positioning and sensing possibilities. Furthermore SPARQL is used to query this environment model. Second, we exploit semantic annotations embedded in Websites, to identify candidate Web page elements to enhance with context-sensitive cues, and to match their associated metadata with (metadata of) the user's context.

The strength of this client-side approach is that it doesn't require any pre-engineering from the Web developer while creating the Website: it works on any Website available on the Web as long as it contains machine-readable semantic markup (our approach currently supports RDFa). We provide two realizations of our approach: a client–side solution, where the user needs to install our mobile browser plugin, and also a server-side solution, where the Web developer simply has to add a reference to our Javascript code in his Website.

The remainder of this paper is structured as follows. Section 2 shortly recaps the SCOUT framework for mobile application development, as this approach contains a mobile application component built on top of this framework. Section 3 gives a general overview of our approach to enhance existing Websites with context-sensitive cues, and introduces a scenario which is used throughout the paper. Section 4 explains how to identify candidate page elements on a Web page using semantic annotations, in order to add context-sensitive cues. Section 5 explains how to match candidate Web page elements with information on the user's context. Section 6 discusses how to inject the context-sensitive cues. Section 7 discusses the implementation of our approach, and section 8 related work. Finally, section 9 provides conclusions and future work.

2. SCOUT IN A NUTSHELL

The SCOUT framework aims to support application developers in building environment- and context-aware mobile applications. These applications can rely on SCOUT's ability to collect and interpret certain sensory data, and to build an integrated, conceptual model of the user's current and past physical environment. First, we recap the SCOUT framework in this section, before elaborating the COIN approach and its mobile application component built on top of this framework.

The SCOUT framework consists of a layered architecture: each layer captures one particular design concern, and provides abstractions and functionalities to the upper layer(s). By pursuing a rigorous separation of concerns, we assure independence between layers and from underlying technologies. We shortly explain each layer (bottom-up) below; a full overview of SCOUT explaining each layer in detail can be found in [1].

The **Detection Layer** is responsible for detecting identifiable physical entities (e.g., shops, monuments, buildings, other mobile users) in the vicinity of the user. The layer contains a number of components, each encapsulating a certain detection technique (e.g., RFID/NFC, Bluetooth). We only assume a detected entity is able to communicate a reference to an online resource describing it, i.e., a URL. For the purpose of this paper, we assume these online resources are RDF(S) files, describing the particular entity, e.g., for a detected entity "the Atomium", a well-known monument in Brussels, this RDF(S) source could contain the name "Atomium" using the rdfs:label property, a description using dcmi:description[6], additional info using rdfs:seeAlso, etc.). One popular and cheap example of a detection technology is Quick Response (QR) codes: 2D codes (encrypting a URL) that can be printed using a regular printer, and subsequently scanned and decoded to its original URL by any camera-enabled mobile device. Another example is an online directory, which provides location-specific resource URL's based on the user's current GPS coordinates. As the different detection technique components implement a uniform interface, the framework allows transparently switching between different detection techniques, or using several in parallel.

The **Location Management Layer** receives raw detection information from the Detection Layer, and conceptualizes it by creating positional relationships: when an entity is determined to be nearby, a positional relation is created; when the entity is no longer nearby, the positional relation is invalidated. Determining proximity (i.e., nearness and remoteness) is done using proximity strategies, which may differ depending on the available detection data and the specific detection technique used. For example, a straightforward proximity strategy for short-range, direct detection techniques (e.g., RFID, NFC, Bluetooth) consists of considering entities nearby whenever they are detected, and no longer nearby when they move out of range. By allowing applications to deal with positional relations in this conceptual way, we allow them to abstract from particular sensing technologies and the details of interpreting raw positional information.

The **Environment Layer** provides applications with an integrated, conceptual view of the user's current (and past) physical environment. For this purpose, it consists of two sub-models: the (positional) Relation Model and the User Model. For each detected entity, a reference to its online source (i.e., a URL), and the creation and invalidation time of the corresponding positional relation is stored in the (positional) Relation Model. Next, information about the user, his characteristics, needs and preferences, is stored in the User Model. The Environment Model combines the information from the Relation Model, the online RDF(S) source associated with nearby (or no longer nearby) detected entities, and the information about the user himself, in order to provide mobile application developers with an integrated view on the user's current and past environment[7].

[4] http://www.w3.org/TR/xhtml-rdfa-primer/

[5] http://microformats.org/

[6] Dublin Core Metadata Initiative, a well known metadata standard, see http://dublincore.org/

[7] SCOUT combines online source and query analysis, together with caching mechanisms, in order to optimize the amount of

In the Environment layer, Semantic Web technologies are heavily exploited: RDF(S) is used to store the aforementioned models, while existing RDF(S) / OWL vocabularies and ontologies are re-used to represent online sources and the user's metadata contained in the User Model (e.g, CC/PP[8] to represent preferences and mobile device capabilities, FOAF[9] for representing certain properties of the user, DCMI[6] to describe documents and items).

The Environment Layer also provides mobile application developers with some basic services that provide access to these models: pull-based data retrieval, where arbitrary SPARQL queries are issued over the different models using the Query Service, and push-based data retrieval, where a Notification Service monitors changes in the environment and alerts registered applications of specific changes.

The final layer, the **Application Layer**, contains the mobile applications that are built on top of SCOUT. One such application is the COntext INjection (COIN) mobile application, part of the COIN approach which we elaborate in detail in the next sections.

3. INJECTING CONTEXT-SENSITIVE CUES: GENERAL OVERVIEW

We are now ready to show how the injection of context-sensitive cues is realized. Let us first elaborate a scenario that illustrates the usefulness of context-sensitive cues, which we use throughout the following sections to exemplify our approach. A mobile user is visiting Brussels, and travelling around to visit various points of interest. While on the road, he uses his mobile device to browse the city's tourist Website in order to find out more information about the places, monuments, museums, etc. he is visiting. This Website offers a wealth of information on Brussels' famous places and attractions, yet for the mobile user, it is cumbersome to locate the information relevant for him at a particular moment. For instance: which attractions are currently nearby or what has he already seen? All the desired information is present on the Website, yet it is buried in the wealth of information available. To aid the mobile user, we inject context-sensitive cues into the Web page he is visiting: i.e., cues which highlight places and/or attractions he is currently nearby (or has been nearby in the course of the day), while also conveying some information about the encountered places/attractions, e.g., the time at which he encountered them, how long he was nearby, and a link to their own Website (if available). Continuing our scenario, towards dinnertime, the mobile user would like to locate a restaurant to have dinner. There are plenty of restaurant listings online, yet a similar problem arises: are there any restaurants nearby his current location, or his hotel?

information to download and store. This however is outside the scope of this paper.

[8] http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430/

[9] http://www.foaf-project.org

Do they cater for his particular tastes (e.g., Italian cuisine)? Here, cues are injected that indicate suitable restaurants, using a recognizable icon, while also conveying why the restaurant was highlighted (e.g., because they serve Italian cuisine).

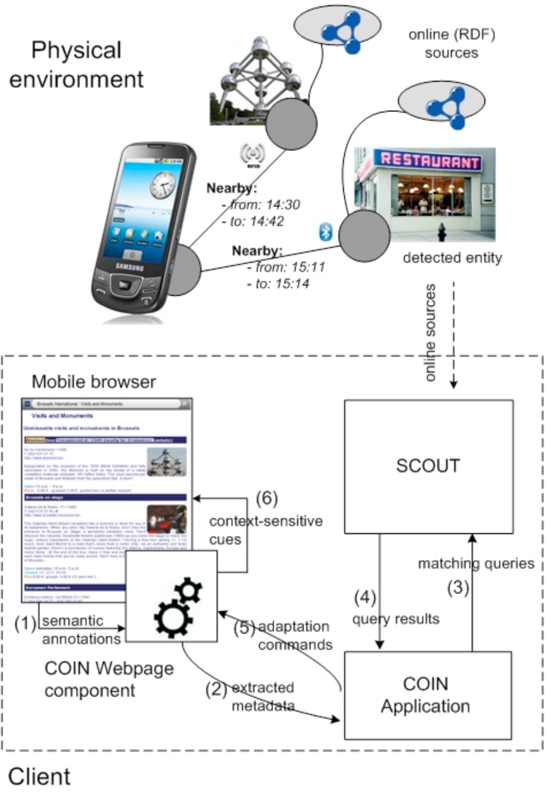


Fig. 1. COIN Architecture

Figure 1 shows a general overview of our approach, without going into technical details. We summarize the injection process below, while the subsequent sections will explain the process in detail.

To be able to inject context-sensitive cues in a visited Web page, the first required step is to analyze the page after it has been loaded, in order to identify the page elements that potentially qualify for context-sensitive cue injection. This is done by a first component of COIN: a Web page component that parses the semantic annotations on the Web page, and extracts the corresponding metadata (1). These annotations provide explicit meaning for specific Web page elements, and can thus be employed to correlate them to the user's current and past context. This metadata, along with their corresponding page elements, is communicated to the COIN mobile application (2), which runs on top of the SCOUT framework. COIN subsequently invokes queries on SCOUT's Environment Model (3), in order to match the semantic descriptions from the Web page with entities available in the user's Environment Model. Additionally, these queries may include conditions over the User Model, in order to further personalize results. The results of these queries (if any) indicate which context-sensitive information can be injected into the Web page, and are communicated back to the COIN application (4). Based on these matching results, the COIN application sends adaptation commands to the COIN Web page component (5), which injects the corresponding context-sensitive cues in the Web page (6) by applying adaptation techniques from the field of Adaptive Hypermedia.

4. ANALYZING THE WEB PAGE

The goal of analyzing the Web page is to identify Web page elements that potentially qualify to be enhanced with context-sensitive cues, and extract their associated metadata. For this purpose, we rely on Web pages that are semantically annotated using RDFa. RDFa, similar in purpose to microformats, is a W3C recommendation that allows Web designers to embed RDF triples in their Web page, thereby endowing any Web page element with explicit semantics using (existing) vocabularies and ontologies. Consequently, these semantic annotations allow us to correlate concrete Web page elements with the user's context, and therefore the Web page elements they annotate are candidates for injection of context-sensitive cues. A common use of RDFa is to indicate which resource(s) are depicted on a picture, or which resource corresponds to a certain string. The latter example is shown in the following HTML snippet, taken from the Brussels' tourist Website from our scenario: for the string "Atomium", the RDFa annotation indicates that this represents the rdfs:label of the resource <http://www.atomium.be/>. Consequently, the span tag containing the string "Atomium" is a candidate page element to be enhanced with context-sensitive cues.

```
<span xmlns:rdfs=".." about="http://www.atomium.be/"
      property="rdfs:label">Atomium</span>
```

With current Web development tools, content management systems and Web application frameworks now supporting RDFa generation[10], and major players such as Google and Yahoo! jumping on board, our assumption of RDFa presence in Web pages is not a liability. Furthermore, in [2], a mechanism is introduced which allows microformats (a currently very popular type of machine-readable markup) to be transformed to RDFa.

Our approach consists of parsing and extracting the RDFa annotations present in an existing Web page, which are then communicated as RDF triples to our COIN application on top of SCOUT. COIN subsequently uses these semantic descriptions of Web page elements in order to match Web page content to the mobile user's context, as provided by the Environment Model (see next section).

5. MATCHING WEB PAGE METADATA WITH THE ENVIRONMENT MODEL

The goal of the matching process is to match the semantic descriptions of the Web page elements to the user's Environment Model: i.e., the metadata about encountered entities, combined with the user's own profile. This matching process is performed by the COIN mobile application component, which runs on top of SCOUT. This component receives the extracted RDF triples, together with the page elements from which they were extracted, from the COIN Web page component, and stores them in a "Page Model". A Page Model is thus an RDF graph consisting of the RDF resources, properties and values found on the Web page, together with a reference to their corresponding Web page element. For example, for the RDFa example snippet of section 4, the triple "<http://www.atomium.be> rdfs:label 'Atomium'" is extracted, with the SPAN element containing the string "Atomium" as its corresponding Web page element. After the Page Model is built, the COIN application queries it to obtain the semantics of each annotated Web page element. In other words, it extracts the concrete RDF resource which this element represents, by for instance providing a label for the resource (e.g., the SPAN element represents the <http://www.atomium.be/> resource by providing an rdfs:label for it). This is done by the following SPARQL query, which is executed over the Page Model using SCOUT's Query Service:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX pm: <http://wise.vub.ac.be/SCOUT/pm-ns#>
SELECT ?subject ?pageEl
WHERE
{
   ?triple rdf:subject ?subject
   {
     { ?triple rdf:predicate rdfs:label } UNION
     { ?triple rdf:predicate dc:title } UNION
     { ?triple rdf:predicate foaf:name }
   }
   ?triple pm:relatedEl ?pageEl
}
```

This query returns resources (?subject) along with the Web page elements (?pageEl) by which they are represented . For this purpose, it looks for triples describing RDF resources using frequently used properties: in the above query, these are rdfs:label, dc:title, and foaf:name. For our example RDFa snippet in section 4, this query returns <http://www.atomium.be> as a resource, and the SPAN element as a corresponding Web page element.

The next step is to match the resources that were returned by the above query to metadata of encountered entities present in the Environment Model, and possibly to the user's profile. As an example of such metadata, consider the following RDF(S) snippet from the Atomium entity's online metadata source, which was added to the Environment Model after detection:

```
<http://www.atomium.be> rdfs:label "Atomium" ;
  dcmi:description "The Atomium is a monument built for
  Expo '58, ..." ;
  rdfs:seeAlso <http://en.wikipedia.org/wiki/Atomium > ;
  ...
```

To achieve the matching between resources found on the Web page and encountered entities, the COIN application generates a SPARQL query over the Environment Model, which performs this matching and collects the necessary information for a specific context cue. The query given below does this for the first context-sensitive cue of our scenario, i.e., the annotation of current and previously nearby entities, and the injection of the time spent nearby and optionally the entity's Website. It uses the resulting resources from the previous query (in our example, <http://www.atomium.be>) and includes the necessary conditions required for matching. In our scenario, we differentiate between entities that are currently nearby, that were nearby in the last four hours, and those nearby between four and twenty-four hours ago; therefore, the condition, specified in the FILTER clause, restricts the encountered entities to those encountered within the last twenty-four hours. In the SELECT clause, we return useful data (found in the environment model) on a matched entity required for the particular context-sensitive cue. In our scenario, this is the time at which it was encountered (denoted by scout:nearbyFrom), the time at which the entity was no longer nearby (denoted by scout:nearbyTo), and, if present (the OPTIONAL clauses), a link

[10] Some examples: an RDFa extension for Adobe Dreamweaver is available; a Drupal module (beta) supports RDFa generation, with other modules supporting RDFa under development; a RDFa library for Ruby on Rails is available; etc.

containing additional information (denoted by rdfs:seeAlso or foaf:homepage). The SPARQL query is as follows:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX em: <http://wise.vub.ac.be/SCOUT/em-ns#>
PREFIX scout: <http://wise.vub.ac.be/SCOUT/scout-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?entity ?from ?to ?webPage
WHERE {
  ?user rdf:type em:User ;
    scout:wasNearby ?entity .
  ?entity scout:nearbyFrom ?from ;
    scout:nearbyTo ?to .
  OPTIONAL
  {
    { ?entity foaf:homepage ?webPage }
    UNION
    { ?entity rdfs:seeAlso ?webPage }
  }

  FILTER ([current] - 24*60*60*1000 < ?from &&
    ?from < [current] &&
    (sameTerm(?entity,<http://www.atomium.be>) ||
     sameTerm(?entity, <...>[11])))
}
```

Such a SPARQL query is thus generated for each type of context-sensitive cue. The query below realizes the second part of our scenario: i.e., finding encountered restaurants fitting the user's tastes. This query therefore illustrates further personalization, by including conditions over the User Model in the query: i.e., taking into account the user's favorite type of cuisine (e.g., Italian):

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX em: <http://wise.vub.ac.be/SCOUT/em-ns#>
PREFIX scout: <http://wise.vub.ac.be/SCOUT/scout-ns#>
PREFIX resto:
  <http://gaia.fdi.ucm.es/ontologies/restaurant.owl#>
SELECT ?restaurant ?cuisine
WHERE
{
 ?user rdf:type em:User ;
   em:prefersCuisine ?cuisine ;
   scout:wasNearby ?restaurant .
 ?restaurant rdf:type resto:Restaurant ;
   resto:typeOfCuisine ?cuisine .

 FILTER([current] - 24*60*60*1000 < ?from
   && ?from < [current] && (sameTerm(?restaurant,
<http://www.brussel.be/Restaurant_La-Truffe-Noir>)
 || sameTerm(?entity, <...>)))
}
```

As can be observed, arbitrarily complex queries can be constructed by the COIN application to facilitate a wide range of context-sensitive cues, based on triples found in the Web page. The above query furthermore takes full advantage of the Environment Model as an integrated conceptual view over the user (User Model) and his environment (Relation Model), as it uses current and past environment information (i.e., restaurants the user is or has been nearby in the last twenty-four hours) and user information (i.e., his favorite type of food).

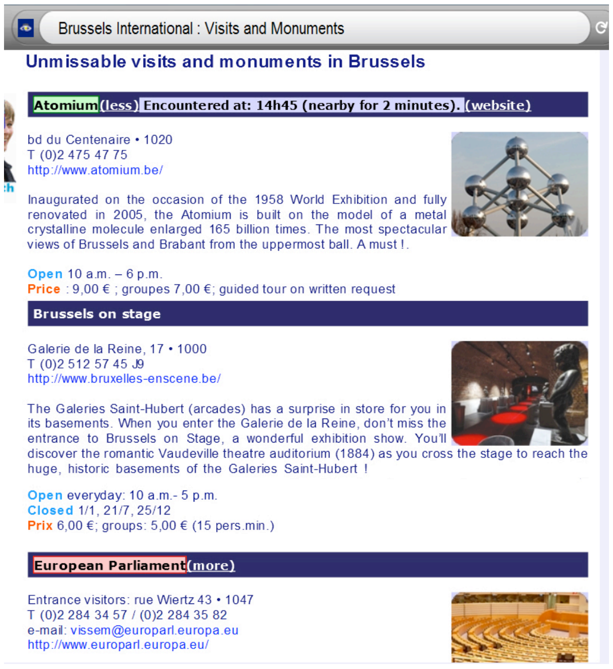
Based on the results of each query, suitable context-sensitive cues are generated. This final step is elaborated in the following section.

[11] <...> represents another candidate resource to match: for each of such resources, a sameTerm(..) operand is generated.

6. INJECTING CONTEXT-SENSITIVE CUES

The next step in our approach is to inject the context-sensitive cues in the existing Web page, based on the information that was gathered from the matching process described in the previous section. More specifically, for each Web page element that matched to the user's environment, the existing Web page is adapted on-the-fly to include the desired context-sensitive cues. The COIN Web page component contains the necessary commands to perform the desired adaptation upon request of the COIN mobile application.

The type of adaptation we employed is based on the well-known adaptive techniques from the field of Adaptive Hypermedia, first presented by Brusilovsky [5] and recently updated by Knutov et al [6]. As explained in [6], three main types of adaptation techniques exist: content adaptation, adaptive presentation and adaptive navigation. Content adaptation concerns showing/hiding or emphasizing/deemphasizing information, with reported techniques such as inserting, removing, altering or dimming content fragments, sorting of information, zooming or scaling and the use of stretchtext. Adaptive presentation concerns the alteration of presentation, with reported techniques such as altering layout (e.g., rearranging, zooming), link sorting and annotation, and combinatorial techniques (e.g., contextual links, site map generation). Adaptive navigation concerns altering the navigation structure, with reported techniques as link generation, local or global guidance and link hiding (disabling, removing, hiding). The interested reader is referred to [6] for more information on these techniques.



Brussels International : Visits and Monuments

Unmissable visits and monuments in Brussels

Atomium (less) **Encountered at: 14h45 (nearby for 2 minutes).** (website)

bd du Centenaire • 1020
T (0)2 475 47 75
http://www.atomium.be/

Inaugurated on the occasion of the 1958 World Exhibition and fully renovated in 2005, the Atomium is built on the model of a metal crystalline molecule enlarged 165 billion times. The most spectacular views of Brussels and Brabant from the uppermost ball. A must !.

Open 10 a.m. – 6 p.m.
Price : 9,00 € ; groupes 7,00 €; guided tour on written request

Brussels on stage

Galerie de la Reine, 17 • 1000
T (0)2 512 57 45 J9
http://www.bruxelles-enscene.be/

The Galeries Saint-Hubert (arcades) has a surprise in store for you in its basements. When you enter the Galerie de la Reine, don't miss the entrance to Brussels on Stage, a wonderful exhibition show. You'll discover the romantic Vaudeville theatre auditorium (1884) as you cross the stage to reach the huge, historic basements of the Galeries Saint-Hubert !

Open everyday: 10 a.m.- 5 p.m.
Closed 1/1, 21/7, 25/12
Prix 6,00 €; groups: 5,00 € (15 pers.min.)

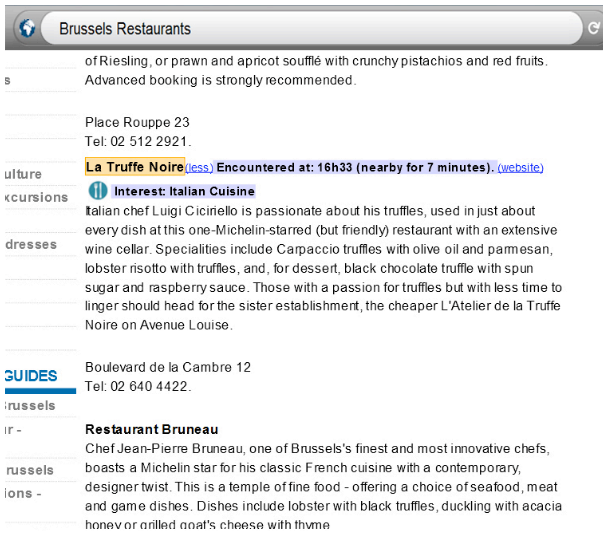
European Parliament (more)

Entrance visitors: rue Wiertz 43 • 1047
T (0)2 284 34 57 / (0)2 284 35 82
e-mail: vissem@europarl.europa.eu
http://www.europarl.europa.eu/

Fig. 2. Screenshot of the Brussels' tourist Website, enhanced using COIN

For the purpose of demonstrating our approach in this paper, we have selected three none-intrusive techniques: inserting fragments, adaptive link generation and stretchtext, and propose a new one, content annotation. The latter is inspired by link annotation, but instead of annotating a link, we annotate (some) relevant text in the Web page. We employ these techniques to realize our context-sensitive cues in the following ways. To highlight that a user has recently been nearby an entity represented on a Web page, the page element corresponding to this entity is highlighted by drawing a colored rectangle around it (content annotation). Depending on the time that has elapsed since the user was nearby the entity, different colors are used (configurable by the user). For example, we configured our component to use green for currently nearby entities, orange for entities that were nearby within the last 4 hours, and red for entities that were nearby more than 4 hours ago. Furthermore, a "more" link is generated next to the Web page elements corresponding to the encountered entities. When the user clicks on this "more" link, some additional context-aware information is inserted inline (stretchtext): the time at which the user last encountered this entity, the time he spent nearby, and, if available, a link to its homepage (adaptive link generation). For the second context-sensitive cue, i.e., indicating encountered restaurants (possibly serving the user's favorite cuisine), we elected to insert a recognizable icon for a restaurant, with an additional indication if it serves the users favorite type of food (content annotation).

Figure 2 shows a screenshot of the Brussels' tourist Website, viewed in the mobile Firefox browser (codenamed Fennec). Observe that the Web page element containing the string "Atomium" has been highlighted with a green rectangle, indicating the user is currently nearby the Atomium . The "more" link has been clicked, and reveals the time the user encountered the Atomium (14h45), how long he has been nearby (2 minutes), and a link to its Website. The "European Parliament" string has been highlighted in red, indicating that the corresponding entity was encountered more than four hours ago (its "more" link has not been expanded). Figure 3 shows an online restaurant listing where, next to the previously mentioned cue,, encountered restaurants are additionally indicated using an icon. Furthermore, in case the restaurant serves one of the user's favorite types of cuisine, the matched cuisine type is also inserted ("Italian Cuisine").



of Riesling, or prawn and apricot soufflé with crunchy pistachios and red fruits. Advanced booking is strongly recommended.

Place Rouppe 23
Tel: 02 512 2921.

La Truffe Noire(less) **Encountered at: 16h33 (nearby for 7 minutes).** (website)

Interest: Italian Cuisine

Italian chef Luigi Ciciriello is passionate about his truffles, used in just about every dish at this one-Michelin-starred (but friendly) restaurant with an extensive wine cellar. Specialities include Carpaccio truffles with olive oil and parmesan, lobster risotto with truffles, and, for dessert, black chocolate truffle with spun sugar and raspberry sauce. Those with a passion for truffles but with less time to linger should head for the sister establishment, the cheaper L'Atelier de la Truffe Noire on Avenue Louise.

Boulevard de la Cambre 12
Tel: 02 640 4422.

Restaurant Bruneau

Chef Jean-Pierre Bruneau, one of Brussels's finest and most innovative chefs, boasts a Michelin star for his classic French cuisine with a contemporary, designer twist. This is a temple of fine food - offering a choice of seafood, meat and game dishes. Dishes include lobster with black truffles, duckling with acacia honey or grilled goat's cheese with thyme

Fig. 3. Screenshot of a Brussel restaurant listing, enhanced using COIN

7. IMPLEMENTATION

In this section, we provide some details about the implementation of the SCOUT framework, the COIN mobile application and the COIN Web page component. Afterwards, we discuss possible ways for Web developers and mobile users to deploy our approach.

7.1 Scout Framework and COIN Mobile Application

SCOUT has been developed in JavaME (due to its portability across devices), using the Connected Limited Device Configuration (CLDC 1.1) and the Mobile Information Device Profile (MIDP 2.0). It currently supports various detection techniques, including RFID, NFC and QR codes, with fitting proximity strategies based on the range of the sensing technologies and GPS coordinates, to detect the presence of nearby entities. The Environment Model is implemented as a virtual, partially cached view over the User Model, Relation Model and the online RDF(S) sources of encountered entities. The MicroJena library[12] is employed to store and access RDF data locally. As it does not yet provide support for SPARQL querying (although efforts are currently underway to implement this support), we temporarily use an external query server to handle SPARQL queries. The COIN mobile application is realized as an application on top of SCOUT, and is also written in JaveME. We have tested SCOUT and COIN on a Nokia N79 and a HTC Touch Cruise device.

7.2 COIN Web Page Component

The COIN Web page component is implemented in Javascript, and provides three core functionalities: RDF triple extraction, communication with the COIN mobile application and Web page adaptation.

The COIN Web page component converts the found RDFa annotations into RDF triples using the RDFa library provided by the W3C[13]. It sends the extracted triples to the COIN application via a HTTP request, and subsequently receives (from the COIN mobile application) appropriate Web page adaptation commands as a response. A generic Javascript library is provided, which consists of objects representing these adaptation commands. More specifically, each object implements an adaptation technique to be applied, and keeps an internal element identifier (identifying an element from which RDF triples were extracted), and additional information (e.g., text/link to be inserted, color of highlighting). The Web page's DOM is subsequently utilized to locate the relevant element, and perform the necessary changes.

7.3 COIN Script Deployment

As mentioned in the previous section, our approach relies on a Javascript running in the targeted Web page. This script can be inserted in any Web page using the (X)HTML script tag. The insertion can be done in two ways. In case the mobile user wants to employ our approach to inject context-sensitive cues in any existing (semantically annotated) Web page, he can install a browser plugin which automatically injects our COIN script tag at client side into each downloaded Web page. We have developed such a plugin for the mobile version of the Firefox browser (called

[12] http://poseidon.elet.polimi.it/ca/?page_id=59

[13] http://www.w3.org/2006/07/SWD/RDFa/impl/js/20070301/

Fennec), which implements this functionality. We successfully tested this plugin on a HTC Touch Cruise device.

In case Web designers are interested in adding context-sensitive cues to their Website, without the need for users to install a browser plugin, they can manually insert the script tag referring to our Javascript file in their Web pages. By integrating it into page templates, the COIN script can be easily duplicated on every Web page.

8. RELATED WORK

In existing Adaptive Hypermedia Systems (AHS), adaptation is mostly pre-engineered: the foreseen adaptation needs to be engineered during the design of the adaptive system. Existing systems utilize adaptation conditions (which mostly refer to user or context information), denoting when to show/hide certain content depending on context or user model. A representative system is example is AHA! [5]. Other systems, such as WSDM [6], use ECA (Event-Condition-Action) rules to perform adaptation: after a certain event occurs (e.g., a page load), an adaptation action is executed in case the associated condition holds. In recent works [7, 8], aspect-oriented techniques are utilized in order to separate adaptation concerns from the rest of the design, and "weave" adaptation logic into the regular design/code. Our approach does not require this adaptation engineering: it works on existing, already deployed Web pages, and doesn't require any adaptation-specific engineering.

An extension for WebML is presented in [9] to support active context-awareness, i.e., adaptation that is triggered from the client-side. However, this adaptation logic still needs to be explicitly engineered in the Website. In [10], a number of SPARQL queries can be embedded inside a Web page, which represent the personalization logic. These queries may reference the user's profile or online RDF sources, and are executed by a client-side GreaseMonkey script. However, these queries need to be provided by the Web developer (i.e., explicitly engineered), and only a desktop setting is envisioned (i.e., no environment data is considered). In [11], the author explores asynchronous technologies (e.g., AJAX) to increase the efficiency and effectiveness of adaptive hypermedia systems, by dynamically replacing certain parts of the page by their adapted counterparts. Similarly, COIN exploits client-side Javascript communication, yet by sending adaptation commands (e.g., annotate text, generate link) instead of page fragments to the in-page Javascript, which subsequently adapts the Web page accordingly.

Our approach leaves the user in charge of his own browsing session. In some related systems, such as [9, 12], the user' browser is automatically navigated towards a page containing information on his current situation. However, we have consciously decided on utilizing adaptation techniques which are non-disruptive: i.e., we avoid techniques which navigate towards a different page or alter the page's structure (e.g., link ordering), as such techniques may annoy and confuse users, and most likely lead to the creation of an incorrect mental map [13].

Our technique to add context-sensitive cues follows a current trend towards client-side personalization or adaptation. For example, Mozilla GreaseMonkey is a plugin for Firefox which allows users to write Javascripts (so-called user scripts) that adapt the Web page currently being browsed; Mozilla Jetpack allows users to write Firefox extensions for the same goal. Bookmarklets are popular ways of executing random Javascript code in any browser and on any Web page, thus enabling cross-browser adaptation of Web pages.

In contrast to existing approaches for context-aware information delivery, we use a decentralized, distributed and Semantic-Web based approach to context-awareness, where no intermediary services are required to provide personalized, context-sensitive data. As such, it is a perfect fit for our client-side solution, as all the necessary context information is directly available locally. In many other approaches, either a centralized Information System (IS) is employed to store and maintain all context-specific data (e.g., [14, 15]), or a centralized service is used which acts as an integrated view over distributed data sources (e.g., [16, 17]). Although such centralized approaches offload a lot of work from the mobile devices themselves, it is also less scalable and flexible, as every client needs to communicate with the same central service. Also, we argue that the possibilities of mobile devices will keep increasing, thus reducing the need for so-called "thin" clients. Finally, as the SCOUT framework employs Semantic Web technology (like in e.g., [18, 19]) to integrate heterogeneous data sources and to represent the user's context and environment, the semantic annotations extracted from the Web page content can be directly matched to the user's contextual information. In conclusion, to the best of our knowledge, there is no other approach that allows on-the-fly, non pre-engineered and personalized context-awareness injection in existing Websites.

9. CONCLUSION AND FUTURE WORK

In this paper, we presented a client-side approach to inject personalized, context-sensitive cues into existing (RDFa annotated) Web pages, in order to assist the mobile user in his browsing activity. To achieve this, the Web page content is matched to the user's context, and useful contextual information is injected into the existing Web page, and/or information related to his context is highlighted. The approach is built on SCOUT, a mobile application development framework that constructs an integrated a conceptual view of the user and his environment. The presented solution consists of two components. Firstly, there is the COIN mobile application, which resides in the SCOUT's Application Layer, and obtains contextual information from SCOUT's Environment Model. Secondly, there is a Web page component (in our implementation a Javascript), which extracts RDF triples from the Web page and performs the adaptations on relevant page elements as directed by COIN. The Javascript is either added on the client-side by a mobile browser plugin, or already present on a Web page, i.e., included by the Web designer. Our approach thus does not require pre-engineering from the Web developer: it effectively allows adding context-sensitive cues to any RDFa annotated Web page.

In future work, we plan to build a more complete and elaborate Page Model, by also taking into account the metadata of the Web pages the original page links to. Based on this extended Page Model, we expect to be able to offer richer context-sensitive cues, and offer new types of context-sensitive cues (e.g., adaptive guidance). Finally, we also aim to support additional annotation types (e.g., microformats).

10. REFERENCES

[1] Van Woensel, W., Casteleyn, S., De Troyer, O. 2009. A Framework for Decentralized, Context-Aware Mobile Applications Using Semantic Web technology. In *On the*

Move to Meaningful Internet Systems: OTM 2009 Workshops, pp. 88--97, Portugal

[2] Adida, B. 2008: hGRDDL: Bridging microformats and RDFa. In *Web Semantics: Science, Services and Agents on the World Wide Web, 6 (1)*, 54--60, Elsevier Science Publishers B. V.

[3] Brusilovsky, P.: Methods and techniques of adaptive hypermedia, *User Modeling and User-Adapted Interaction. 6 (2-3)*, pp 87-129, Springer Science+Business Media B.V. (1996)

[4] Knutov, E., De Bra, P., Pechenizkiy, M. 2009. AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques. In *The New Review of Hypermedia and Multimedia, Volume 15, Issue 1*, pp 5—38, Taylor & Francis, Inc.

[5] De Bra, P. and Calvi L. 1998. AHA! An open Adaptive Hypermedia Architecture. *The New Review of Hypermedia and Multimedia 4*, pp. 115—139, Taylor & Francis, Inc.

[6] Casteleyn, S. 2005: Designer Specified Self Re-organizing Websites. PhD thesis, Vrije Universiteit Brussel, Belgium

[7] Bausmeister, H., Knapp, A., Koch, N., Zhang, G. 2005. Modelling Adaptivity with Aspects. In *International Conference on Web Engineering*, pp. 406--416, Australia

[8] Casteleyn, S., Van Woensel, W., van der Sluijs, K., Houben, G.J. 2009. Aspect-Oriented Adaptation Specification in Web Information Systems: a Semantics-based Approach. In *The New Review of Hypermedia 15 (1)*, Peter Brusilovsky and Paul De Bra (eds.), 39--91, Taylor and Francis.

[9] Ceri, S., Daniel, F., Facca, F., Matera, M. 2007. Model-driven Engineering of Active Context-Awareness. In *World Wide Web 10 (4)*, 387--413, Kluwer Academic Publishers.

[10] Ankolekar, A., Vrandecic, D. 2008. Kalpana – Enabling Client-Side Web Personalization. In *19th ACM conference on Hypertext and hypermedia (HT '08)*, pp. 21--26, USA.

[11] Putzinger, A. 2007. Towards Asynchronous Adaptive Hypermedia: An Unobtrusive Generic Help System. In *Lernen – Wissen – Adaption (LWA '07) Workshop*, pp. 383--388, Germany

[12] Challiol, C., Fortier, A., Gordillo, S.E., Rossi, G. 2008. Model-based concerns mashups for mobile hypermedia. In *6th International Conference on Advances in Mobile Computing and Multimedia (MoMM '08)*, pp. 170--177, Linz, Austria.

[13] Brusilovsky, P. 2007. Adaptive Navigation Support. The Adaptive Web, pp. 263--290

[14] López-de-Ipiña, D., Vazquez, J.I., Abaitua, J. 2007. A Context-aware Mobile Mash-up Platform For Ubiquitous Web. In *3rd IET International Conference on Intelligent Environments*, pp. 116--123, IEEE, Germany.

[15] Tummala, H., Jones, J. 2005. Developing Spatially-Aware Content Management Systems for Dynamic, Location-Specific Information in Mobile Environments. In *3rd ACM international workshop on Wireless mobile applications and services on WLAN hotspots, Mobility support and location awareness*, pp. 14--22, ACM, Cologne, Germany.

[16] Cappiello, C., Comuzzi, M., Mussi, E., Pernici, B. 2006. Context Management for Adaptive Information Systems. In *Electronic Notes in Theoretical Computer Science 146*, 69—84.

[17] Xue, W., Pung, H., Palmes, P.P., Gu, T. 2008. Schema matching for context-aware computing. In *10th international conference on Ubiquitous computing*, pp. 292--301, Korea

[18] Euzenat, J., Pierson, J., Ramparany, F. 2008. Dynamic context management for pervasive applications. In *Journal of Knowledge Engineering Rev. 23*, 21--49

[19] Frkovic, F., Podobnik, V., Trzec, K., Jezic, G. 2008. Agent-Based User Personalization Using Context-Aware Semantic Reasoning. In *12th international conference on Knowledge-Based Intelligent Information and Engineering Systems (KES '08)*, pp. 166--173, Zagreb, Croatia.