SMOP: A Semantic Web and Service Driven Information Gathering Environment for Mobile Platforms

Özgür Gümüs[1], Geylani Kardas[2], Oguz Dikenelli[1], Riza Cenk Erdur[1], and Ata Önal[1]

[1] Ege University, Department of Computer Engineering, Bornova, 35100 Izmir, Turkey
{ozgur.gumus, oguz.dikenelli, cenk.erdur, ata.onal}@ege.edu.tr
[2] Ege University, International Computer Institute, Bornova, 35100 Izmir, Turkey
geylani.kardas@ege.edu.tr

Abstract. In this paper, we introduce a mobile services environment, namely SMOP, in which semantic web based service capability matching and location-aware information gathering are both used to develop mobile applications. Domain independency and support on semantic matching in mobile service capabilities are the innovative features of the proposed environment. Built-in semantic matching engine of the environment provides the addition of new service domain ontologies which is critical in terms of system extensibility. Therefore the environment is generic in terms of developing various mobile applications and provides most relevant services for mobile users by applying semantic capability matching in service lookups. GPS (Global Positioning System) and map service utilization cause to find near services in addition to capability relevancy. The software architecture and system extensibility support of the environment are discussed in the paper. The real life implementation of the environment for the estate domain is also given as a case study in the evaluation section of the paper.

1 Introduction

Location aware and personal interest based information gathering from mobile devices is very active application and research area. Different experimental applications with such capabilities have been introduced in the literature [1] [2] [3]. From our perspective, one of the most critical problems of such applications is the extensibility of the software architecture. In our context, extensibility of the architecture describes domain independency: the addition of the new service domains to the running mobile application.

In this paper, we will introduce a software environment to develop location aware and semantic web based mobile information gathering applications. From now on, we will call this environment as SMOP (A Semantic Web and Service Driven Information Gathering Environment for Mobile Platforms). The innovative feature of SMOP is the use of a semantic matching engine which can identify semantically related knowledge, based on user queries. This semantic matching engine supports the addition of new service domain ontologies and this capability is critical in terms of domain independency. Moreover, service oriented infrastructure of SMOP further contributes to the extensibility, since it is possible to introduce new services without affecting the core of the architecture.

The paper is organized as follows: Section 2 introduces the system overview and architecture of the environment. System extensibility from domain independency perspective is discussed in section 3. Section 4 gives an example case study and evaluates the environment while a new domain is added to the system. Section 5 gives an overview of related works in the literature and compares SMOP with those works. Conclusion and future work are given in section 6.

2 System Overview

2.1 General Architecture

SMOP has a three tiered architecture. Mobile client, the server side and platform web services exist in the corresponding tiers. The components of each tier and interactions between those components are shown in Fig. 1. Each tier will be discussed in more detail in the following subsections.

2.2 Mobile Client

The client side is responsible for getting the GPS data, providing the interface for specifying the user requests and displaying the results, sending the requests to the server in XML format and parsing the results received in XML format.

The “GPS Data Parser” component is responsible for parsing the location data retrieved from the internal or external GPS receiver. To get the most current position of the client/user, it reads periodically raw GPS data, parses it and gets the Latitude and Longitude coordinates of the client/user.

Dynamic creation of visual interfaces at run-time is the responsibility of the “User Interface Generator” component. User screens are usually limited in mobile devices; hence, the user interfaces created are not so complex. The primary user interface window provides view of all data received along with user-selectable menu choices for controlling the application. When the mobile client connects to the server at first time, the domain names which are added to the platform up to that time are retrieved and shown to the user by the “User Interface Generator”. Then, user selects a domain and the XML files containing the concepts belonging to selected domain’s ontology are transferred from the server. After that, the transferred XML files are parsed and a visual interface is created to let the user specify his/her choices. Hence, a user interface where users can specify their choices is created independently at run-time for each different domain by the “User Interface Generator”. In fact, the ontologies are represented in Web Ontology Language (OWL) in the knowledgebase of the Semantic Matching Service (SMS). However, since mobile devices are resource limited, we simplified and represented these ontologies in simple XML format to make the parsing process efficient in the mobile device. Otherwise, the mobile device should execute the code necessary to parse OWL documents. When the results of semantic match query are returned, the "User Interface Generator" lists the found domain instances and shows detailed information about these instances. User may want to see one of those instances on the map. In this case, the "User Interface Generator" shows the map which is provided by the Map Service (MS) on the screen.

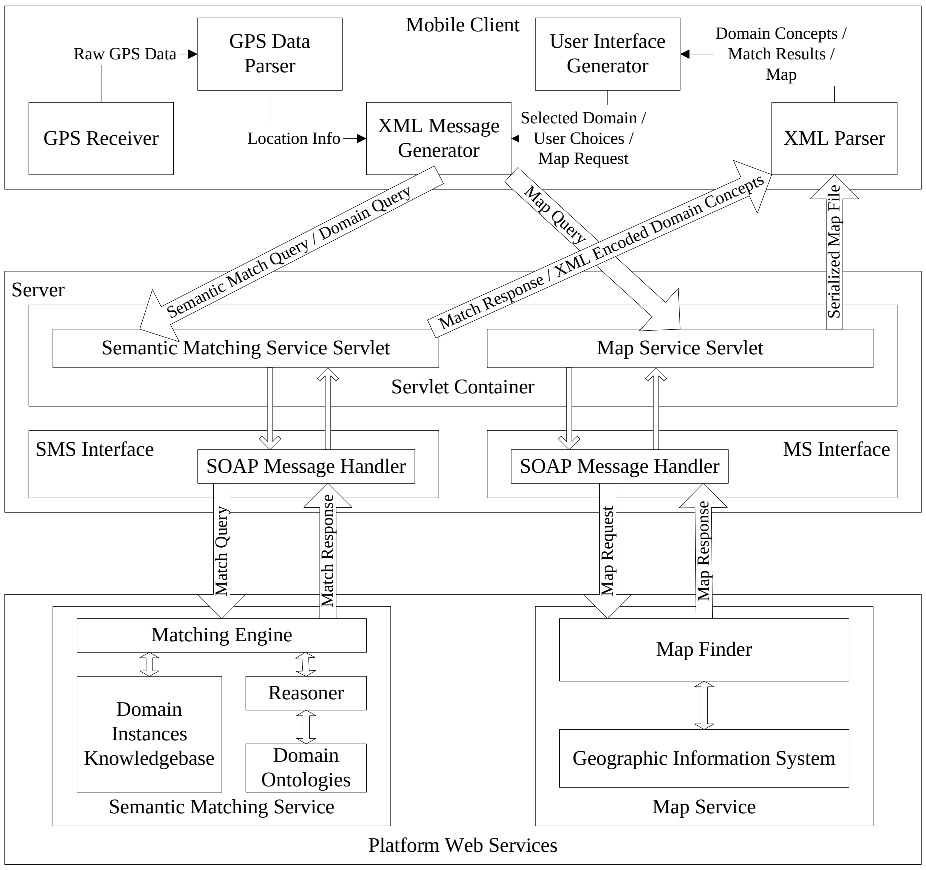


Fig. 1. SMOP's three tiered architecture

We preferred the requests and results to be transmitted in XML format, since it is a well-known web standard. So, the "XML Message Generator" component of the mobile client converts the (1) request of selected domain concepts, (2) request of semantic match according to user choices and (3) request of a map into the XML format. It also inserts the GPS location data of the client/user into last two requests.

Then, it sends the requests in XML format to the server using a GPRS Http network connection.

The “XML Parser” component parses the XML response document received from the server. So, depending on the given request, this XML document may include (1) concepts of a selected domain or (2) domain instances satisfying user choices in the form of a collection or (3) a serialized map showing one of found instances. Mobile devices have a limited memory. For this reason, the “XML Parser” component has been designed to be small and light. The pull parser technique, in which the software drives the parsing, has been used. In this technique, only some part of a XML document is read at once; hence, it does not need a large memory size. The application drives the parser through the document by repeatedly requesting the next piece. Our mobile client can process and display information as it is parsed after being downloaded from the server. In this case the “XML Parser” component basically iterates over the XML tree and finds the items. The parsed data is then passed to the “User Interface Generator” component to be printed on the screen of the mobile device.

2.3 The Server

The server side has the components that are responsible for meeting the client requests, fulfilling these requests via web services and returning the results to the client: servlet components interact with the client, interface components interact with web services and corresponding servlet and interface components interact with each other.

The “SMS Servlet” component takes the XML encoded match request, decomposes it and prepares the inputs of the SMS. These inputs are the domain concept (including its some properties that are chosen by the user) to be discovered and the required “degree of match” value. They are sent to the “SMS Interface” component. This component invokes the SMS and sends the outputs to the corresponding servlet. These outputs are the discovered domain instances that are matched with request sorted by the “degree of match” values. The “SMS Servlet” takes these instances, re-sorts the ones which have equivalent “degree of match” value according to distance to the client/user using the GPS location data and finally it inserts them into a collection. It then converts this collection into an XML message and sends the formed XML message to the client as an Http response.

The “MS Servlet” and “MS Interface” components work similar to the corresponding SMS components explained above. But, the inputs of the MS are the GPS location data of the mobile client/user and the instance that will be shown on the map. The “MS Servlet” takes the returned map from the “MS Interface” and serializes it into an XML message and sends it to the client.

2.4 Platform Services

2.4.1 Semantic Matching Service

The basic idea behind the matching process is to find the advertised concepts that are identical to the requested one. However, the advertised and requested concepts can be semantically related with each other but are not directly identical. In this case, a semantic matching process is required. Semantic matching process is a matching process that can identify the semantic relationships between the advertised and requested concepts. SMS executes this process. It has a registry to keep records of knowledge about advertised domain instances. It can be searched for the semantically most suitable instances using specific domain concepts.

The "Domain Instances Knowledgebase" component stores the instances of all domains. In this study, we have defined an abstract concept named as *Domain.* To add a new domain to the platform, a new concept that is specific type of *Domain* is defined. This new domain concept has its own data type properties and object type properties. Instances of this concept are created using different predefined domain ontologies for the object type properties and stored in the knowledgebase.

The "Matching Engine" component realizes matching of requested domain concept with advertised domain instances and produces the list of suitable instances sorted by "degree of match" values. It uses the "Reasoner" component to determine subsumption relation between ontological concepts. It uses an algorithm similar to one that is especially for discovery of semantic web services proposed in Paolucci et al's study [6] and it has explained in detail in our previous work [7].

2.4.2 Map Service

MS provides a satellite map enclosing specified locations in an area. The "Geographic Information System" (GIS) component is a software package named as Mapxtreme Java Edition of Mapinfo Corporation. It stores the information about geographic objects on the earth including their attributes, positions and shapes. It provides an API named MapJ to form a map image and make some operations and analysis on this image. The "Map Finder" component interacts with the GIS using this API. It sends the position of objects that must be enclosed in the map. GIS forms a map image on which the given objects (in this case the client and selected domain instance) are marked, and returns it to the "Map Finder" in binary format.

3 System Extensibility from Domain Independency Perspective

SMOP provides a mobile services environment in which capabilities of both semantic web and location-aware information gathering are utilized to develop mobile applications for various business domains. Considering system extensibility aspect, domain independency support is an important feature and should be provided within the platform via software reusability. In this section, domain independency support in SMOP is discussed.

When service capability matching is required for a new domain, it is enough to initialize knowledgebase of the internal semantic service matching engine of the platform with this new domain ontology without any need of software architecture modification. Only domain concept and related semantic service capability advertisement ontologies are needed to be added into the knowledgebase of the engine. Communication between the engine and the outer environment is realized over standard OWL messages: Match requests and responses are composed of RDF (Resource Description Framework) triples. Hence, change in domain only effects the communication content, neither structure nor software architecture. More about the internal execution and capability matching algorithm of our matching engine are beyond the scope of this paper. However, they have been discussed in [7] and [8].

Taking into consideration of Model-View-Controller system pattern [9] decomposition of the SMOP's architecture; it can be said that domain ontology and related knowledgebase represent *model*, mobile GUI components residing on cell phones represent *view* and finally servlet container and related web services stand for the *controller* layer within the system. We applied an abstract domain model into the controller layer of the SMOP to support various business domains without any code modification (Fig. 2).

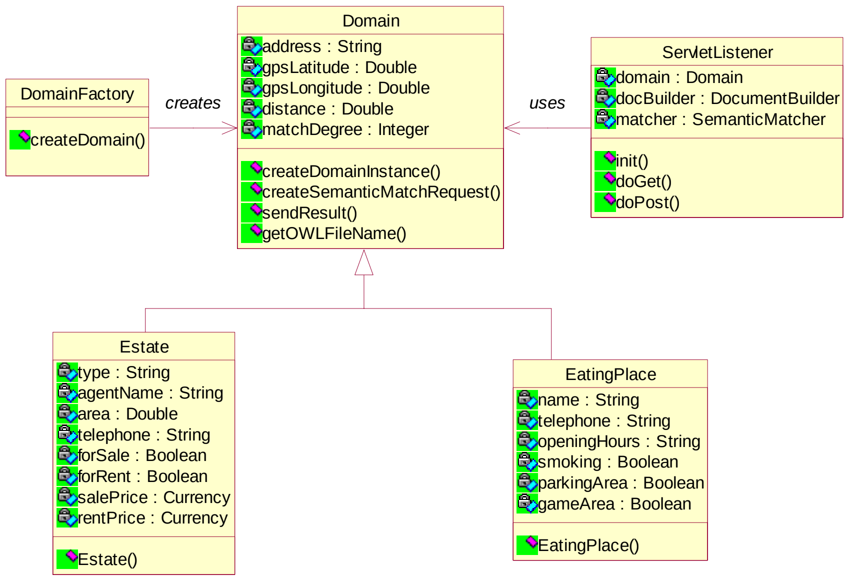


Fig. 2. Domain model of the SMOP to support extensibility in domain perspective

Domain is an abstract class which is extended by domain dependent wrapper classes to utilize the SMOP for new domains. For example, in Fig. 2, two subclasses of the *Domain* are given: *Estate* and *EatingPlace*. *Estate* is used for a "Real Estate Discovery System" in which mobile clients may search for geographically near real estates those match with clients' preferences – semantically appropriate with client's needs. On the other hand, *EatingPlace* instances represent eating places (like restaurants, cafes, patisseries, etc.) within an "Eating Place Discovery and Reservation" system in which semantic matching of eating place services with mobile clients' eating preferences is realized.

Apparently, it is enough to write such domain-dependent wrapper classes (subclasses of `Domain`) when a new domain is needed to be added into SMOP-based environment. Three abstract methods of *Domain*, called *createSemanticMatchRequest*, *createDomainInstance* and *sendResult* should be implemented within wrappers to handle domain knowledge.

createSemanticMatchRequest method receives related domain instance and domain type as input and returns its semantic match request counterpart which is processed by the engine during semantic match process. For example, eating place preferences of a mobile client are encapsulated within an *EatingPlace* object in a system. By calling this object's *createSemanticMatchRequest* method, controller servlet retrieves those preferences in RDF triples to form service match request and hence, they are compared with the advertised service capabilities by the built-in semantic matching engine.

Implementation of the *createDomainInstance* provides reverse information flow inside the system. As it is discussed in the architecture section, SMOP's semantic service capability matching engine –called Semantic Matching Engine- returns semantic match results in a collection of *SemanticMatchResult* objects. Those objects store matched service advertisements as OWL individuals with their match degrees and GPS data. Those ontological result properties should be converted into the domain-specific attributes of the wrapper class. So, they can be processed by controller servlets and transferred into the mobile clients. This match result – domain class conversion is realized by calling *createDomainInstance* method of the related domain object.

sendResult method implementation provides transmission of domain instance content into mobile clients in XML format. Controller servlet sends those XML representations of the semantic query results to the view components of SMOP residing on the mobile phone. Those software components also don't need to be re-written in case of a domain change. Because, as it is mentioned in the previous section, "User Interface Generator" module of the mobile software initially retrieves concepts belonging to a specific domain's ontology from controller servlet also as XML data –they are not hard coded in GUI components- and it dynamically creates the visual interface. During semantic query communication, SMOP's mobile GUI components only need to parse received XML data and print out the content into the phone's screen in appropriate to the domain's desired format.

On the other hand, the constructor method of the wrapper class should be implemented in a way that attributes are set with the values which are retrieved from request XML document. This request document is received from the mobile client.

Runtime employment of wrapper classes is realized by applying *Factory* creational design pattern [10]. Initially, *DomainFactory* processes a document in which various domain definitions are specified and it creates desired domain-related class instance to be used within the controller servlet during system interactions. Notice that application of the Factory pattern and `Domain` class abstraction in software design provide the representation and use of the above mentioned domain-related wrapper classes in the system in a completely domain independent way; because wrappers are always referenced over their superclass (*Domain*) inside the software.

4 Case Study and Evaluation

As a case study, a mobile services application for the real estate domain has been designed and deployed on SMOP. In order to realize such an application environment, we have first defined a concept named as “Estate” to advertise the places for sale and/or rent to the internal matching engine of the SMS. One of the properties of this concept is the “type” which takes value from an OWL ontology shown in Fig. 3. This property is used during semantic matching process to find out semantically related real estates with user’s request. The other critical properties are “for sale” and “for rent” to define a real estate is for sale, for rent or both. These properties are the additional options that users can specify in addition to real estate type. “GPS latitude” and “GPS longitude” properties define the geographic position of the advertised real estate. The other properties are address, area, sale price, rent price, agent name and telephone.

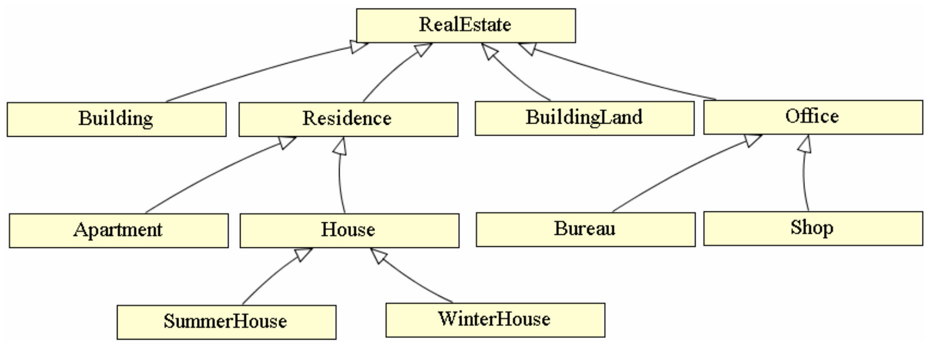


Fig. 3. An example *Real Estate Type* ontology to show the taxonomy of instances in a domain

We mapped a simplified version of *Real Estate Type* ontology to XML format and stored in the server. We also prepare an XML document representing of “for sale” and “for rent” properties of “Estate” concept and possible values of these properties. So, whenever the real estate domain is selected, these XML files are transferred to the mobile device for the creation of the visual interfaces at run-time. The XML document corresponding to the *Real Estate Type* ontology given in Fig. 3, is shown below:

```
<level1 type="Real Estate">
   <level2 type="Residence">
      <level3 type="House">
         <level4 type="Winter House"/>
```

```
        <level4 type="Summer House"/>
      </level3>
      <level3 type="Apartment"/>
    </level2>
    <level2 type="Building Land"/>
    <level2 type="Building"/>
    <level2 type="Office">
      <level3 type="Bureau"/>
      <level3 type="Shop"/>
    </level2>
</level1>
```

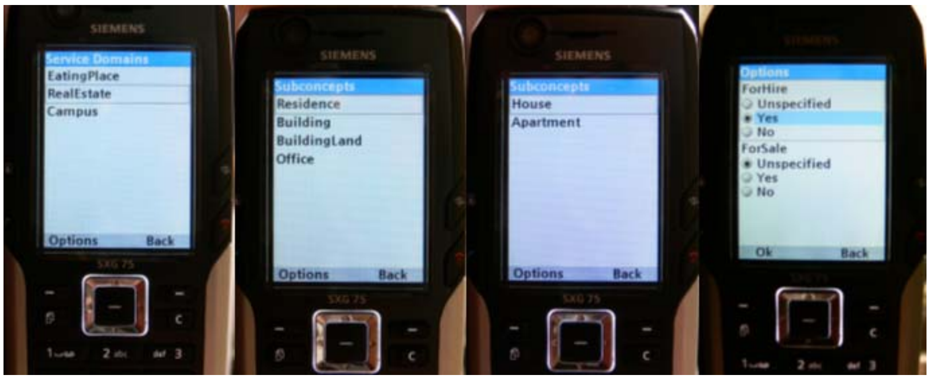
For a test scenario, we have created six instances of real estate concept and advertised to the SMS. Some important properties of these instances are shown in Table 1. Notice that, "type" property takes value from predefined *Real Estate Type* ontology.

Table 1. Instances of "Estate" concept that are advertised to SMS

No	Agent Name	Type	For Sale	For Rent
1	Green House	*Building*	Yes	Yes
2	Homecity	*SummerHouse*	No	Yes
3	House&House	*House*	Yes	No
4	RE/MAXX	*WinterHouse*	No	Yes
5	RE/MAXX	*Office*	No	Yes
6	RE/MAXX	*House*	Yes	Yes

Mobile client components of the application software have been deployed on a Siemens SXG75 cell phone with built-in GPS receiver and Java 2 Micro Edition (J2ME) 1.1, Mobile Information Device Profile (MIDP) 2.0 and Connected Limited Device Configuration (CLDC) 1.1 support.

When the client application on mobile phone is started by the user, first of all, it connects to the server and gets the available service domains. Let us assume that the user selected the real estate domain, then following the taxonomy of real estate types within this domain, he/she selected *House* concept to find instances of this concept. And he/she also specifies that he/she seeks real estates for rent. Screen snapshots showing user's selection of his/her request are given in Fig. 4.



Options
ForHire
○ Unspecified
● Yes
○ No
ForSale
● Unspecified
○ Yes
○ No

Fig. 4. Screen snapshots showing user's selection of his/her request

After user completes his/her request, the *House* concept, additional choices about the requested real estate and the position of the user are sent to the server. So, corresponding request XML document is shown below:

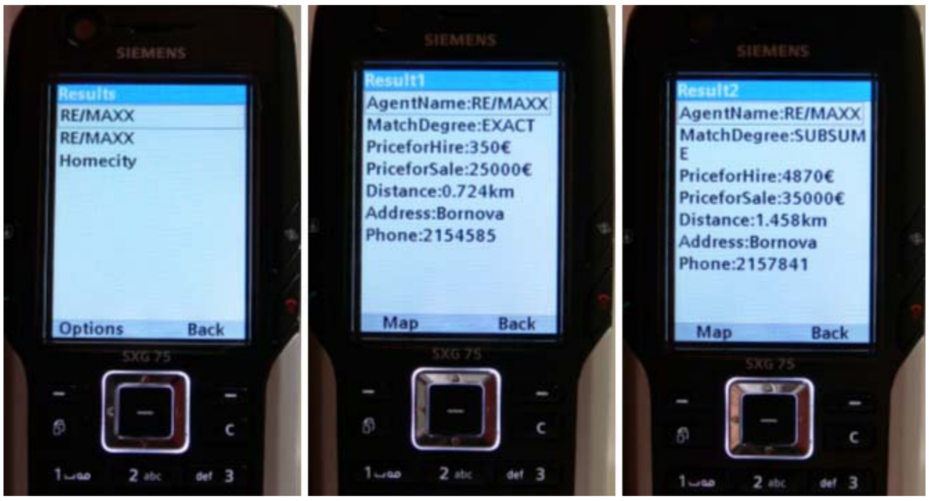
```
<request>
   <type>House</type>
   <forRent>Yes</forRent>
   <forSale>Unspecified</forSale>
   <gpsData>
      <latitude>22.333E</latitude>
      <longitude>52.444N</longitude>
   </gpsData>
</request>
```

Default value of "degree of match" parameter is given as *subsumes*. So, the server invokes the SMS with the given request. The SMS performs semantic matching in the following way: it excludes the first and fifth instances because their "degree of match" values are *fail*. It also excludes the third one because it isn't for rent. The sixth one has the *exact* "degree of match" value and second and fourth instances have the *subsumes* "degree of match" values. So, the SMS returns the matched real estate instances with their "degree of match" values in the following order: sixth, second, and fourth.

After the server receives match results from the SMS, it performs another sort operation on semantically equal match results regarding their distances to the user. Although the second and fourth ones have the same "degree of match" values, the fourth one is closer than the second to the user. Hence, final list contains match results in the following order: sixth, fourth, second. Finally, the server creates an XML document that includes resultant domain instances and sends it to the mobile client. As an example, a part of the result XML document is shown below:

```
<matchResults>
   <result>
      <type>House</type>
      <degreeOfMatch>EXACT</degreeOfMatch>
      <distanceToClient>0,724km</distanceToClient>
      <agentName>RE/MAXX</agentName>
      <address>Bornova Street 1</address>
      <tel>2154585</tel>
      <forRent>Yes</forRent>
      <forSale>Yes</forSale>
      <area>200</area>
      <rentPrice>350€</rentPrice>
      <salePrice>25000€</salePrice>
      <gpsData>
         <latitude>27.229E</latitude>
         <longitude>38.455N</longitude>
      </gpsData>
   </result>
   <result>
      ...
   </result>
   ...
</matchResults>
```

The mobile device parses result XML document and creates a visual interface to show returned real estates to the user. The user can select one of the real estates from the list of match results to see detailed information about it. Snapshots showing list of match results and details of first two matched real estates are given in Fig. 5.


Results
RE/MAXX
RE/MAXX
Homecity
Options Back

Result1
AgentName:RE/MAXX
MatchDegree:EXACT
PriceforHire:350€
PriceforSale:25000€
Distance:0.724km
Address:Bornova
Phone:2154585
Map Back

Result2
AgentName:RE/MAXX
MatchDegree:SUBSUM
E
PriceforHire:4870€
PriceforSale:35000€
Distance:1.458km
Address:Bornova
Phone:2157841
Map Back

Fig. 5. Snapshots showing list of match results and details of first two matched real estates

The user may choose to see a satellite map which shows one of these locations' and his/her geographic position. In this case, the GPS data of both user and the real estate are sent to the server in XML format. As an example, corresponding XML document for the first result is shown below:

```
<locations>
   <user>
      <gpsData>
         <latitude>22.333E</latitude>
         <longitude>52.444N</longitude>
      </gpsData>
   </user>
   <domain instance>
```

```
        <gpsData>
           <latitude>27.229E</latitude>
           <longitude>38.455N</longitude>
        </gpsData>
     </domain instance>
  </locations>
```

The server takes this XML document and invokes the MS. The MS marks the user and the real estate with different colors (blue for the user and red for the real estate) on a satellite map using its GIS. Then, this map is received by the mobile client through the server in binary format. Screen snapshot showing a satellite map enclosing first one of the matched real estates and the user is given in Fig. 6.

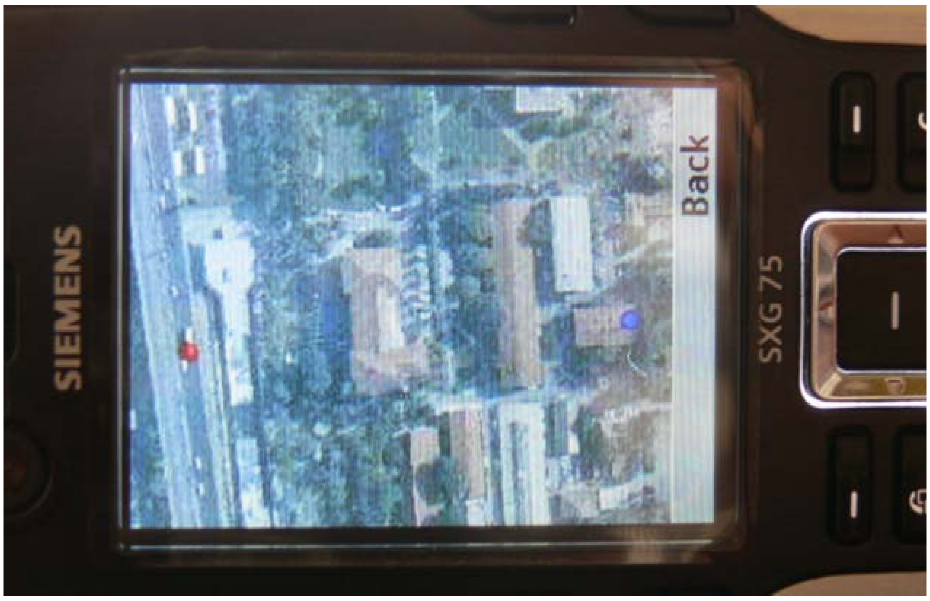


Fig. 6. Screen snapshot showing a satellite map enclosing first one of the matched real estates and the user. The blue point on the map marks current position of the mobile user while red point represents position of the real estate.

To demonstrate domain independency feature of SMOP, we have added a new domain, called eating place, to the above application environment. We didn't need to change mobile client and platform web services tiers as we expected. On the other hand, we have naturally designed concept ontology of this new domain, added corresponding domain individuals into the SMS knowledgebase and prepared XML documents representing concepts of the domain ontology. Notice that, the wrapper class stands for this new domain has also been implemented by extending *Domain* abstract class as discussed in section 3. Hence the mobile client could interact with the SMS by means of this wrapper. Upon completion of above preparations, we examined that the environment successfully provides semantic web based information gathering on both domains for mobile clients.

5 Related Work

In the pervasive computing literature there are studies to develop software environments for location aware and context aware application development. Below, we will summarize some of these studies by comparing them with our system so that we can show in what ways SMOP is different from them.

Hessling et al. [1] devised an application, where the semantic user profiles which are stored in the mobile devices are matched against the semantic services broadcasted by stations. When the users with their mobile devices enter the range of a station, the station's services are matched against the semantic user profiles. Although the algorithm of how the user profiles are matched semantically is not given in detail in their paper, from the semantic matching point of view, the work of Hessling et al. [1] can be considered as the nearest one to our study. As we mentioned above, our aim is not to discover services, but to search the predefined semantic knowledge using semantic matching techniques so that mobile users can get the most relevant results for their queries. In addition, we focus on the extensibility of the architecture which is not considered in work of Hessling et al.

The Agents2Go [2] is an agent based distributed system that allows the creation of location dependent and service-based information systems. Although the proposed agent based architecture may allow the generation of new location dependent information systems, the extensibility perspective is not clear and not discussed in the paper. Also, a semantic matching engine, which makes it possible to extend the architecture by adding new service ontologies at run time, is not considered in Agents2Go.

Intelligent Computing group in University College Dublin developed some location dependent and context aware mobile applications like Gulliver's Genie [3] based on their Agent Factory framework. Their focus is in application development in space limited mobile devices and they have proposed an approach called as collaborative agent tuning [4] to incrementally develop such applications. Although Agent Factory framework and ACCESS context aware infrastructure [5] provide the necessary software architecture to implement agent based location dependent applications, extensibility in terms of adding new service domains at run time is not the focus of their works.

On the other hand, there are many classical location-based information search services in mobile environments commercialized by GSM operators. For example, there are systems where users with mobile phones can be directed to the nearest local restaurants, shops, etc. These systems can be considered as standard information search services for mobile users. There are two features, which make SMOP different from them. The first feature is being domain independent based on an extensible software architecture. Supporting semantic matching is the second feature where the system that we have developed differs from them. Using semantic matching, a result list ranked by the degree of semantic match can be presented to the user in response to his/her request so that he/she can have the option of accessing to the most semantically related information. So, we take the previous works one step further by integrating semantic matching capability into the information gathering process in pervasive environments and by modeling the system in a way that it supports domain independency.

6 Conclusion and Future Work

We have introduced a mobile services environment in which semantic web based service capability matching and location-aware information gathering are used to develop mobile applications. The proposed architecture has been fully implemented and tested for different service domains.

The environment is adaptive for various mobile applications due to its domain independency and provides most relevant services for mobile users by applying semantic capability matching in service lookups. GPS and map service utilization cause to find near services in addition to capability relevancy and hence we believe this increases quality of the mobile services.

We currently work on to integrate mobile service execution into the environment. The system in use provides an enhanced service information gathering. However, invocation of remote services - except semantic service discovery and map services - by the mobile clients is not currently supported. Our aim is to provide an ultimate mobile services system in which semantic service discovery and execution are both fulfilled. In such a system, for example, a mobile client may first choose a relevant restaurant service and then reserve a table at this restaurant by only using his/her cell phone; or considering the real estate system given in this paper, the user may also arrange a meeting with the related estate agent after determination of the semantically most suitable and nearby estate.

Acknowledgements

We would like to thank members of Mobile Software Development Group in Ege University Computer Engineering Department (Seymen Ersen Diraman, Mehmet Niziplioglu, Oguz Karakus, Serkan Kaba, Murat Colak and Salim Asan) for their great effort in this study.

This study is partially funded by Ege University Scientific Research Projects Directorate with the project number 2003MUH039.

References

1. Hessling, A., Kleemann, T., Sinner, A.: Semantic User Profiles and their Applications in a Mobile Environment, In the Proc. of Artificial Intelligence in Mobile Systems 2004 (AIMS'04) In conjunction with UbiComp 2004, Nottingham, UK (2004)
2. Ratsiomor, O., Korolev, V., Joshi, A., Finin, T.: Agents2Go: An Infrastructure for Location-Dependent Service Discovery in the Mobile Electronic Commerce Environment, In ACM Mobile Commerce Workshop 2001, available at: http://research.ebiquity.org/v2.1/papers.

3. O'Grady, M. J., O'Hare, G. M. P., Sas, C.: Mobile agents for mobile tourists: a user evaluation of Gulliver's Genie, Interacting with Computers 17(4): 343-366 (2005)
4. Muldoon, C., O'Hare, G. M. P., O'Grady, M. J.: Collaborative Agent Tuning, ESAW'05, Kusadasi, Turkey (2005)
5. Muldoon, C., O'Hare, G. M. P., Phelan, D., Strahan, R., Collier, R. W.: ACCESS: An Agent Architecture for Ubiquitous Service Delivery, CIA 2003, pp. 1-15 (2003)
6. Paolucci, M., Kawamura, T., Payne, T. R., Sycara, K.: Semantic Matching of Web Services Capabilities, In the proc. of the first international semantic web conference (ISWC), Sardinia, Italy (2002)
7. Erdur, R. C., Dikeneli, O., Önal, A., Gümüs, Ö., Kardas, G., Bayrak, Ö., Tetik, Y. E.: "A Pervasive Environment for Location-Aware and Semantic Matching Based Information Gathering", Computer and Information Sciences - ISCIS 2005, Lecture Notes in Computer Science, Springer-Verlag, Vol. 3733, pp. 352-361 (2005)
8. Kardas, G., Gümüs, Ö., Dikeneli, O.: "Applying Semantic Capability Matching into Directory Service Structures of Multi Agent Systems", Computer and Information Sciences - ISCIS 2005, Lecture Notes in Computer Science, Springer-Verlag, Vol. 3733, pp. 452-461 (2005)
9. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad P., Stal, M.: "Pattern-Oriented Software Architecture, Volume 1: A System of Patterns", John Wiley & Son Ltd, New York USA (1996)
10. Gamma, E., Helm, R., Johnson R., Vlissides, J.: "Design Patterns - Elements of Reusable Object-Oriented Software", Addison-Wesley, Massachusetts USA (1994)