What'sUp: A mobile application for searching ongoing cultural events

João Amaro Silva, Paulo Urbano, and João Balsa

Faculty of Sciences, University of Lisbon
Campo Grande 1749-016 Lisboa, Portugal
{pub,jbalsa}@di.fc.ul.pt
fc35975@alunos.fc.ul.pt
http://www.fcul.pt

Abstract. What'sUp is a project which focuses on the development of a tourism and culture oriented mobile context-based application that helps the user to explore local cultural offers. The goal is to inform and to lead the user about the ongoing and upcoming cultural events in the user's geographical area, without the need to specifically state his location or the exact date/time of the intended event. We built a Web ontology in OWL 2, which answers four questions about the cultural events: *When*, *What*, *Where* and *How Much*. These events are on the Web and its most important data is periodically extracted, refined and inserted in the ontology, using web scraping and Definite Clause Grammar. The user's sentences in natural language are transformed in queries that run in a unified OWL 2 database. What'sUp can greatly improve the access to context-based information in digital cities, through the use of natural language interaction and Semantic Web technologies.

Keywords: Semantic Web, Ontologies, Natural Language, Mobile Application, Grammar

1 Introduction

We are witnessing a widespread interest in mobile recommender systems for tourists and city inhabitants, that tailor their content based on the user's current location, personal profile, activity, weather conditions or other contextual factors [1,2,3]. Mobile context-aware recommendations systems are the natural outcome of the advances in positioning technologies and the increased sensing capabilities of smartphones. A central issue in mobile recommendation systems is the exclusive exposition to filtered content that matches user profiles, the filter bubble [4], preventing the access to novel and serendipitous content [5]. The support of serendipitous discovery of interesting events in an urban environment will certainly help urbanites that emphasize spontaneity. Semantic Web is a promising technology for organizing, sharing, integrating and searching distributed information about city cultural events, enabling machine processing of the massive amount of event information available on the Web. Ontologies will

play a fundamental role, structuring web documents with a semantic vocabulary (understandable by machines) that may be shared by different cultural event providers [6]. Currently, most information about city events is scattered across event providers web sites and on-line cultural agendas. Traditional web information is targeted towards humans, but must be extracted and converted to the ontologies vocabularies, in order to be manageable by machines. The scenario of an universal and unambiguous adoption of Semantic Web technologies is still unreal.

What'sUp is a mobile recommendation system for supporting serendipitous discovery of events in an urban context. It is a Semantic Web mobile application that guides the user throughout the city activities, making use of contextual data, location and time. We built a web ontology in OWL 2.0, for representing cultural events in a city. The content is periodically extracted, refined and updated from Lecool web site, a free weekly e-magazine, featuring a selection of cultural events and leisure activities in Lisbon, as well as MyGuide, a web site about cultural events occurring in Portugal. In order to extract events information from LeCool e-magazine and Myguide web site, we rely on web scraping and natural language processing with Definite Clause Grammars (DCGs) to deal with the events temporal aspects. Users can discover what is happening "here and now" by querying the mobile application using natural language. For example, he can type in the smartphone "What is up now?" or "What is up today at noon?" and the natural language queries are transformed into corresponding SPARQL queries, using a keyword matching process. The results are sorted by proximity to the mobile device's GPS position. The results list top events are the nearest ones to the user. The user can restrict the search by choosing the geographical search radius around him. The event information is enriched using the Google Maps service, providing the event location on the map and presenting an itinerary.

2 Related Work

In order to establish a basis for comparison, in this section we will overview similar research and applications that are somehow related to What'sUp project.

2.1 STAAR System

The Semantic Tourist information Access and Recommending System (STAAR) [10] exploits the advantages of Semantic Web technology providing touristic guidance through Web and smartphone.

Helping a tourist who visits a travel destination for the first time is the main aim of STAAR, by giving advices about which places to take a look and even the best itinerary from a starting point. Ontologies play an important role in STAAR for representing knowledge about touristic resources, including also personal touristic interests (points of interest, activities, events and services). Ontologies can model (i) what a traveller can see and visit at a given destination;

(ii) the location of interesting places and what their attracting characteristics are; and (iii) what are the relations between travel resources. Semantic and geographic enriched touristic data is matched with touristic preferences to search and recommend information in a smart way. One important feature of STAAR is context-awareness: touristic recommendations rely heavily on the users personal profiles and on their positions through GPS navigation systems.

Recommendations are the response to user friendly interaction transformed in SPARQL queries. Search can be done at 3 different levels of complexity in a smartphone using an interface that is based on the ontology. Simple queries like "Find a luxury restaurant" can be made browsing ontology concepts. More complex questions are possible using search with constraints on a particular concept through its properties, like "Find a well-known place related to an architecture topic". The most complex queries rely on the specification of property constraints over multiple objects, making possible queries like "Find dinning service place with European-style, nearby a mall that sells clothes". Finally, STAAR can enrich travel data extracting related information from Linked Data repositories and also from on-line data services such as Flickr or Youtube.

Despite STAAR system helps the user with location-based problems in an effective way, it does not consider the date of any event or the scheduling of the tourist itineraries that are calculated. The temporal part is one of the main aspects that our system considers.

2.2 Eiffel

The main goal of the Eiffel project is to provide users with an intelligent and multilingual semantic search engine dedicated to the tourism domain, allowing tourism operators and local territories to highlight their resources. The end users will be able to use the touristic search engine in order to organize their trips on the basis of contextualized, specialized, organized and filtered information [8]. By posting queries, depending on their profiles, they will have access to distributed and highly heterogeneous data in the tourism domain.

Eiffel application is driven by a tourism ontology based on conceptual graphs [7] and is able to automatically detect, select and classify tourism content distributed in the web, populating the ontology. In particular, it is capable to extract temporal and geographic information about touristic offers. Eiffel can also analyse touristic behaviours, extracting users models and profiles, in order to be able to give personalized recommendations, through the use of a reasoning engine.

Researchers from the Eiffel project adopted a symbolic approach relying on patterns and rules for the detection, extraction and annotation of temporal expressions in unstructured web pages. Their method is based on the use of transducers [9], being able to deal (i) with complex and imprecise temporal information, (ii) heterogeneous tourism web pages and (iii) the difficulty of linking the temporal and spatial extracted information to the proper tourism object.

What'sUp differs from the Eiffel approach mainly in terms of the techniques used for extraction of event information from web pages, being limited to struc-

tured web pages, where event's temporal and spatial information can be easily tracked. It differs in the ontology language used: OWL 2.0, in our case. Note also that Eiffel was not designed as a mobile application.

Comparison to Siri

We may find certain characteristics of our application alike some functionalities offered by Siri, that allows a person visiting a new city to quickly discover the best places to have dinner or the best cultural spots within the city, e.g., monuments, views, historical locations. Nevertheless, our application is not place-driven, but rather event-driven. We focus on events that occur within the space of a month or during all the year and consider always the temporal aspect (date and time). That aspect is not so deeply analysed in Siri, as it is driven by places and not cultural events. Therefore, Siri will not be so reliable and precise when returning results about ongoing or upcoming events in a city, with all the details that are stored in the Web. Another main difference between these two approaches is the possibility of producing the input using voice, faculty which is not offered in our system, once we give priority to a more reliable and precise searching mode, using a DCG grammar. Also, voice input analysis has many complex issues that we do not mean to address in this project.

3 The Ontology

The fundamental element in our project is the ontology, as it is the place where data is stored, queried and retrieved. We decided to create our own ontology, named What'sUp Ontology, given the characteristics and objectives of our project. However, we took some guidelines in consideration [11]. This ontology was built in such a way that, in the future, it could be used on the Web. Moreover, it is extensible and reusable for other purposes. There are already many ontologies on the Web that can be reused and extended in order to create our own ontologies. One of them is Time Ontology, which defines temporal relationships and uses multiple time definition levels (time intervals, time instants) as well as different time units (weeks, days, hours, minutes). We analysed this ontology to verify if it had potential to help us building our ontology, as its domain is closely related to what we need for describing our cultural events characteristics, namely the temporal aspect. We came to the conclusion that this ontology is too simple to cover so many cases of time relations that we had to handle. Despite that some of the time expressions that we deal with are capable of being described using this ontology, there is a significant set of expressions that could not be represented with the terms defined on the Time Ontology. However, we can still make a term mapping between Time Ontology and What'sUp Ontology, using the *owl:sameAs* built-in property, to state that two URI references refer to the same concept. For example, our ontology has an enumerated class named *WeekDay* that represents the same concept as the enumerated class *DayofWeek* in the Time Ontology.

3.1 Ontology structure

Our OWL 2.0 ontology has been created to shape the event structure in terms of its temporality (*When*), spatiality (*Where*) and the type of event *What*. We will focus on the most important ontology classes and properties. We will have a taxonomy of events with root class *Event* and each particular event will be an instance of the category in the taxonomy that best characterizes it. Feira da Ladra, a Flea Market in Lisbon, will be an instance of *Fair*, for example. As the *Event* class incorporates the event type, we will need further classes for the the remaining event aspects: *When*, *Where* and *How Much*. Three properties are defined to relate the *Event* class with those three aspects: *hasLocation* (domain *Event* and range *Location*), *hasWhen* (domain *Event* and range *When*) and *hasCost* (domain *Event* and range *xsd:String*, as illustrated in Figure 1. As said before, all these events are categorized in types, such as *Theatre*, *Concert*, *Party*, *Gallery Exposition*, among others (all subclasses of the OWL class *Event*). Each *Location* individual has a property representing its geo-coordinates. We will focus on the time aspect, as it is the most complex aspect of our ontology, due to its variety and heterogeneity.

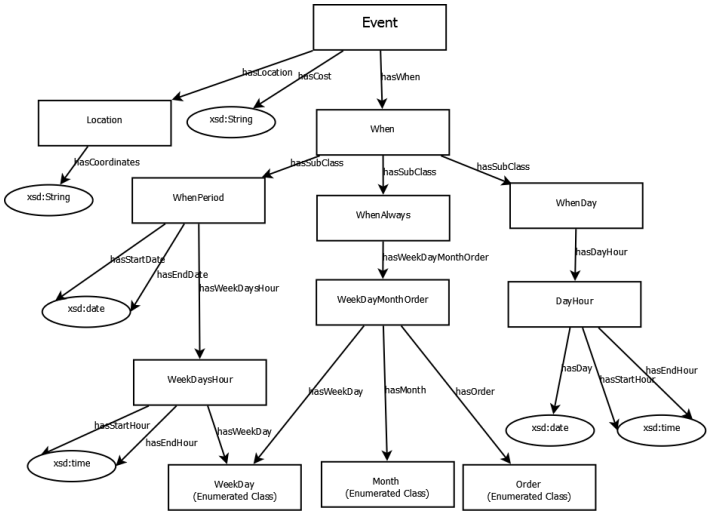

Fig. 1. The What'sUp Ontology structure

Every *Event* instance is related with a particular *When* instance through the object property *hasWhen*. We have divided the temporal specifications of the events in three main categories (being represented as subclasses of the class *When* in our OWL 2 ontology):

WhenDay. Corresponds to a particular date and time. The events that have this time relation type occur in certain dates, restricted within the period of one

day and at a certain hour or during a certain hourly period. Examples are: “23rd April at 11 pm” or “25th April from 10 am to 1 pm and from 2 pm to 6 pm”. We may have one or more hourly periods in a single day on these event types, like “22nd May from 10 am to 1 pm and from 2 pm to 6 pm”. This configuration generates two *WhenDay* individuals, one for each time interval, although with the same day. For events that occur in several days, for example “3rd, 5th and 14th July at 1 pm”, it generates one individual for each pair <date - time>. In its turn, each of these *WhenDay* individuals have an associated *DayHour* individual, where the proper date value and time value/interval are specified. One *WhenDay* individual has one and only one *DayHour* individual associated. The *DayHour* class deals with the aggregation of a particular date and a starting hour or an hourly period. This means that, in this last example, three individuals of *WhenDay* will be generated. This feature will simplify queries and answers will be more specific and focused on the user’s interests.

WhenPeriod. Periodic events will be handled by another category, *WhenPeriod*, corresponding to consecutive dates and represent repetitive time information. There is always a starting date and an ending date in a *WhenPeriod*. Examples are: “from 23rd of June to 15th August at 1 pm” or “Thursday, Friday and Saturday at 11 pm ,from 1st to 3rd July”. Analogously to $<$*WhenDay-DayHour*$>$ relation, a *WhenPeriod* instance will always have an individual of the class *WeekDaysHour*, in order to aggregate a particular weekday or weekday interval and a starting hour or hourly period. This relation has a one-to-one cardinality. We have an enumerated class that lists all the weekdays, named *WeekDay*.

WhenAlways. The third category, *WhenAlways*, contains events that happen during all the year, like for instance, the Flea Market that opens “every Tuesday and Saturday from 6 am to 6 pm” or the Old Pipes market that opens “every first Sunday of each month, from 10 am to 6 pm”. As it happens in the other *When* class types, there’s a one-to-one relation between the *WhenAlways* instance and the *WeekDayMonthOrder* instance, which aggregates a particular nth weekday (or weekday interval) of the month and an hour or hour interval that happens during all the year or during a certain month. Two object properties are used for accessing the weekday and order of a certain instance of *WeekDay-MonthOrder*, namely *hasWeekDay* and *hasOrder*. *Order* is also an enumerated class representing the nth occurrence of a certain weekday in a month: first, second, third, fourth or last. *Month* lists all the months in a year. We have the possibility to represent the “last Sunday of each month” as an instance of the class *WeekDayMonthOrder*, which aggregates the class *WeekDay* with the class *Order* and *Month*. For example, the Flea Market occurring any Tuesday and Saturday, from 6 am to 6 pm will imply one instance of *WhenAlways* with one instance of *WeekDaysHour* with value of the property *hasWeekDay* aggregating Tuesday and Saturday and the time interval associated with that instance.

Each *DayHour*, *WeekDayMonthOrder* or *WeekDaysHour* instance will have a start time or a hourly period. We did not create special classes to handle those concepts and, instead, used two Datatype properties: *hasStartHour* and *hasEndHour*. While only the first one is necessary to define a starting hour, both are used to define an hourly period.

An event like a theatre show from 13th to 30th June, on every Tuesday and Wednesday at 9 pm, Thursday, Friday and Saturday at 9.30 pm. There will be an instance of *WhenPeriod* for every <weekdays - time> pair of the event: one for Tuesday and Wednesday and another for the rest of the weekdays.

4 System Overview

In this section, we describe the multiple modules of our application with more detail and explain how they are related with each other.

4.1 Web Scraping

The scenario that was previously described in the Introduction section would allow us to obtain useful information from the Web in a more suitable way. As this scenario is not yet a reality, we must gather the information we need from multiple sources on the Web, with different knowledge structures.

For accomplishing this task, we use a Firefox plug-in to perform Web scraping. A scraper is a piece of software that dissects a web page and extract useful data from it. Using this tool, we can create one different scraper for each website structure model, according to its characteristics. The scraper scans the target website HTML code for a specific sequence of tags. This sequence can be discovered by analysing the HTML source code, locating the content that we want to extract and identify the tags that exist before and after this content. We could define more complex sequences using regular expressions. The scraper scans the page until it finds these sequences and therefore it is able to extract the right pieces of information to a file in an organized way. All the events information existing on the websites can then be processed as we wish.

With this tool we can apply a certain scraper to multiple web pages. This is useful to fetch all the issues of a cultural journal, for instance. We can also create scripts to execute scrapers in a regular period. We use this functionality to regularly and automatically execute the scrapers and update the information in the application event database (Web Ontology).

4.2 Grammar Predicates Evaluation

To characterize and recognize temporal expressions we use the DCG formalism with Prolog. Although more powerful than Context-Free Grammars, we use DCGs (with context-free rules only) due to its simplicity and ease of use.

Grammar rules were developed taking into account a thorough analysis made on the scraping results. The relevant information is the *When* feature of an event.

As mentioned above, temporal expressions might have values with very diverse structures, incorporating information regarding months, weekdays, times of day and possibly involving combinations of these features. For instance, we need to recognize expressions as different as:

- "Todos os sábados, das 10h às 15h" (*every saturday, from 10 am until 3 pm*)
- "Até 8 de Outubro, às 18h" (*until October 8, at 6 pm*)

Grammar rules are organized in such a way that for each relevant *when* expression component a non-terminal symbol is defined, while the main syntactic category (`quando/1`) has as argument the temporal expression recognized structure. We defined a few dozen relevant temporal expression components, and illustrate here one of them.

To recognize expressions representing sets of time intervals in a day, the main rules are:

```
% top level category rule
quando(intHorasL(HMHM)) --> intHorasL(HMHM).

% recognize one time interval
intHoras(intHoras(Hora1,Hora2)) --> det,hora(Hora1),conector,hora(Hora2).
```

These rules, together with `intHorasL/1`, that is just a list builder, allows the system to recognize the component "das 10h às 15h" mentioned above, providing the structure:

`quando(intHorasL([intHoras(hora(10), hora(15))]))`

that will subsequently be processed in order to update the ontology.

4.3 Data handling and modification

We have a set of conditions defined to decide if a cultural event has all the elements to be included on the ontology, so it could then be queried. We must ensure that all the ontology individuals are considered in a query and no events are left behind in a response, due to lack of information on those events properties.

Coordinates Extraction The events that are scraped from the Web have a location associated. In order to provide more information to the user about every event, we focused on obtaining the geographical coordinates of each location. We use Google Geocoding API service, along with a performance enhancement solution that we developed. This service is subject to a query limit of 2,500 geolocation requests per day and the number of requests to the service has a limit per second and per IP. To bypass that obstacle, we decided to maintain a database with the Address – Coordinates relation of every location. This approach brings the advantage of not making requests every time a location is extracted from the Web, but only when that location is new and is not in our mapping structure. If it is not mapped yet, we do a geolocation request and add the result to our structure, for future use. This solution makes the event classification process

faster (less requests required), because successive geolocation requests require a time interval between each request. So, the lesser requests are done, the better classification time we get, in order to then populate the ontology. Nevertheless, location is not a mandatory requirement.

Name, Date and Category handling In order to enter the ontology, an event must comply with some requirements. First, an event must have a name, so it could be presented on the results list. The String containing the name is modified, so it can be inserted on the ontology without any inconvenience (unwanted characters are replaced). This goes to any expression that is to be inserted on the ontology. Also, all Datatype values that are introduced on the ontology, such as *xsd:date*, *xsd:string* or *xsd:time* values, are assured to have the right format, so any ontology inconsistency problem is discarded. As we considered the temporal aspect of the cultural event the most important in our application, it is a requirement to all the events stored on the ontology. Any event that lacks date information will be discarded and, consequently, will not be inserted on the ontology. The events which date expression could not be processed by our grammar predicates, will as well be discarded. Another requirement that events must fulfil is having a valid category. There are several categories that we defined and our grammar has predicates to validate those categories. If a category expression is not recognized or if it is not a category that we want to include in our system (for example, an event of the type "Restaurant", which is not properly a cultural event but a place), the event containing that category will not be inserted into the ontology.

5 An Illustrating Example

In our project, data suffers multiple modifications and is dynamically handled. The path from the web page HTML code to the ontology is complex and must be explained and illustrated. We will pick a simple time expression as an example and will describe all the phases that occur in the process. We will consider the following time expression:

`De 20 de Junho a 2 de Julho, às 21h30`

(From 20th June to 2nd July, at 9.30 pm)

This expression denotes a time interval (measured in days) that lasts from 20th June to 2nd July, and has an hour indication, referring to 9.30 pm.

The expression is scraped from the Web and stored in a document. There, a Java application reads the expression and modifies it according to certain rules, so it can be in conditions of being evaluated by the Prolog predicate, which was previously defined in the Definite Clause Grammar. The set of modifications made by the Java application include splitting the expression in words, each of them separated by commas. Non-numerical words are surrounded by ' symbols.

Some additional characters are concatenated to the final string for indicating that an hour or minute expression is present, 'h' and 'm', respectively). The result of this treatment, in our example, is the following:

```
'De',20,'junho','a',2,'julho','às',21,'h',30,'m'
```

('From',20,'june','to',2,'july','at',21,'h',30,'m')

This format allows the DCG grammar to recognize this expression and therefore, evaluate it, returning a result from that evaluation. So, considering our example, the DCG rule responsible for evaluating this expression is:

```
quando(intDiaMesHora(IDM, Hora)) --> intDiaMes(IDM), horas(Hora).
```

(when(dayOfMonthIntervalHour(DMI,Hour)) –> dayOfMonthInterval(DMI), hours(Hour).)

The rule returns the following result:

```
intDiaMesHora(diasMeses(diaMes(20,06,2012),diaMes(2,07,2012)),[horaMinuto(21,30)])
```

(daysOfMonthIntervalHour(daysOfMonth(dayOfMonth(20,06,2012), dayOfMonth(2,07,2012)),[hourMinute(21,30)]))

After evaluating each String and parsing the result, we create an OWL instance of the class *Event* that represents the event containing this time expression and fills in its properties, namely the date related property, according to the parsed results.

The *Where* and *How Much* expressions are considered simple strings given its evaluating complexity. However, these expressions still receive minimal treatment, as they also must be introduced on the ontology. Figure 2 shows a Protégé 4.2 screen and illustrates how the *When* instance is created and filled in on the ontology.

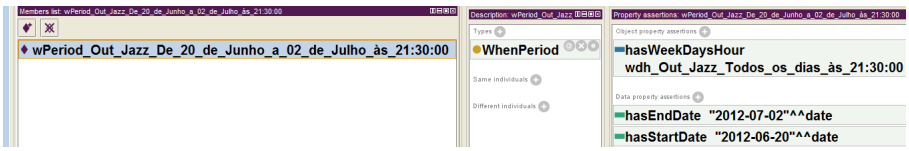

Fig. 2. The resulting *When* instance created on the example

6 Mobile Application Description

What'sUp is developed for Android mobile OS and its interface is divided into multiple activities (Android screens). The user navigates the application by exploring the different activities during his experience. Our application uses the

Google Maps service for providing the event location on the map or even point an itinerary for the event. This location data is sent to the server, together with the user natural language question. The questions could be based on the date ("What is up today?") but could also have a time restriction, as an hour in the day ("What is up today at 12 am?"). This query returns the events that occur on the present day at 12 am. The query expression can also include event category restrictions: "What can I watch in Cinema today at 12 am?". The conversion of these questions is made on the server side. They are transformed in SPARQL queries, according to a keyword-based translation process. The user natural language question is scanned and if it contains some pre-defined keywords, like "today","afternoon" or "night", a pre-set SPARQL query is executed on the ontology. Time restrictions (corresponding to a number from 0 to 23) or any event category (Cinema, Party) restrictions are also taken in consideration. The user does not need to state his location because it is automatically extracted from the GPS system. We assume that the user is always on our target city perimeter, as our application geographical scale concerns only a city (not a worldwide scale) and all our application content is location dependable. For those reasons, we did not concern about time-zone adaptations on the event dates, depending on the user location, because there's no space to ambiguity on the time scale.

The user can also restrict the search results, by choosing the geographical search radius around him. The search results are be filtered and only the events within the desired radius (500 or 2000 meters,for instance) are displayed. The details of an event include its name, category, date, location, price and distance from the user.

The results are presented according to their date and location. The resulting events that appear in the applications GUI (as a response to the user's queries) are sorted by proximity to the mobile device's GPS position. This way, the results list top events are the nearest from the user. The events that lack coordinates information are put in the bottom of the list.

7 Conclusion

We developed a context-aware mobile recommendation system, named What'sUp, to support serendipitous discovery of city events and leisure activities. We built a web ontology in OWL 2.0, for representing cultural events and leisure activities in a city in order to promote a semantic web standard. Our event data is periodically extracted from Lecool e-magazine and Myguide web sites. In order to extract events from these sources, we rely on Web scraping and natural language processing. Our preliminary evaluation suggests that What'sUp can be useful for users that want to discover what's happening "here and now" fostering spontaneity.

The expression evaluation is a complex process that returns very satisfying results, as the recognizing process is effective and relies on simple rules. The performance of the ontology population is satisfying and every individuals and properties are filled in as we expected.

In the near future, we plan to expand the information extraction to other sites with city cultural and leisure offers. We can also gather more information about locations and categorize them on the ontology (museum, garden, theatre, casino and so on). We should also enrich our application capabilities with the introduction of a user profile concept, as this feature is very important nowadays. It would allow the user to like an event, share it with friends and to have preferences that would restrict the results display. The query translation approach can also be improved, by adding new keywords and expression variations (abbreviation, orthography errors admittance and context-related expressions processing). We will conduct field study with groups of users to evaluate the effectiveness of What'sUp in supporting serendipitous discovery.

References

1. Tintarev, N., Flores, A., and Amatrain, X.: Off the beaten track - a mobile field study exploring the long tail of tourist recommendations. In Proceedings of the 12th International Symposium on Human-Computer Interaction with Mobile Devices and Services (MobileHCI) (2010)
2. Kruger, A., Baus, J., Heckmann, D., Kruppa, M., and Wasinger, R.: Adaptive mobile guides. In The Adaptive Web, Brusilovsky, P., Kobsa, A. and Nejdl, W.(eds.), vol. 4321. , pages 521 – 549, Springer (2007)
3. Zheng, V.W., Zheng, Y., Xie, X., and Yang, Q.: Collaborative location and activity recommendations with GPS history data. In Proceedings of the 19th international conference on World wide web, pages 1029–1038, ACM (2010)
4. Bellotti, V., Begole, B., H. Chi, E. et al.: Activity-based serendipitous recommendations with the Magitti mobile leisure guide, In Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08, pages 1157-1166
5. Forsblom ,A.,Liikkanen, L., Nurmi, Aman, P.: Out of the Bubble - Serendipitous Event Recommendations at an Urban Culture Festival, IUI'12, ACM (2012)
6. Knublauch, H.: Ontology-Driven Software Development in the Context of the Semantic Web: An Example Scenario with Protégé/OWL,1st International Workshop on the Model-Driven Semantic Web (MDSW2004)
7. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley Systems Programming Series, Boston, MA, 1984
8. Noël, L., Carloni, O., Moreau, N. et Weiser, S.: Designing a knowledge-based tourism information system. In Int. J. of Digital Culture and Electronic Tourisme, Special Issue on National Tourism Organisations and Exploitation of Information Technologies, Inderscience Publishers Ltd.(2008)
9. Weiser, S., Laublet, P. et Minel,J.-L.: Automatic identification of temporal information in tourism web pages. In E. L. R. A. (ELRA) (Ed.), Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), Marrakech, Morocco
10. Cao, T.-D., Phan, T.-H., Nguyen, A.-D.:An ontology based approach to data representation and information search in Smart Tourist Guide System. In 2011 Third international Conference on Knowledge and Systems Engineering, pages 171-175
11. Cardoso, J.: Developing an OWL ontology for e-Tourism. Chapter 4, Springer (2006)