Realizing an Open Ubiquitous Environment in a RESTful Way

Yong Liu, Kay Connelly
Indiana University
{yolliu, connelly}@indiana.edu

Abstract

Web services have been widely used to support context-aware applications for context information retrieval in a ubiquitous environment. However, most of the existing research efforts in this field only focus on using SOAP web service as an enabling technology. In this paper, we compare REST and SOAP web services for supporting ubiquitous environments. We describe our approaches to deploy an open ubiquitous computing environment using REST style services along with semantic web and mobile technologies. We also discuss an example context-aware application developed for the environment and show how REST style services can contribute in moving ubiquitous computing technologies into the real world.

1. Introduction

In an open ubiquitous environment, a large quantity of heterogeneous services and context-aware applications dynamically join and leave the network. The applications act on behalf of their owners to request certain services. The users, though engaging many devices and applications during their ordinary activities, may not necessarily be aware of this. This is the vision of ubiquitous computing, widely regarded as the next paradigm of computing and a post-desktop model of human computer interaction.

For a ubiquitous environment to function as a whole, some standard protocols and languages must be adopted for the devices and software agents in the environment to communicate. Web services have been proved to be able to well support integration across heterogeneous platforms in a loosely coupled manner [31]. In recent years, many research efforts in the ubiquitous computing community focused on applying web service technology to enable open or limited spatial ubiquitous environments [13][24]. Certain service discovery and invocation mechanisms, such as UDDI/OWL-S registry [20], were developed for context-aware applications to discover and request services available in the environment.

However, most of the existing studies in this field proposed using SOAP [25] web services as the enabling service technology for ubiquitous environments. In this paper, we examine another web service style, the Representational State Transfer (REST) [11], and discuss how it can be used to support an open ubiquitous environment. We demonstrate through a proof-of-concept context-aware application that combined with semantic web technologies and existing web infrastructure, RESTful web services can well support automatic context information discovery and consumption in an open ubiquitous environment.

The rest of this paper is structured as follows. In section 2, we briefly introduce the supporting technologies to implement an open ubiquitous environment. In section 3, we compare REST and SOAP web service as the enabling technology for context information provisioning and consumption. In section 4, we discuss our strategies to deploy a ubiquitous environment using RESTful services. In section 5, we describe the supporting platform of the ubiquitous environment and an example context-aware application. We discuss several issues on developing RESTful ubiquitous services in section 6 and conclude in section 7.

2. Background

Today, the WWW has become the largest body of human knowledge and information related services on the planet. Standard protocols and formats enable information to flow across various devices and domains. Web information retrieval technology works as a third party organizer in this highly decentralized system and makes information discovery possible to a large degree. Given the facts that the WWW has established a solid infrastructure and already been an important part of our life, it is not only possible, but also convenient to exploit the WWW as to support service discovery in an open ubiquitous environment.

Search engines are key to the success of the WWW. A significant feature of search engine centered web information organization is that information publishers do not have to register or publish their resources to a single or limited number of directory services, which makes a large scale heterogeneous and dynamic information system possible. However, existing general-purpose search engines cannot directly solve the service discovery issue in an open ubiquitous environment since the current web information retrieval is largely based on syntax, not semantics of the indexed content. Commercial general-
purpose search engines actually leverage end users to finish the most important part of the information retrieval process. This imposes a heavy cognitive load on users, which is contradictory to the key objective of ubiquitous computing.

Semantic web, as a W3C initiative to specify machine-understandable meanings for the information and services on the web, comprises a variety of enabling technologies. Among them, Resource Description Framework (RDF), RDF Schema (RDFS) and Web Ontology Language (OWL) provide a mechanism to generate formal descriptions of concepts and relationships within a given knowledge domain on the web. Semantic web service [15] is the combination of semantic web and web service technologies. OWL-S [19] is an OWL based web service ontology which supplies web service providers with a core set of constructs for describing the properties and capabilities of their web services. Web Service Modeling Ontology (WSMO) [32] is another similar semantic web service initiative. By using the standard and domain specific ontologies, web services can have machine-understandable annotations just as the static semantic web documents. Users and software agents can hence automatically discover, compose and invoke these semantic web services.

Mobile phone is another enabling technology for an open ubiquitous environment given the large number of phone users and the device’s ubiquitous nature and increasing capability. According to the U.S. Geological Survey data, mobile phone sales are projected to exceed 1 billion units per year in 2009 [30]. By March 2008, Apple has sold over 4 million iPhones, capturing 28% of U.S. smart phone market [17]. There is an obvious trend that mobile phones are becoming more powerful and smarter. Given the current processing and communication power, software support and battery life, many modern smart phones can work as a full-featured web server to support real world web applications [18]. In a ubiquitous environment, mobile phones can play three different roles simultaneously, which are sensor platform, service requester, and service provider. In many previous researches such as ContextPhone [21], Place-Its [28], and [4][5][26], mobile phones are used as sensor platforms or service requesters. As mobile phones become increasingly powerful, we will see mobile phones act as the service providers in more ubiquitous systems.

3. REST vs. SOAP web services for ubiquitous environments

SOAP web service has been widely used in a variety of IT environments to integrate heterogeneous applications. However, it is not the only Service-Oriented Architecture (SOA) design pattern that ubiquitous application developers can choose. We argue it is even not the most appropriate one, compared to the REST style web service, for an open ubiquitous environment.

Representation State Transfer is a style of software architecture first introduced by Roy Fielding in 2000 [11]. Initially it refers to a set of network architectural principles on how resources are defined and addressed to achieve scalability. Now it is often used to describe any simple interface that transmits domain-specific data over HTTP without any additional message layer such as SOAP [23]. Many popular websites including Yahoo! [33], eBay [8] and Amazon [1] provide REST services as programmable interfaces to their resources.

An important concept in REST is the existence of resources, each of which can be referred to using a resource identifier (URI). In the REST architectural style, all resources are accessed through a generic interface, which results in a dramatic decrease in the complexity of the service interface semantics during the service interaction [7]. We believe REST style web service is a better enabling technology for establishing SOA in an open ubiquitous environment based on the following two reasons.

First, different from the activity-oriented SOAP style web services, REST web services, by definition, are resource-oriented. In the REST architecture, one can identify and access resources by a URI, and the operations which can be performed against the resources, including retrieving (GET), modifying (POST), creating (PUT), and deleting (DELETE) resources, are defined by the HTTP specification. In this case, the design of the service interface allows the service requester to move the information around and manipulate it according to their needs [27].

Similar to REST style web services, ubiquitous environments also have a resource-oriented nature. Here the resources are the context information. In such an environment, context-aware applications sense context information and make use of it to accomplish certain tasks for their owners. Most of the context sources are atomic. Each of them publishes only a certain type of context information, such as location, temperature, noise level, etc. These resources can be easily wrapped with REST style web services and accessed by various context-aware applications through a small predefined set of operations. Since service providers usually do not know the intention of service requesters, they just simply leave it up to the requesters to figure out how to deal with the information. In activity-oriented SOA styles, such as SOAP web services, since the resources are implied in activities, a large number of operations need to be defined and understood by service requesters to access the resources. This imposes extra burden and unnecessary constraints on context-aware applications.
Second, compared to SOAP web services, REST services are usually regarded as lightweight and easy to build. The REST style improves server scalability by reducing the need to maintain communication state. It also requires less client-side software support to consume the services [23]. On the contrary, the communication overhead and complex invocation procedure of SOAP web service make it a less attractive supporting technology for service provisioning and consuming in a ubiquitous environment considering the relatively constrained processing and communication power of mobile devices and sensors.

4. Strategies on deploying a RESTful open ubiquitous environment

In the upcoming ubiquitous computing era, several key technology challenges facing an open ubiquitous environment are:

- The large quantity of heterogeneous and dynamic services in the environment requires a decentralized service organizing approach;
- Applications should be able to discover and consume a service by the semantics of its capability description on demand;
- The service provisioning mechanism adopted should be lightweight and with minimal communication overhead;
- As the major computing platform, mobile phones play three important roles (sensor platform, service consumer, and service provider) simultaneously;

With the help of semantic web technologies and existing WWW infrastructure, REST style web services can be applied to manage context sources and enable context information provisioning in a ubiquitous environment. In this section we discuss several strategies to realize a RESTful open ubiquitous environment which can answer the above challenges. These strategies include a web based component development and deployment approach, using ontologies for knowledge sharing by context-aware applications, leveraging general-purpose search engines for ubiquitous service discovery, and using mobile phone for service provisioning.

The WWW is the most complete and accessible collection of electronic information in existence. For ubiquitous application developers who need to consume information in a wide area environment, it makes sense to build upon the WWW’s existing infrastructure. There are two types of services in a ubiquitous environment. The first one consists of the services that only publish context information to the environment. In this case the service consumers need to receive information. The second one consists of services that publish information and act according to a consumer’s requests. An example in this category is the ticket booking service. In this case, the service provider acts upon the request of the consumer, modifying its resources by purchasing a ticket on behalf of the consumer. Both types of ubiquitous services can be implemented using REST style web services.

In our open ubiquitous environment, all components are implemented using existing web technologies, i.e. HTTP, XHTML, PHP etc., except for where a native component was necessary for collecting a certain type of context information. This design reconciles mobile and web programming and encourages the web developer community to engage in context-aware application development by leveraging their web programming experiences and existing toolkits. Another advantage of this design is that once a component or an application is written, it can run on mobile phones, desktops or servers without modification as long as a certain web server is running on that device. This leads to a low deployment and maintenance cost.

All of the REST style services and their represented resources in the environment are exposed to the WWW and able to be accessed by external web clients, even browsers. This provides extra ways for users to access the context information published by devices in the open ubiquitous environment. Users from other domains can still find useful context information through browsers on their laptops or desktops. This greatly extends the scope of the ubiquitous environment and creates impact to traditional web users.

There is existing research on designing shared ontologies for ubiquitous applications, such as SOUPA [3] and GAS Ontology [6]. Ontologies are formal representation of a certain knowledge domain and building blocks of the semantic web. In terms of functionality, ontologies are key to the scalability of ubiquitous systems, and play an important role in deploying ubiquitous systems outside of a lab setting. However, in an open ubiquitous environment, a service annotated with certain domain ontologies can still be useless, unless it can be discovered by its capability semantics on demand.

As W3C standards, RDF, RDFS and OWL can be used to generate machine-understandable semantics for ubiquitous services. Some technologies, such as SPARQL [29], enable basic query and reasoning over RDF/OWL service descriptions. With the help of a central service repository such as a UDDI/OWL-S registry [20] or a service description RDF triple store, these technologies make service discovery by semantics possible. However, for an open ubiquitous environment, there is no central repository that can hold all the service descriptions. This is analogous to the situation that it is impossible to maintain a central registry for all web pages on the WWW. Since REST web services have made an open ubiquitous environment part of the WWW, we can
leverage general-purpose web search engines for ubiquitous service discovery.

There is an emerging interest in both academia and industry in deploying the semantic web. Many standards and tools have been developed to reconcile the semantic web with the current web, helping the WWW to gradually evolve into an information body understandable to both humans and machines. These efforts include Microformats [16], RDFa [22], eRDF [9], GRDDL [14] etc. In our research, we annotate service representations, which are human-readable XHTML pages, with semantics in Microformats or RDFa formats. Hence service representations are both understandable to the context-aware applications and indexable by general-purpose search engines whose target audiences are human users.

For search engines to discover ubiquitous services by semantics, some additional information needs to be provided along with the service representations left on the web. Swangler is a research project aiming to enable semantic web documents discovery over search engines by applying a swangling technique on the documents to enrich them with additional RDF triples [12]. A hash function is used to convert the service fingerprint into tokens acceptable to search engines. We adopt a similar approach to make semantic annotations on service representations.

In our open ubiquitous environment, service representations are annotated with Microformats or RDFa tags. Then RDF statements are extracted from these XHTML pages and some key RDF triples are used to generate the service fingerprints through a hash function. These service fingerprints are then added back to the original service representations left on the Web. Once indexed by search engines, these fingerprints become the keywords used to query their attaching services.

In previous ubiquitous projects, mobile phones were used mostly as sensors and clients. This is partly due to the limited processing and communication power of earlier mobile phones. In recent years, more capable mobile phones are available on the market and take a large market share. Among those, Apple’s iPhone is a good example with 4 million shipments in 8 months since its first release. In our open ubiquitous environment, every mobile phone can run its own web service and publish its context information directly to the web. The mobile phones act as sensor platform, service consumer and service provider at the same time. Users can certainly disable any of these roles, but the enabling mechanism is ready.

5. OUCE & an example context-aware application

By applying the strategies discussed in the previous section, we developed OUCE, a loosely coupled REST style context service platform, to support open ubiquitous environments. In our development we chose the iPhone as the target mobile computing platform considering its large market share and representative features. However, except for the native components implemented with the iPhone SDK, all other components can be deployed on other mobile or non-mobile platforms with minor or no modifications required. We also developed an example context-aware application to demonstrate the platform’s capability. It is a typical ubiquitous computing application and has already been discussed in previous research projects [2][10].

5.1. OUCE overview

As shown in Figure 1, the OUCE platform consists of several standard components. The standard OUCE components include multiple mobile computing units and the supporting domain ontologies, SPARQL endpoints, RDF triple stores, and external search engines. This section describes each of these components.

![Diagram of OUCE platform components](image)

Figure 1. Standard components of the OUCE platform

5.1.1. Mobile computing unit. The mobile computing unit is the major component of the OUCE platform. This unit typically runs on top of mobile phones, but it can also run on other computing devices with proper low level system support. It serves as both context information provider and consumer in the open ubiquitous environment. Its functionality as a ubiquitous service provider largely depends on the types of sensors it controls or has access to.

The mobile computing unit on a mobile phone can be conceptually partitioned into five layers. The first layer is system hardware and sensors such as the microphone and cellular communication module. The second layer is system software and application databases. The third layer is the web server. An application server can also be deployed in this layer if certain technologies, such as Tomcat or Axis server for SOAP web services, are used to enable ubiquitous services. The forth layer is the...
ubiquitous service and application layer, which enables the mobile computing unit to act as both a service provider and a service consumer. And the fifth layer is the representation of ubiquitous services provisioned by the unit and the embedded service fingerprints.

In our OUCE implementation, native objective-c libraries are developed to collect context information through various sensors on the iPhone, including the Cell ID and signal strength for location triangulation purpose. The phone owner’s social context information is extracted directly from application databases on the phone without accessing any low level sensors. Information in this category includes the user’s contacts, scheduled activities, etc. The context information collected from both layers is available for upper layer ubiquitous services or applications to further manipulate.

All ubiquitous services and applications in a mobile computing unit are written in scripting languages such as PHP, and made available in the form of web applications. Ubiquitous services selectively publish the collected context information to the environment. Ubiquitous applications, on the other hand, act on the users’ behalf to discover and consume available services in the environment to accomplish certain tasks. These applications also make use of context information collected within the same unit in some scenarios.

To publish certain context information, a ubiquitous service first annotates the information with semantics in Microformats or RDFa formats. Figure 2 shows an example of a place review service representation annotated with context information.

As discussed earlier, a service representation directly annotated with compact URIs in RDF format is not discoverable by general-purpose search engines. An extra service fingerprint generation process is required in this case. In a mobile computing unit, an MD5 hash function is used to generate fingerprints for context services. Figure 3 shows the RDF triples extracted from the above geo-annotated review. Figure 4 shows the fingerprints generated for two geo information triples.

5.1.2. SPARQL endpoint and RDF triple store. OUCE context-aware applications leverage SPARQL endpoints to query the RDF statements returned from context services by semantics. In some scenarios, such as scheduling a meeting involving multi-parties, RDF statements from multiple services are used in the query process. These statements are stored in a RDF triple store to which the SPARQL endpoint has access.

In our OUCE implementation, a MySQL database server on a desktop serves as the RDF triple store. There are SPARQL endpoints on both the mobile phones and desktop servers for the application to select from based on the complexity of the queries and the number of RDF statements involved. The RDF triple store can also be deployed on mobile phones equipped with a database system, such as iPhone’s SQLite database.

5.1.3. Domain ontologies. Domain ontologies are central for knowledge sharing among ubiquitous services and applications running on the OUCE platform. The ontologies used by OUCE components can be any existing standard ontologies or ontologies developed specifically...
for the platform. SOUPA [3] is an example of the former, and the GeoReview ontology discussed in the next section is an example of a dedicated OUCE ontology. By “dedicated”, we mean these ontologies were initially developed for OUCE applications. However, they can certainly be reused by any semantic web applications inside or outside of the platform.

The domain ontologies used by OUCE applications can be hosted anywhere on the web, not necessarily on mobile or non-mobile devices in the OUCE domain. There is no central authority to publish and maintain these ontologies. Any OUCE user and developer can write their own ontologies and publish them to the web in the form of RDF or OWL documents.

5.1.4. General-purpose search engines. Yahoo! is the default search engine in the current OUCE platform implementation. This is simply because it provides a full-featured REST style search interface, which is lightweight and conformed with the OUCE internal service style. Other search engines, such as Google, can be used as long as a proper wrapper is deployed for applications to invoke the new search API and consume the results.

Once a service representation is published on the web with incoming links from external pages already indexed by Yahoo!, it can be crawled and indexed in days or weeks depending on the importance of the incoming links. A user can create such links from her personal websites, blogs, or even SNS pages. If a user does not want his services publicly accessible, he can use the “nofollow” tag on the incoming links, or simply remove all external links. It is also possible to deploy finer access control mechanisms in OUCE, but it is beyond the scope of this paper. One consideration is that complex access control mechanisms may affect how lightweight the OUCE services are, and should be designed carefully when they are necessary.

5.2. GeoReview

GeoReview is a context-aware application which senses the reviews attached to a physical place left by other GeoReview users. Similar applications were introduced in previous research, such as E-graffiti [2] and GeoNotes [10]. However, GeoReview is developed based on the OUCE platform, and hence possesses more flexibility and scalability than the previous closed context-aware systems can provide.

The GeoReview application runs on an iPhone platform equipped with a tailored Apache web server and PHP 5.2 environment. The iPhone leverages a dynamic DNS service from DynDNS.com to obtain a dedicated domain name which can be used to access the phone when it gets connected to the Internet through WLAN or EDGE. A native cell tower triangulation component is implemented in the OUCE platform to feed the phone’s current geo location information to upper level applications. It collects the Cell ID and signal strength data through the cellular communication module of the phone and queries the remote Google Map server to fetch the geo coordinates of each of the detected cell towers. Then a triangulation algorithm is applied to calculate the phone’s current location.

GeoReview allows the user to attach a review to any physical location in-situ by storing the review text along with the geo coordinates of the place on her phone. The message is then annotated with proper semantics and published through the web server running on the phone. Figure 5 shows a partial GeoReview ontology used to annotate the service.

```
<rdf:Description rdf:about="#Review">
  <rdf:type rdf:resource="#Review"/>
  <rdfs:comment>A review of a place</rdfs:comment>
  <rdfs:label>Review</rdfs:label>
  <rdfs:range rdf:resource="#Review"/>
  <rdfs:domain rdf:resource="#Review"/>
  <rdfs:comment>The text of the review</rdfs:comment>
  <rdfs:label>text</rdfs:label>
  <rdfs:range rdf:resource="#text"/>
  <rdfs:comment>The person that has written the review</rdfs:comment>
  <rdfs:label>reviewer</rdfs:label>
  <rdfs:range rdf:resource="#reviewer"/>
  <rdfs:comment>The longitude of the place being reviewed</rdfs:comment>
  <rdfs:domain rdf:resource="#reviewer"/>
  <rdfs:range rdf:resource="#hasLat"/>
  <rdfs:comment>The latitude of the place being reviewed</rdfs:comment>
  <rdfs:domain rdf:resource="#reviewer"/>
  <rdfs:range rdf:resource="#hasLong"/>
</rdf:Description>
```

Figure 5. A partial GeoReview ontology

Later, other users visiting the same place can read the reviews attached to the place using their GeoReview application. The application generates several service fingerprints from its current geo coordinates and the GeoReview ontology, and fires a query containing the fingerprints to the web search engine. After the matched service representations are returned, the application extracts the RDF statements and sends them to a RDF triple store. By submitting queries to a SPARQL endpoint connected to the triple store, the application retrieves related reviews and displays them on the phone screen. The service representation and corresponding fingerprints
are shown in Figure 2 and 4. Figure 6 shows an example SPARQL query used to retrieve the reviews.

![SPARQL Query](image)

**Figure 6. An example SPARQL query for review retrieval**

As a ubiquitous application running on the OUCE platform, GeoReview has two major advantages over the previous context-aware geo-notes systems. First, its service representations are not limited to a closed ubiquitous system, but understandable to any applications using the GeoReview ontology. Second, dedicated servers are not necessary for GeoReview applications. The applications can publish user reviews through either the built-in web server on the phone, or some generic proxy server deployed on the OUCE platform. Since a proxy server can host services for various types of ubiquitous applications without the need of customization, the operating efficiency of the ubiquitous environment as a whole is much higher.

### 6. Discussion

Although many existing real-world uses of REST style web services publish service representations in XML format, in OUCE we use XHTML documents for ubiquitous service representations. One major reason is the search engine friendliness of XHTML pages. Since OUCE applications leverage web search engines for service discovery, the service representations need to be easily indexed by the search engine and the expected services should be accurately returned once a query containing certain keywords is fired. Modern general-purpose search engines are largely designed to crawl and index HTML pages. They usually consider the contributions of certain HTML tags during their tokenization process, which can be exploited by ubiquitous service providers to emphasize certain capability semantics in the service representations.

Another reason to choose XHTML as the service representation format in OUCE is that it is understandable to both machine and human users. Semantically annotated with Microformats or RDFa attributes, the XHTML service representations can be used to generate RDF triples using GRDDL, and hence can be understood and reasoned about by the context-aware applications in OUCE. In the mean time, human users can browse these XHTML pages to find interesting information about the represented resources without further help of any ubiquitous applications. This helps an open ubiquitous environment to seamlessly integrate into the web and bring positive impact to people’s everyday lives immediately.

Current implementation of the OUCE platform focuses on the retrieving operation of RESTful ubiquitous services against context resources through HTTP GET requests. In a ubiquitous environment, a context-aware application only needs to fetch context information from external resources without modifying them in most cases. However, to make the OUCE platform a complete solution for ubiquitous service provisioning and consumption, certain mechanisms need to be provided to enable more operations of RESTful ubiquitous services, such as resource modification though an HTTP POST request. And more semantic annotations need to be added to the service representations to help context-aware applications automatically determine how service can be invoked and through which operations. We plan to tackle these problems in our future research.

### 7. Conclusions

In this paper we have presented an approach of applying RESTful services to support context information provisioning and consumption in an open ubiquitous environment. We discussed several advantages of deploying REST against SOAP web services to enable context-awareness in such an environment and the development strategies we followed. By introducing the OUCE platform and an example context-aware geo-review application, we demonstrated the feasibility of this approach and how it can be used to address real-world needs.

### 8. References


