A Framework for Decentralized, Context-Aware Mobile Applications Using Semantic Web technology

William Van Woensel, Sven Casteleyn, Olga De Troyer

Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel, Belgium
{William.Van.Woensel, Sven.Casteleyn, Olga.DeTroyer}@vub.ac.be

Abstract. The recent evolution in mobile devices, combined with rapid advancements in identification techniques, has lead to new opportunities for mobile application developers: mobile applications that can be made aware of their environment and the objects in it. Additionally, by combining mobile devices and identification technology with the Web, mobile applications can be developed that exploit services and information associated with nearby objects. In this paper, we present an application framework that supports the development of such mobile applications, without having to rely on a central service to provide information on the user's environment. Furthermore, by deploying Semantic Web technology, the integration of information from various information sources is facilitated, allowing for expressive and powerful personalized information delivery.

Keywords: Mobile Web, development framework, context-awareness, mobility, personalization

1 Introduction

In the current mobile Web, users typically use their mobile device (e.g., smart phone, PDA, portable game console) to access the Web using a dedicated mobile browser (e.g., Skyfire, Opera Mini). Although this makes the Web accessible anywhere and anytime, the limitations of mobile devices (e.g., small screen, limited input capabilities, processing power and bandwidth) still hinder the widespread mobile use of the Web. Furthermore, in a mobile setting (e.g., driving, walking, sightseeing), users are often unable or reluctant to spend large amounts of time browsing and locating the information or services they need at that particular moment and place.

Nevertheless, modern mobile devices often have features that enable the tailoring of information and service delivery. For instance, GPS-enabled mobile devices offer the possibility to determine the user's location, and current identification technologies (e.g., RFID technology) allow a mobile device to detect entities and other (mobile) users in the user's environment. Furthermore, by combining these capabilities with some basic knowledge on the mobile user (i.e., background, preferences, etc.), it is possible to offer a fully personalized mobile experience.

In this article, we present the mobile application development framework SCOUT (Semantic COntext-aware Ubiquitous scouT) that supports the development of

context-aware mobile applications, which offer relevant information and services depending on the mobile user's environment and particular needs at a given time and place. In contrast to most existing approaches (see related work), SCOUT is a scalable, decentralized and distributed solution, where no single centralized server is required and where each identifiable entity is responsible to provide and manage its own data and services in the form of a Web presence [2-5]. These can range from simple Websites/Web services to online sources providing structured information (e.g., RDF files).

Due to its open, decentralized and distributed nature (together with its ubiquitous availability), the Web is the ideal platform for deploying these presences, as it fits the desired properties of our framework perfectly. Furthermore, it allows re-use of the wealth of descriptive information already available online (e.g., existing Web pages, RDF information such as FOAF profiles) as Web presences. By employing Semantic Web standards and vocabularies to describe Web presences in a uniform and expressive way, the SCOUT framework allows seamless integration and querying of data from (several) different entities, thereby providing mobile applications with a richer and more complete view on the global environment.

In this article, we illustrate the framework and its usage by means of a real-world scenario, consisting of a mobile user exploring the authors' university campus using his/her mobile device. Here, we will focus only on the delivery of information. The remainder of this article is structured as follows. In the next section, we provide a global overview of the proposed SCOUT framework, and elaborate on each of the layers using our university campus scenario. In section 3, we present related work, and section 4 states conclusions.

2 SCOUT: An overview

The SCOUT framework uses a layered architecture, which clearly separates the different design concerns and thus assures independence between layers and from underlying technologies. Fig. 1 shows an overview of the architecture; each layer is explained in more detail in the subsequent subsections.

2.1 Detection Layer

The Detection Layer is responsible for detecting identifiable physical entities in the vicinity of the user, and subsequently obtaining the reference to the corresponding Web presence. The Detection Layer thus contains components that encapsulate different detection techniques (e.g., RFID, NFC, Bluetooth, etc), which can extract references to Web presences for use by other layers. By encapsulating these components in a separate layer and by having them implement a uniform interface, the framework can transparently switch from one technique to another (or using several of them in parallel), depending on what detection techniques are available and which are supported by nearby entities.

Consider our university campus scenario, where several points of interest and test subjects on the campus have been tagged with an RFID tag, which contains a reference to their corresponding Web presence (i.e., a URL). RFID detection is an example of a "direct" detection technique: the reference to the Web presence is directly obtained from the detected entity. In contrast, "indirect" detection techniques are used when the reference to the Web presence cannot be directly obtained, and a third party service needs to be accessed to obtain the Web presence. For instance, a remote location service can be deployed for a specific region, providing a set of URL-position bindings in a certain area around the user (respectively denoting the Web presence reference and the physical entity's coordinates). SCOUT supports the aforementioned techniques, RFID and third party services, by means of dedicated components in the Detection Layer.

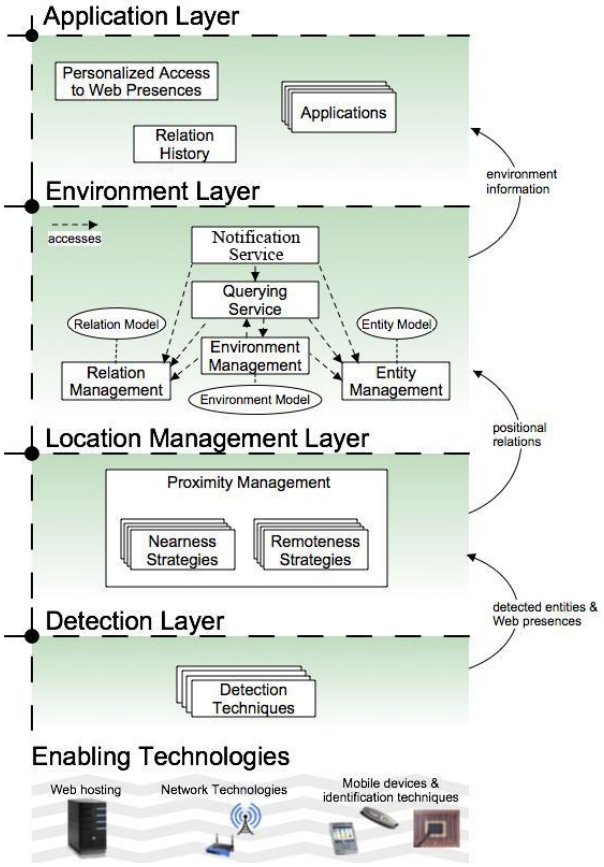

Fig. 1. SCOUT architecture layers.

The direct detection technique has been implemented by a component that wraps an RFID reader; for the indirect detection technique, we have provided a component that

frequently queries the location service for nearby Web presences, as well as the location service itself (i.e., a Web server application that serves URL-location bindings).

2.2 Location Management Layer

The responsibility of the Location Management Layer is to determine which entities are nearby (and no longer nearby) the user, based on information obtained from the Detection Layer. A "positional" relation is created when a detected entity is determined to be *nearby*. Once the relation is created, the Location Management Layer is responsible for maintenance of the relation; more specifically, to invalidate the relation when the related entity is no longer nearby. By providing positional information in this conceptual way, applications can abstract from the details of dealing with raw positional information.

In order to determine when an entity is nearby or no longer nearby, the SCOUT framework deploys so-called nearness and remoteness strategies, respectively. Collectively, we call them proximity strategies. We now refer back to our university campus scenario where a mobile application, using the Detection Layer, is now able to determine which entities are in the user's vicinity. Our mobile user walks past a physical entity, e.g., a reproduction of Rodin's The Thinker statue on the university campus. As The Thinker statue is tagged with an RFID tag, it can be detected by a mobile device supporting RFID reading. Because the mobile device in our setup features a short-range RFID reader, nearness of the artifact can be directly inferred, as the detection range of our reader is smaller than the nearness distance. However, this also means that remoteness cannot be directly inferred (i.e., the fact that the entity cannot be detected does not necessarily mean it is no longer within the nearness distance). Therefore, we must deploy a remoteness strategy where the entities' absolute positions are compared (namely the absolute position of the user and the entity's position) and the positional relation in question is invalidated in case the difference exceeds the nearness distance. For non-stationary entities, this remoteness strategy is extended to allow for regular position updates of the detected physical entity. This scenario illustrates that the choice of remoteness strategy can be decoupled from the choice of nearness strategy; in this case, this allows us to cope with the limited range of our RFID reader. In the current version of our framework, we provide support for a variety of nearness and remoteness strategies.

2.3 Environment Layer

The Environment Layer stores and integrates data about the entity and its current environment, and provides services to obtain information from nearby Web presences in both a push-and pull-based manner. As mentioned before, our framework leverages Semantic Web technologies to more easily and effectively combine information from different Web presences; additionally, the reasoning capabilities of the Semantic Web

standards can be exploited to infer additional data.[1] In the following subsections, we discuss the components from this layer in more detail.

2.3.1 Relation and Entity Models

These models contain the basic information needed for personalization. The Relation Model provides fundamental environmental information by storing the positional relations the entity is / has been involved in, along with references to the Web presence of the related entities. This information is provided by notifications of the Location Management Layer upon creation and invalidation of positional relations. The Relation Management component (see Fig. 1) provides a view on these relations, allowing both querying of and programmatic access to this information. Aside from currently active positional relations, this model also keeps all past relations the entity was involved in, along with creation and invalidation time stamps.

The Entity Model is responsible for storing metadata on the entity. In most cases, the entity will be a person, in which case this model is commonly called the User Model (e.g., [16]) and stores certain preferences and characteristics of the user. Several existing ontologies can be reused to represent a person's metadata: CC/PP[2] to represent preferences and mobile device capabilities; FOAF[3] or vCard[4] for representing a person's profile, DCMI[5] to describe documents and items, etc. The Entity Management component (see Fig. 1) allows applications to pose queries over the Entity Model, and also provides the necessary API's to programmatically access this information.

2.3.2 Query Service

The core component of the Environment Layer is the Query Service, which allows client applications to query the models of Web presences of (nearby) entities. These queries can range from simple queries retrieving metadata to queries containing complex semantic conditions. These semantic conditions may also reference the user's Entity Model, thus effectively personalizing information retrieval. Using the Web presence references provided by the Relation Management component, an application can use the Query Service to obtain (personalized) information on a nearby entity.

Referring back to our campus scenario, we deploy an application that enables a mobile user to request relevant information on a nearby physical entity (e.g., The Thinker artifact). After obtaining the Web presence reference from the Relation Model, the application poses a query to the Web presence (consisting of an RDF file) using the Query Service. The query below requests general information

[1] The latter is however outside the scope of this article.
[2] http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430/
[3] http://www.foaf-project.org
[4] http://www.w3.org/TR/vcard-rdf
[5] http://dublincore.org

(dcmi:description) on the entity, personalized to fit the mobile user's education level (dcmi:educationLevel) and language (dcmi:language):

```
SELECT ?descr
WHERE  {
         ?user rdf:type em:User ;
         ?entity rdf:type scout:WebPresence ;
            dmci:description ?descr .
         ?descr dcmi:language ?language ;
            dcmi:educationLevel ?level .
         ?user dcmi:language ?language .
            dcmi:educationLevel ?level .
       }
```

This query references both the user's Entity Model and the Web presence of the nearby entity; in order to execute this query, the Query Service thus needs to integrate the data from user's Entity Model, and the data from the remote Web presence. The em:User type is used to refer to the mobile user, while the scout:WebPresence type allows the query to reference the remote Web presence. In general, the "scout" namespace is used to describe resources and properties that are generated by the framework. The DCMI ontology is used here to represent the user's language and education level.

The above approach is clearly pull-based, as the client application queries information from the environment on the user's request. However, many applications exhibit behavior that needs to be executed when certain changes occur in the environment, i.e., when certain entities become (or are no longer) nearby. If the pull-based approach is used in this case, the application developer is left with the burden of implementing a constant polling of the Relation Model for new positional relations, in order to cope with the mobile user's volatile environment. As this is neither practical nor efficient, a Notification Service is provided that allows for selective, push-based information gathering.

2.3.3 Notification Service

The Notification Service allows applications to obtain references of nearby entities in a push-based manner, thus allowing them to become responsive to changes in the mobile user's environment. More specifically, an application can register itself with this service in order to automatically receive events when nearby entities are encountered, or are no longer nearby. By default, the application is simply notified of all nearby entities; additionally, the framework allows filtering by enabling the application to specify a condition (in the form of a SPARQL query) which must be satisfied by the nearby entity before the application is notified. As was the case with the Query Service, this filtering condition may also utilize information from the user's own Entity Model, thus allowing for personalized filtering.

In our campus scenario, we now deploy an application that automatically displays information on certain interesting entities as they become nearby. For this purpose, the application uses the Notification Service, and provides a condition specifying that for university students, only buildings (campus:Building) are to be displayed where

the student takes courses (em:followsCourse). The following query represents this condition:

```
SELECT ?entity
WHERE  {
         ?user rdf:type em:User ;
            em:followsCourse ?course .
          ?entity rdf:type scout:eventEntity ;
            rdf:type campus:Building ;
            campus:contains ?room .
         ?room campus:givenCourse ?course .
       }
```

The above query references both the user's Entity Model (using the "em" namespace) and the Web presence of the nearby entity (using the "campus" namespace). The "campus" namespace is used to convey information on objects present in the VUB campus. The scout:eventEntity type allows a query to reference the entity that caused the event, in this case the entity that became nearby. The following line of code registers the application to receive events only in case the above query returns results, i.e., if the filtering condition is satisfied:

```
service.registerForEvents(this, EventTypes.ENTITY_NEARBY, query,
                          INTERESTING_BUILDINGS);
```

Note that the last argument in the method invocation represents an identifier for this specific registration (more specifically, an integer). The identifier is communicated back to the application upon notification, so that it knows for which registration it receives a given event.

2.3.4 Environment Management

Until now, we have illustrated how queries, which integrate the mobile user's metadata (stored in the Entity Model) and the metadata of the nearby Web presence, can provide for personalized retrieval of information. However, the full strength of the SCOUT framework lies in combining metadata from multiple Web presences, obtained from both past and current positional relations, and integrating it with metadata of the mobile user. We call this integrated view on the environment the Environment Model, and it effectively allows the application developer to extensively query any piece of the user's context and environment.

In order to illustrate the benefits of this model, let us refer back to our campus scenario, where we further extend our application by referring to more than one entity in our personalization condition. The following query retrieves shops on campus and their sold items (campus:sells) in case they are related (campus:relatedTo) to either an interest of the user (em:hasInterest) or a building that the user spent a certain amount of time nearby (scout:wasNearby):

```
SELECT ?entity ?item
WHERE  {
         ?user rdf:type em:User ;
         {
             ?user scout:wasNearby ?prevEntity .
             ?prevEntity scout:nearbyFrom ?from ;
                 scout:nearbyUntil ?till .
             FILTER (?till - ?from > 900)
             ?prevEntity campus:relatedTo ?concept .
         }
         UNION
         {
             ?user em:hasInterest ?interest .
             ?interest campus:relatedTo ?concept .
         }
         ?entity rdf:type scout:eventEntity ;
             campus:sells ?item .
         ?item campus:relatedTo ?concept .
       }
```

In this query, information from the Entity Model (i.e., interests), Relation Model (i.e., nearby entities) and metadata of other Web presences (i.e., shops and sold items) is referenced. The benefit for the application developers thus lies in the fact that they can access this distributed information by posing a single query to the Environment Model. It should be noted that the push-based approach described in section 5.3 can also benefit from the Environment Model; for instance, we could pass the above query as an argument to the `registerForEvents` method of the Notification Service. As a result, our application will be notified in case a shop selling interesting items (defined by the user's either explicit or implicit interests) is encountered.

3 Related work

As discussed in the introduction, our approach is based on the central notion of linking physical entities to their associated information, a concept first introduced by the HP Cooltown project [2-5]. Others have also since realized the potential of connecting physical objects to related online information. Touchatag[6] from Alcatal-Lucent Ventures is a commercial initiative that uses RFID technology to connect objects to online applications. In [12], an open lookup framework is presented where objects tagged with RFID tags are linked to resource descriptions, allowing users to retrieve information on specific tagged objects.

An important characteristic of our solution is its decentralized nature, which manifests itself in two ways: decentralized data storage and query resolving. In contrast, many existing approaches that focus on the location-specific retrieval of information (e.g., [9-11]) employ a centralized Information System (IS), which stores and maintains all location-specific information. Clearly, this requires significant effort to setup and keep up-to-date; moreover, such an IS mostly requires a pre-defined location model of the area (e.g., symbolic, geometric, etc [11]), linking absolute positions/identifiers/etc to the physical entity data, which is not very flexible or well-

[6] http://www.touchatag.com

suited for dealing with non-stationary entities (e.g., moving art exhibitions, persons, etc). In our approach, Web presences are put online and maintained by content-providers themselves, thus also increasing control over their own data and significantly reducing the threshold for participation.

Other approaches [6-8] provide a central service that acts as an integrated view over a set of distributed (context) data sources, and thus represents a centralized way to query resolving. Although it is clear that such an approach offloads a lot of work from the mobile devices themselves [8], it is also less scalable and flexible, as every single query needs to pass through this service and each data source needs to be registered with it. We also believe that the possibilities of mobile devices will keep increasing as it has been for the last decade, thus reducing the need for "thin" clients. An approach that also focuses on the decentralized, context-specific retrieval of data is Context-ADDICT [15]; however, this system only supports schema-based filtering and only very limited data-based filtering of information, and does not provide an event-based way of obtaining information from the environment (see below).

Another feature that sets us apart from such centralized approaches ([6-8]) is our built-in support for providing location-specific information, allowing applications to get the information when and where they need it. More specifically, we separate the logic necessary to infer the "nearness" of an entity into separate layers; consequently, different technologies and strategies can be used interchangeably. Additionally, most centralized approaches are not event-based, and do not alert applications when a certain a contextual configuration occurs. Two other approaches also provide event-driven notification, namely GeoNotes [13] and Sentient Graffiti [14]. However, both are centralized solutions; furthermore, GeoNotes is not an application framework but rather a single application, and Sentient Graffiti does not focus on integration of different data sources.

4 Conclusions

In this paper, we have presented the SCOUT framework, which allows for the development of personalized, context-aware mobile applications in a ubiquitous setting. The framework architecture is based on the separation of concerns, where each design concern is dealt with in a different layer. As elaborated in the paper, the framework supports a variety of different detection techniques and proximity strategies, which can be used transparently and interchangeably. The most important part of the framework is the Environment Layer, which (among other things) provides applications with an integrated view on the user's physical environment, and allows the application developer to seamlessly combine and query information from different Web presences. By allowing applications to pose arbitrarily complex queries over the Environment Model, and by taking into account mobile user's context and particularities, the SCOUT framework supports true context-aware, personalized mobile application development.

5 References

1. Debaty, P., Caswell, D.: Uniform Web Presence Architecture for People, Places, and Things. In: Personal Communications, Issue 4, Volume 8, pp. 46--51, IEEE (2001).
2. Debaty. P., Goddi, P., Vorbau, A.: Integrating the Physical World with the Web to Enable Context-Enhanced Services. Technical report, Hewlett-Packard (2003).
3. Kindberg, T., Barton, J.: A Web-Based Nomadic Computing System. Computer Networks, vol 35, no. 4, pp. 443--456, Elsevier (2001)
4. Barton, J., Kindberg, T.: The Cooltown User Experience. In: CHI 2001 Workshop: Building the Ubiquitous Computing User Experience
5. Cappiello, C., Comuzzi, M., Mussi, E., Pernici, B.: Context Management for Adaptive Information Systems. Electronic Notes in Theoretical Computer Science 146, 69--84 (2006).
6. Xue, W., Pung, H., Palmes, P.P., Gu, T.:Schema matching for context-aware computing. In: 10th international conference on Ubiquitous computing, pp. 292--301, ACM, Seoul, Korea (2008).
7. Judd, G., Steenkiste, P.: Providing Contextual Information to Pervasive Computing Applications. In: 1st IEEE International Conference on Pervasive Computing and Communications, pp. 133--142, IEEE Computer Society, Fort Worth, Texas, USA (2003).
8. Tummala, H., Jones, J.: 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 (2005).
9. López-de-Ipiña, D., Vazquez, J.I., Abaitua, J.: A Context-aware Mobile Mash-up Platform For Ubiquitous Web. In: 3rd IET International Conference on Intelligent Environments, pp. 116--123, IEEE, Ulm, Germany (2007).
10. Challiol, C., Rossi, G., Gordillo, S., De Cristófolo, V.: Designing and Implementing Physical Hypermedia applications. In: ICCSA 2006, UWSI 2006, pp. 148--157, Springer Berlin / Heidelberg (2006).
11. Roduner, C., Langheinrich, M.: Publishing and Discovering Information and Services for Tagged Products. In: 19th International Conference on Advanced Information Systems Engineering, pp.501--515, Springer Berlin / Heidelberg, Trondheim, Norway (2007).
12. Espinoza, F., Persson, P., Sandin, A., Nyström, H., Cacciatore, E., Bylund, M: GeoNotes: Social and Navigational Aspects of Location-Based Information Systems. In: Ubicomp 2001: Ubiquitous Computing, pp 2--17 (2001)
13. López-de-Ipiña, D., Vazquez, J.I., Abaitua, J.: A Context-aware Mobile Mash-up Platform For Ubiquitous Web. In: 3rd IET International Conference on Intelligent Environments, pp. 116--123 (2007)
14. Bolchini, C., Curino, C., Schreiber, F. A., Tanca, L.: Context Integration for Mobile Data Tailoring. In: 14th Italian Symposium on Advanced Database Systems, pp. 48-55, Portonovo (Ancona), Italy (2006).
15. Brusilovsky, P.: Adaptive hypermedia. User Modeling and User-Adapted Interaction 11(1-2), 87--110 (2001).
16. Alain Barrat, A., Cattuto, C., Colizza, V., Pinton, J.f., Van den Broeck, W., Vespignani, A.: High resolution dynamical mapping of social interactions with active RFID. In: CoRR abs/0811.4170 (2008).