MoReCon: A Mobile Restful Context-Aware Middleware

Erdogan Dogdu
Georgia State University (adjunct)
TOBB University of Economics and Technology
Ankara, Turkey
edogdu@etu.edu.tr

Onur Soyer
Core Trust, Inc., South Korea
onrsyr@gmail.com

ABSTRACT
Context-aware applications utilize the user’s context information, such as location, activity, user’s status, etc., and provide a better user experience by utilizing the appropriate information that suits the context. Context-aware applications are being widely used and developed as mobile device usage is increasing constantly worldwide. MoReCon is a mobile middleware we developed for providing context-information as a service from mobile devices. It is designed with the purpose of making the context-aware mobile application development easier. MoReCon provides an extensible RESTful web service that makes the context information management available to all kinds of applications including web and mobile applications.

Categories and Subject Descriptors
D.2.12 [Interoperability]: data mapping, distributed objects; H.3.5 [Online Information Services]: data sharing, web-based services.

General Terms
Design, Performance, Experimentation.

Keywords
Context-aware mobile applications, RESTful web services, middleware.

1. INTRODUCTION
Context-aware applications are not new, but they are gaining popularity as more mobile devices and applications are widely deployed all over the world. It is expected that the main internet access medium in near future will be mobile devices instead of desktop computers. Therefore, future mobile applications will be utilizing the context information extensively to better serve the users.

Mobile devices are always with the user and the new generation smart devices come with location-aware services such as GPS sensors as a standard. Therefore context data such as the location will be utilized more and more with new applications. Since mobile computing is still in its developing stages, standards are not yet established in platform architectures, application development environments, and services. Context-aware application development for mobile devices is usually a difficult process for application developers. Each mobile device has its own sensor-dependent context providing API. Developers mostly use ad-hoc programming techniques when accessing the context information provided by these different sensors and packages and then storing and updating the context information in relevant data stores. Besides the sensor dependent context data, user supplied context information such as status are also generated, saved and used by different techniques. Since the context-aware application development is usually done using ad-hoc techniques, maintaining such software is also difficult. Adding new context information and utilizing this information in the application requires modifying the whole application and redeveloping.

A better approach, which we propose in this paper, would be to standardize the modeling, usage and simply management of context data for mobile platforms and also abstracting it from the context-aware application by way putting context data management in a middleware.

We therefore present a new approach for developing context-aware applications. We designed a novel middleware called MoReCon, short for Mobile Restful Context-Aware Middleware that abstracts the management of context data and services from the context-aware application. This new middleware provides a universal API for accessing context data using web services from any device and application. New context data type and values can be added via middleware without affecting the context-aware application and without interrupting context-aware services and applications.

In the next section we present the related work. In section 3, we show how MoReCon is used and present the generic design of MoReCon as a layered software. In section 4, the current prototype implementation and its service oriented design details are given. RESTful API of MoReCon is presented in section 5. MoReCon extensibility features are presented in section 6 and implementation features are given in section 7. Test results presented and discussed in section 8 and we conclude in section 9.

2. PREVIOUS WORK
Context-aware applications and environments is an active study area in both academia and industry. Earlier works are mainly concerned with providing a useful solution utilizing the context information using ad-hoc techniques and propriety packages. Therefore, these solutions are not interoperable, and usually require heavy programming efforts to bind to every new application framework in the area of context-aware applications. This was due to the fact that there are many mobile platforms on many different mobile device hardware choices. Next generation solutions need to be developed in common platforms and subject to certain standards, so that interoperability can be realized easily without much effort in programming. Below we present some of the closely related work on service-oriented and mobile context-aware work.

Keith presents a review of current architecture choices for building context-aware services [1]. He points to the availability of service-oriented solutions for service discovery, implementation and management as a natural next step in context-aware mobile application development.

Service-oriented architecture is mainly an enterprise architecture choice for distributed applications involving application servers, enterprise service bus (ESB) management, messaging in XML, etc.
But, the mobile computing on the other hand has the issue of limited device capabilities in terms of memory size, computing power, and low-bandwidth in connection speeds. Therefore, standard web services solutions do not fit in the mobile computing environments.

RESTful web services is a newer solution for light-weight web service implementation. It is based on resource-oriented access where RESTful web services are basically access paths to web resources [1]. Utilizing RESTful web services in mobile applications is also a new approach. Christensen discusses utilization of RESTful web services in cloud computing for next generation mobile applications [3]. But he limits the utilization of RESTful web services to accessing the services from mobile applications in client mode only.

Yong Liu et al proposes a RESTful web services based context aware application environment for mobile application development [4]. But they work on the propriety iPhone platform and therefore stop short at providing an interoperable and application independent context-aware solution. Besides they do not discuss the details of OUCE platform they implemented from the point of how the context information is represented and accessed.

An earlier platform that is developed to provide context information from mobile devices is ContextPhone [6]. ContextPhone is designed to provide the context information via short messages (SMS) and multimedia messages (MMS). Therefore, it is not very appropriate for today’s highly connected and real-time demanding applications.

Gu et al present SOCAM, service-oriented context-aware middleware, which can be considered an earlier work in this area showing the potential of using service-oriented architecture for context-information management. But they are mainly focused on context information modeling using OWL [9].

In this paper we present a service-oriented architecture choice for mobile computing where mobile applications deployed on mobile devices including smart phones and tablets are capable of providing context-aware services to both on-device applications as well as to remote applications on servers and other mobile applications in the cloud [7]. Our architecture is based on RESTful web services, and therefore it is a lightweight solution perfectly fitting in mobile environments. We also present a prototype implementation and its evaluation on JavaME platform.

3. MoReCon USE CASE MODEL

“Mobile RESTful Context Aware Middleware”, or MoReCon in short, is a middleware we designed and developed for context-aware mobile application development. MoReCon middleware can be used to develop context-aware applications in a layered way such that MoReCon abstracts and hides the details of context-aware data and services from the designer and developer of context-aware applications. MoReCon is used via a service-oriented API that will be explained later in the paper. MoReCon is platform-independent, it can be ported to other devices and platforms easily.

Figure 1 shows the layered (multi-tier) usage of MoReCon. In this model, context-aware application, that is located at the top, accesses and utilizes the context information from the MoReCon middleware layer. MoReCon, which is located in the middle, runs on the mobile device platform that is located at the bottom on the device hardware.

With this multi-tier, layered structure, context-aware application development has a few advantages over the current ad-hoc context-aware application development approaches:

- 1) Context data retrieval: Application does not know the details about how the context information is extracted from sensors and other relevant hardware and software. MoRecon is responsible for obtaining the context information and hides this process from the user of the context information.
- 2) Context data model: Context data model is hidden from the end-user application. Context data is accessed via the middleware and the application does not need to know how the data is modeled, saved, or updated.
- 3) Context data access: Context data is accessed using a standard service protocol. It is independent of the data model and any application can access via different programming languages and packages.

4. SERVICE-ORIENTED DESIGN

MoReCon middleware is developed using a service-oriented approach. The services MoReCon middleware provides are accessed using RESTful web services [2] only. Therefore, the service implementation and consumption are lightweight compared to SOAP-based web services.

RESTful web services is the new approach for implementing web services [11]. It is simple and based on the standard HTTP protocol. Resources are located on the web (or on the device) and accessed via their unique URIs using HTTP commands GET, POST, UPDATE, DELETE.

For example, suppose that the resources that we will be servicing are the pictures in a folder, and each picture has an ID number. Then to get the picture 2, we might use:

```
GET http://example.com/picture/2
```

To remove the picture, we can use:

```
DELETE http://example.com/picture/2
```

Classical SOAP-based web services are heavier than RESTful web services, they require more processing time and memory. Therefore they are not suitable for mobile environments [5]. And, that is why we chose to use RESTful web services to implement the MoReCon API.

We developed a prototype of MoReCon implementation using Java Mobile Edition (JavaME). Our current implementation can be depicted as in Figure 2. In this design JavaME is another added layer, providing virtual machine facilities for portability so that the application is platform independent. JavaME has a Recordstore
package as a storage facility that we are using for storing context model and data. MoReCon accesses sensors for context data such as using GPS sensor for location information (Figure 2).

5. RESTful API
MoReCon has a unified access API using RESTful web services approach. In this method, all context data is entered or the middleware is instructed to receive the context data from a sensor, updated, deleted, or accessed via RESTful web services interface. Below we describe the RESTful interface for "location" context information only (Table 1). But the current prototype has been extended to include calendar, user profile data, and other context information. As a matter of fact, MoReCon is an extensible context management middleware in which new context data can added and accessed as explained later.

A detailed Java implementation design is shown in Figure 3.

Figure 2. MoReCon implementation model.

Figure 3. MoReCon detailed design.
Table 1. Context information access via RESTful API for MoReCon.

<table>
<thead>
<tr>
<th>REST URI</th>
<th>HTTP Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>/location</td>
<td>GET</td>
<td>Retrieve location information</td>
</tr>
<tr>
<td></td>
<td>PUT</td>
<td>Update location information</td>
</tr>
<tr>
<td></td>
<td>POST</td>
<td>Add location information</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
<td>Delete location information</td>
</tr>
<tr>
<td>/location/geo</td>
<td>GET/POST/PUT</td>
<td>Update location information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>as latitude and longitude</td>
</tr>
<tr>
<td></td>
<td></td>
<td>information</td>
</tr>
<tr>
<td>/location/city</td>
<td>GET/POST/PUT</td>
<td>Update location information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with city name property</td>
</tr>
</tbody>
</table>

Below we present the RESTful web service interface that is supported in MoReCon for all possible context information management.

Querying Context:
- GET /{context}
  Retrieve all {context} information
- GET /{context}/{property}
  Retrieve a certain {property} value for the {context}

Creating New Context:
- POST /{context}
  Create new {context} information with property and values. Data is posted as an attachment in XML.
- POST /{context}/{property}
  Create new {property} value for the given {context}. Data is posted in the attachment in XML.

Updating Context:
- PUT /{context}
  Update {context} information. All data is posted in the attachment in XML.
- PUT /{context}/{property}
  Update {property} value for the given {context}. All data is posted in the attachment in XML.

Deleting Context:
- DELETE /{context}
  Delete [context] information.
- DELETE /{context}/{property}
  Delete {property} value for the given {context}

As expected some of the commands require exchange of XML data to deliver complex and structured data. For example “GET /location” command returns all “location” context information for the current user in XML format. The response for example is the following XML data.

```xml
<location>
  <geo>
    <lat>48.69096</lat>
    <lon>9.14062</lon>
  </geo>
  <city>Ankara</city>
</location>
```

And as this implies, there is a hierarchical data structure that is captured in the XML data. The data paths in the hierarchical data structure can also be accessed directly using the RESTful interface, just like writing XPath queries, as shown in the following pattern.

```
Command /{context}/{property}/(property)/...
```

Property chain shows the access path to a specific property in the complex context data. For example for the above location data following is a valid command.

```
GET /location/geo/lat
```

It gets the latitude value of the geographic location for the current user and location.

6. EXTENSIBILITY

MoReCon RESTful API is extensible in the sense that it can be extended in the future to include new context information whenever necessary. This can be done dynamically by using the same RESTful API by posting new context data and resources via the middleware. Below is the syntax of all RESTful commands that are supported in MoReCon for adding new context data type and new context data properties:

Adding a new context type
- POST /meta/{context}

Deleting a context type
- DELETE/meta/{context}

Adding a new context property
- POST /meta/{context}/{property}

Deleting a context property
- DELETE/meta/{context}/{property}

7. IMPLEMENTATION

MoReCon is implemented using JavaME. This implementation has the following distinguishing features:

- Small packet size: MoReCon has a small footprint of 114KB. It can be deployed on any mobile device with compatible Java ME support.
- Modularity: MoReCon provides services for context-aware information management and it abstracts this functionality from the other context-aware application requirements. Therefore it is easy to develop modular context-aware applications with less effort on context-aware information management.
- Service-oriented design: MoReCon provides its services via RESTful services. Therefore, any device/user context data can be accessed from other devices and platforms seamlessly. This is truly better, making context data available to all.
- Interoperability: Due to service-oriented design, context data is available from any device to any application. For example context data can be shared between any applications on any type of platforms as long as they adhere to the RESTful protocol we depicted.
- Portability: Since MoReCon is developed using Java, it is portable to all mobile device platforms and environments as long as there is Java support.
Language independent: Since MoReCon provides context-aware information management functionality via RESTful web services, context-aware applications can be developed in any programming language and environment. Application developers only need to develop a simple RESTful client to produce and consume service requests to MoReCon interface.

Location transparency: With the help of directory services (future work) MoReCon allows location transparency to users and applications. A mobile user/application can register its device and location to a directory server and other devices and applications can communicate with the MoReCon services without knowing the actual device/user location.

8. EVALUATION
We have tested the RESTful protocol we developed on a standard PC with 2GB RAM and running Sun Java Wireless Toolkit 2.5.2_01 for CLDC (SDK 2.5.2_01) on Windows operating system.

For testing purposes the following RESTful commands are used (Table 2). It is for a specific user’s location information with longitude and latitude data for geographic location.

<table>
<thead>
<tr>
<th>Command</th>
<th>Request URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/location</td>
</tr>
<tr>
<td>POST</td>
<td>/location &lt;lat&gt;48.69096&lt;/lat&gt;&lt;lon&gt;9.14062&lt;/lon&gt;</td>
</tr>
<tr>
<td>PUT</td>
<td>/location &lt;lat&gt;48.69096&lt;/lat&gt;&lt;lon&gt;9.14062&lt;/lon&gt;</td>
</tr>
<tr>
<td>DELETE</td>
<td>/location</td>
</tr>
</tbody>
</table>

Table 2. Test queries

Each command is executed 50 times on recordsets with differing sizes from 1000 to 5000 records. Then the average response times are calculated. The reason the recordset is set to a high number of records (from 1000 to 5000) is because we assume that a shared recordset is maintained in the mobile platform for many users and applications. MoReCon finds the relevant user’s context information first in the batch and then searches for the relevant context data for the user in question.

Figure 4 shows the results for all 4 commands. We see that POST command has the lowest response time. This due to the nature of POST command, it basically adds the new context data to the end of the recordstore dataset. PUT is the slowest of four. This is due to the fact that the system first needs to locate the context information of the user in the recordset. This takes more time for larger datasets. DELETE and GET also acts similarly, slower than POST but responding somewhat faster than PUT.

9. CONCLUSIONS AND FUTURE WORK
In this paper we presented a novel service-oriented middleware MoReCon for context-aware mobile application development. MoReCon is designed to provide context information management functionality via lightweight RESTful web services. It is an interoperable device middleware for all kinds of devices, platforms and applications. MoReCon is application-, platform-, and language-independent. It is portable since it is developed in JavaMe.

Figure 4. RESTful commands average response times.

We presented some of the test results showing the response times for RESTful service requests under differing data loads and it is shown that MoReCon is responsive and performing well in all cases with varying degrees.

We are currently extending MoReCon to utilize semantic web technologies to provide “intelligent” context management services with simple reasoning capabilities. It can even be extended to include rule-based reasoning to take intelligent actions on the device. For example, if the user enters a grocery store (location context change), he/she can be reminded of the grocery list via a short message.

Since RESTful web services is based on an open protocol, there should be security measures to prevent unauthorized access. This is one of the issues that need to be dealt with in MoReCon.

10. REFERENCES

