Journal of Universal Computer Science, vol. 18, no. 11 (2012), 1410-1431
submitted: 30/11/11, accepted: 15/5/12, appeared: 1/6/12 © J.UCS

A Simple Model Based on Web Services to Exchange Context Information between Web Browsers and Web Applications

Jordán Pascual Espada
(University of Oviedo, Spain)
pascualjordan@uniovi.es)

Oscar Sanjuán Martínez
(University Carlos III of Madrid, Spain)
oscar.sanjuan@uc3m.es)

B. Cristina Pelayo G-Bustelo
(University of Oviedo, Spain)
crispelayo@uniovi.es)

Juan Manuel Cueva Lovelle
(University of Oviedo, Spain)
cueva@uniovi.es)

Patricia Ordoñez de Pablos
(University of Oviedo, Spain)
patriop@uniovi.es)

Abstract: Nowadays mobile devices are equipped with sensors and hardware elements capable of capturing many types of information from the real world, location, orientation, light level, temperature, etc. This information is known in some areas as context information. For years many mobile native applications use context information to support specific tasks. Most of the applications developed with traditional technologies don't have mechanisms to use most types of context information. This paper presents a lightweight approach to use context information in conventional web applications. The proposal defines a set of highly customizable XML tags, and included web applications that can express-specific requests for context information. A web browser designed following the proposed specification is responsible for processing the XML tags and send the context information to the web application using web services. In this paper we present the proposed architecture, then develop and evaluate a GPS navigator application based on this proposal.

Keywords: Web applications; Web services, Web browser, Mobile devices, smartphone
Categories: H.4.0**,** H.3.5**,** H.5.4

1 Introduction

At present the use of internet and web browsers has become popular in many electronic devices of different types, such as video game consoles, TV, music players, mobile phones, etc. Smart phones are positioned as one of the most popular devices for accessing to online applications and web applications. A significant percentage of accesses to web sites are made from mobile phones, nearly 6 billion mobile phones are connected to the internet worldwide, with a fast growing trend in recent years [ITU, 11]. Several factors have increased this trend, the low cost of mobile devices, high-speed wireless networks, and “unlimited-data” pricing plans and so on.

Commonly smart phones use native applications. These applications are specially designed for the device or a group of specific devices. Currently there is a big variety of mobile phone models with different specifications: screen size, memory, processing capacity and operating systems (iOS, Android, WebOS, Windows Phone, Symbian, etc). This heterogeneity between mobile devices makes the development of a valid application for different platforms very expensive for developers, because it requires deploy multiple programming languages and use the APIs and platform-specific technologies.

To reduce the overhead caused by having to develop and maintain the same application on different mobile platforms, many developers have decided to develop web applications rather than native applications. This measure not only saves development costs also ensures that their applications can be used by as many users as possible since most of the mobile phones of today include a web browser [Hernandez, 09].

Web applications and native applications are very different technically. Native applications running over the device's operating system, Web applications run on an external server that sends various types of files (HTML, scripting languages, CSS, etc.) to the mobile web browser, so that it can interpret these files. Depending on the objectives and functionality of the application, it can be developed as a web application, native or either. There are some limitations, native applications have several features that are difficult to “imitate” by web applications, such as complex 3D graphics, the management of device hardware components such as sensors, GPS, camera, and so on [Gossweiler, 11].

There are various definitions about what is the “context information” [Schmidt, 99]. We usually refer to the context information as information about the real world captured by electronic devices, such as location, the light level, the device orientation, temperature, etc. This information is usually obtained by a group of sensors and other hardware elements.

Mobile phones are very conducive to the collection of context information; these devices include various types of sensors, microphones, cameras, etc. The number of mobile applications that use context information has grown considerably in recent years. Today there are many applications that use real-world information to complete its functionality [Kim, 09][Lahti, 06] such as: the GPS for location, the camera to read bar codes, the microphones to capture voice commands, position sensors to interact with the interfaces, and so on. The context information can be a key aspect of how people interact with electronic devices, the integration of physical and digital world can develop applications that were previously difficult or not possible [Chua, 11].

The ability to capture the context information is one of the weaknesses of web applications in comparison to native mobile applications. The traditional mobile web browsers (Chrome, Opera Mini, Firefox, Safari, etc) do not have any mechanism to allow web applications to use the device sensors to capture context information and send this information to the web server. The introduction of the context information in web applications provides new possibilities for development, giving the possibility of implementing some features that were previously exclusive to native applications [Gossweiler, 11].

There are several proposals that introduce the use of some types of context information in web browsers and web applications [Coppola, 10] [Noltes, 08]. Many of these proposals focus only on a few types of context information, such as location; the location is perhaps one of the most useful context information types, but certainly not the only one [Schmidt, 99]. Another feature that is common to most proposals is the use of context information in the user background, the web browser and the web application exchange context information without the user perceiving it, this approach is commonly used to provide context-sensitive content or services, recommendation systems, etc.

This proposal raises a system that makes possible for web applications to have access to the greatest possible number of context information types, according to the technical possibilities of the device (GPS, camera, sensors, etc). This approach is based on the idea of context information being managed by the application user, in this proposal, sending context information to a web application is conceptually similar to the sending of any other information on the web; text in an input form, upload a file, select an option in a checkbox, etc. This approach aims to provide a simple system so the context information can be used by the business logic process of the web application, not only as an element of personalization services and content consumption.

The structure of the paper is as follows. Section 2 presents and analyzes some proposals that establish a relationship between web browsers and the context information. Section 3 presents the design considerations and design decisions of the proposal. Section 4 provides a description of the proposed architecture. Section 5 presents a prototype of a web GPS navigator developed using the proposal. Section 6 presents an evaluation and analysis of the proposal in two different scenarios. Finally sections 7 and 8 present the conclusions and future work.

2 Related Works

2.1 Context aware applications

More than fifteen years ago, the advantages of using context information in software applications began to be visible [Schilit, 94]. From then until today, there were many proposals that included the use of context information in the computer area. Many proposals are specific applications or systems that use some type of context information in a way or another. These proposals usually adapt the management of context information to specific scenarios, such as social networks IYOUIT [Boehm, 08] CenceMe [Miluzzo, 08] and tourism [Lamsfus, 10].

Other proposals are aimed at developing applications that use context information. Most of these proposals are mainly frameworks or architectures that support the development of context-aware native applications [Johnson, 07] [Biegel, 04] [Raento, 05]. Some proposed frameworks combine the management of context information with reasoning elements [De, 09]. In most cases the native applications are developed in different types of mobile computing devices (PDAs, smart phones, etc). These devices have adequate technical features to capture context information [Chua, 11].

Some frameworks provide supports to simplify the development tasks and some aspects of application functionality. For example the Context Toolkit [Dey, 01] offers one solution for supporting context-aware application prototyping. Its design separates context acquisition from use and supports context interpretation, distributed communication, constant context availability, context storage, and resource discovery. Several of these proposed frameworks include interesting features like rapid development and reusability. Most of the proposals are useful for developing context aware native applications, but these frameworks have not been conceived specifically for developing web applications based on existing web technologies.

2.2 Context aware web applications and context-aware web browsers

Some authors propose to include the use of different types of context information in web applications, some of these proposals modify the traditional web browsers to make it context-aware [Challiol, 07], other proposals are based on the use of additional applications or plugins to complete the process of managing the context information [Noltes, 08]. Integration with the context information is useful not only in web applications, also in other web technologies such as web services [Ennai, 08], sometimes the context-aware web services can be used to build web applications [Kapitsaki, 08].

SENSE-SATION system [Shirazi, 10], which facilitates the development of web applications based on a community of mobile phones. The system consists on a runtime environment that is installed on the phone and a web based application platform. It is an extensible platform for integration that provides means to collect and manage information available on phones and making it accessible. Context aware web browser [Coppola, 10] uses the information provided by the surrounding environment in order to carry out a more refined search of web contents.

A different approach to exchange context information with web applications is the use of specific protocols. These protocols allow the management of few types of context information, mainly the location. Several protocols have been specifically designed to promote the exchange of context information between web applications and web browsers. Some of these protocols are: Secure User Plane location [Goze, 08], Mobile Location Protocol [Mobile, 02], Presence Information Data Format [Sugano, 04] and HTTP Enabled Location Delivery [Barnes, 08].

Web-Centric Application platforms are related in some respects with context-aware web applications [Gossweiler, 11]. These platforms have been expanded since the advent of HTML5 [Hickson, 11] and WebKit [WebKit, 11], like PhoneGap [PhoneGap, 11] that allows the development of applications using web technologies and specific libraries that access the device hardware API that can be used to capture context information. Applications developed with this platform are exported to various mobile systems: iOS, Android, WebOS, Windows Phone, and Symbian.

Most of the analyzed systems that supported the development of context-aware web applications are very useful for developing applications in specific scenarios, such as context-sensitive searches, recommendation systems, social networks, tourism, etc. But these systems are not designed to provide normalized and simple mechanisms that allow web applications to request any type of context information that can be captured by the device.

Many of the proposed systems only manage a small set of context information types, such as location. To promote the integration of context information and the independence of web applications that use it, the use of context information should not be conditioned by specific libraries, components, or web technologies.

3 Considerations and design decisions

The main objective of the proposal is to allow web applications to use all the types of context information that the device is able to capture.

There are many different mobile platforms, the architecture of context-aware web browser must be independent to be able to be applied to any platform (iOS, Android, WebOS, Windows Phone, and Symbian, etc). The implementation of some architecture elements that have access to the hardware APIs will have to be partially dependent on the platform.

The context information is personal and private; the user has to approve the communication context of context information to the application.

The types of Context information that devices can capture may increase in the near future; the proposal must have a structure that allows adding support for new context information types.

Web applications are developed using many different technologies, the use of context information should not be conditioned by any particular technology. The use of context information should be a simple and easy way for developers to base their processes on common web technologies such as HTML, XML, HTTP, etc, without having to use particular libraries or specific components.

4 Proposed system

4.1 Model

On one hand, the proposal defines an open group of context information XML tags that define the context information required. These tags can be included in the presentation layer of web applications.

On the other hand is the context-aware web browser, which is responsible for processing the context information XML tags and notifies the user that requires a specific type of context information. If the user gives permission to send the context information the browser runs the context information scheduled task linked to the active context information XML tag, this task captures the context information, processes it and sends it to the web application (Fig.1).

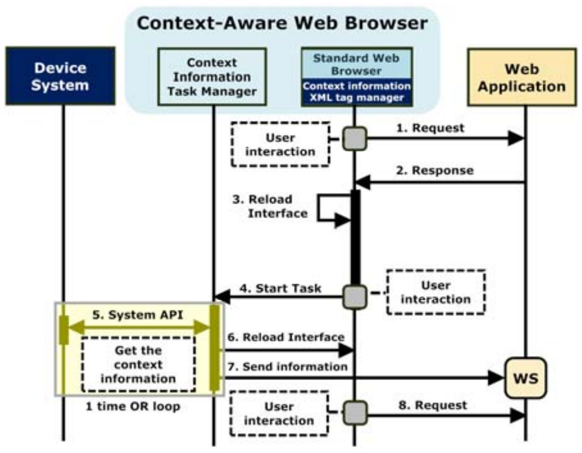

Figure 1: Sequence diagram of a simple context information request.

4.2 Request the context information: Context Information XML Tags

This proposal defines an open group of XML tags that correspond to many of the different types of context information that current mobile devices can capture. This group is open because the number of XML tags can grow in the future. Each XML tag notifies the context-aware web browser to request a specific type of context information.

The XML tags are included in the presentation layer of the web application, for example in the HTML code. The use of these tags is fully compatible with traditional web browsers that do not implement this specification and simply ignore the unknown tags.

Currently the specification defines the following context information tags.

Position:

- The three position sensors type most commons are: <magneticfield/>, <proximity/> and <orientation/>. These tags request the current value of the corresponding sensor, this value is usually a number or a set of numbers, for example the orientation sensor "<orientation/>" measurement is composed of three values; rotation around the Z, X and Y axis.

Motion:

- The most popular motion sensor is the accelerometer '<accelerometer/>' request the acceleration value in the three axes. There are others motion sensors such as <gravity/> or <gyroscope/> but are still rare.

Environment:

- <light/>, <temperature/>: these sensors get the value of an environmental parameter, usually the measurement of these sensors is recorded as a number.

Multimedia:

- <camera/>: takes a photo using the camera. Optionally, this tag can contain an attribute to specify the quality of the image.

- **<audiocapture/>:** activates the device microphone to capture the audio. By default, the audio recording ends when the user presses the button again. This XML tag may contain optional attributes to specify the recording time and the audio file format. One of the most commonly uses of the audio capture is the speech detection, the <audiocapture/> tag includes 'speech' and ‘language’ attributes that allows the execution of a speech detection process, as a result the web browser sends the text string instead of the audio file. There are several popular procedures to process some types of context information, as in this case the speech detection. Popular procedures (based on context information) included in the web browser functionality can reduce the complexity of the web applications and avoid code duplication.

  Location:
- The <location/> tag requests the device current location**.** An optional attribute is the source from which it obtains the location: GPS or Wi-Fi.

Sometimes a context information type can be captured in different ways, so some XML tags define several optional attributes that customize the behavior of the task that captures and sends the context information requested. Optionally, each of the context information XML tags can be combined with several attributes.

Common attributes:

- **‘timeinterval’** allows a request for context information to run in a loop, for example every 3 seconds.
- **‘id’** identifies a particular context information XML tag, the value of this attribute is sent with the context information to the web service. When a web application includes more than one tag of the same type it may include the optional attribute ‘id’, used to identify which specific XML tag corresponds the information that the web service receives.
- **‘group’** usually sending the context information is made individually, ie, each request for information of context generates a call to a Web service.
- **‘receiver’** specifies the URL of the web service to send the captured context information. If this attribute is not specified all context information XML tags are assigned a default value. For example < orientation /> is equal to: <orientation receiver=’baseurl/orientation’/>

Below are some combinations of context information XML tags and attributes:

```
<!-- Get location with default parameters, the response is sent to
'baseURL/location'-->
<location/>

<!-- Get location using GPS every 60s, the response is sent to
'baseURL/location'-->
<location
    method="GPS" timeinterval="60"/>

<!-- Get two camera picture with ids 'before' and 'after',  the response
is sent to 'cameraServlet' -->
<camera
    id="before" receiver="cameraServlet"/>
<camera
    id="after" receiver="cameraServlet"/>
```

```
<!-- Get the light and temperature sensor value,  the responses are sent
together to 'env.php'  -->
<light
    id="light-1" group="env" receiver="env.php"/>
<temperature
    id="temp-1" group="env" receiver="env.php"/>
```

Generic attributes can be used by any context information XML tag. Several context information XML tags can use unique attributes. For example, only "<location/>" tag can be combined with "method" attribute. Some of this unique attributes can only take constant values. In the case of "method" attribute that can only take the values GPS or Wi-Fi. These restrictions may result in errors. To allow developers to validate the XML code that defines the context information XML tags we describe the structure of the tags in a group of XML Schemas.

Below is a snippet of code from the xsd file that defines the <location/> tag structure.

```
<xsd:element name="location">
    <xsd:complexType>

        . . .

        <xsd:attribute
            name="timeinterval" type="xsd:nonNegativeInteger"
            use="optional" />
        <xsd:attribute
            name="accuacy" type="xsd:nonNegativeInteger"
            use="optional" />
        <xsd:attribute
            name="sensitive" type="xsd:boolean"
            use="optional" />
        <xsd:attribute name="method" use="optional" >
            <xsd:simpleType>
                <xsd:restriction base="xsd:NMTOKEN">
                    <xsd:enumeration value="GPS"/>
                    <xsd:enumeration value="Wi-Fi"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
 </xsd:element>
```

4.3 Context information exchange

When the web application user presses a button that corresponds to a context information XML tag, the context aware web browser captures the required context information and then sends it to the web application. The context information is usually plain text or a file, this information is encapsulated in a POST request and sent to the URL specified on the context information XML tag. To receive the information sent by the context-aware web browser the application must include appropriate mechanisms to be able to process the POST request; RESTful [Yong, 08] [Fielding, 00] web services, php pages, jsp pages, java servlets, and so on.

In addition to the context information the POST request always contains the session identifier of the device (DSK) header; this parameter is used by the web application to identify the sender of the request and the session. The value of the DSK is the same for any request sent by the context-aware web browser in a session with a device. Besides the DSK the request headers can contain other parameters: ‘id’, ‘group’ and ‘members’.

In the most simple communication scheme the context-aware web browser sends a POST request for each context-information XML tag. The use of ‘groups’ is the proposed mechanism to reduce the number of request and improve the application efficiency. When the context information of all XML tags in the same group has been captured the web browser sends the information together using a single POST request (Fig.2).

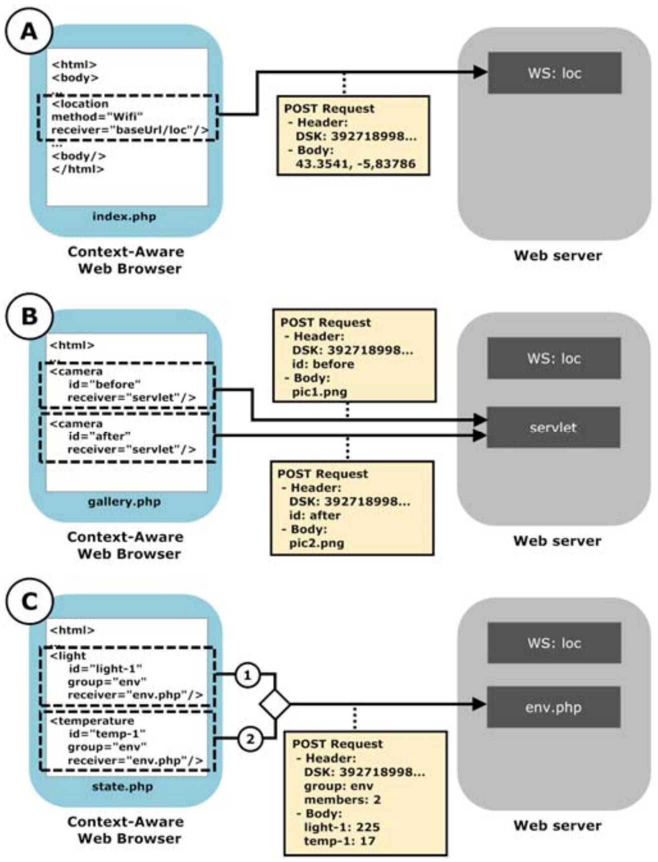

Figure 2: Context information XML tags communication schemes. (A) Simple scheme. (B) Using ‘id’ attribute to identify the context information XML tags. (C) Using ‘group’ attribute to reduce the number of requests.

4.4 Context-aware web browser

The context aware web browser is a type of web browser that is designed to meet the context information requirements expressed by the context information XML tags.

The specification of this browser is platform independent so it can be implemented using multiple platforms: iOS, Android, WebOS, Windows Phone, Symbian, etc. The context aware web browser main functions are:

- Interpreting the web pages based on web standards and conventional languages. In the same way as other classic web browsers.
- Notifying user requests for context information, if a website contains a context information XML tag the browser should display a specific button that lets the user decide whether to send that context information or not.
- When the user gives permission to share the context information, the web browser must run the corresponding context information scheduled task. These tasks are independent code modules contained in the context-aware web browser that capture context information using the operating system device API and then send that information to the web application.

The architecture of proposed context aware web browser is composed of three layers (Fig. 3) (1) standard web browser, (2) context tag manager and (3) the context information scheduled task manager.

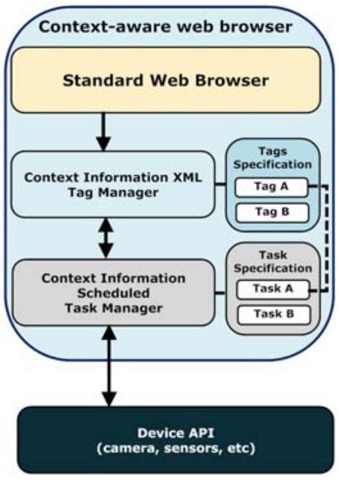
Context-aware web browser
Standard Web Browser
Context Information XML Tag Manager
Tags Specification
Tag A
Tag B
Context Information Scheduled Task Manager
Task Specification
Task A
Task B
Device API (camera, sensors, etc)

Figure 3: Diagram of the context-aware web browser architecture. The context-aware web browser is composed by tree layers, (1) standard web browser, (2) context tag manager and (3) the context information scheduled task manager.

Standard Web browser. The objective of the first layer is to interpret and visualize the web standards code. This layer has the same functionality as a conventional mobile web browser, such as: Dolphin HD, Opera Mini, Mozilla Fennec, Safari, etc.

Context Tag Manager. The second layer is responsible for the management of context information XML tags. First, the browser parses the web response looking for XML tags that correspond to the context information XML tags that the device can process. If the device has technical resources to capture the context information the context-aware web browser adds a new button in a graphical layer above the HTML.

Each context information XML tag is represented by button that has a different icon depending on its category or function. All mobile devices do not include the same sensors; some buttons may be disabled if the device does not include the necessary hardware to capture the context information defined in the context information XML tag.

Context Information scheduled Task Manager. This layer contains and manages the context information scheduled tasks. Generally these tasks capture context information using sensors or other hardware element, depending on the task. Each context information schedule task has a specific purpose, such as obtaining the GPS location, the light level, etc. This proposal defines a concrete specification for each context information scheduled task. Each specification defines the processes involved in the task in a generic and platform independent way: the receiver parameters, constraints, data returns, etc. The implementation of a context information schedule tasks is a native code module, as well as the implementation of the rest of the context-aware web browser that would develop for a specific mobile platform (Android, iOs, Symbian, etc.), however some modules such as the standard web browser and Context tag manager could be developed using cross-platform technologies. The code of the tasks uses the API system to manage the hardware elements of the device that capture the context information. Regardless of the implementation language and mobile platform, a task will have exactly the same functionality and will return the same result (Fig. 4).

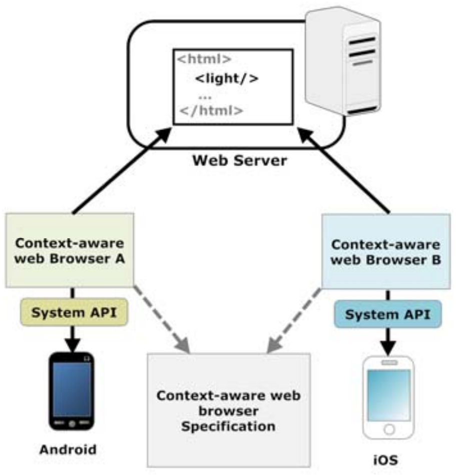
<html>
<light/>
...
</html>
Web Server
Context-aware web Browser A
Context-aware web Browser B
System API
System API
Android
Context-aware web browser Specification
iOS

Figure 4: Different implementations of the context-aware web browser developed according to the specification gets the same results when the user visits a web application that contains context information XML tags. In both cases the web browser functionality is equivalent, each context information scheduled task uses the platform system API to access the hardware elements.

The information obtained in the context information scheduled task is encapsulated in a POST request and sent to the default URL or the URL defined in the “receiver” attribute of the context information XML tag. The POST request is always completed by the header DSK and depending on the attributes of the context information XML tag attributes with the headers: ‘id’, ‘group’, and ‘members’.

The context information scheduled tasks have four common states: (1) Ready, (2) Processing, (3) Completed and (4) Error. When tasks change their state they will notify to context tag manager layer. The task receives as input parameters the context information XML tag attributes. The attributes define important properties of the task behavior. Depending on the attribute values, the task may be finish automatically after sending the context information to the web application, unless it has been defined as a loop using the *"timeInterval"* attribute, in this case the task will keep running until the user decides to finish it or to close the web application.

5 Prototype: web Application GPS Navigator

Context information XML tags allow the developing of web applications with features that would be very difficult to imitate using common web technologies. There are many different web applications that could be developed to illustrate the use of this proposal. As a prototype application, we decided to implement a car/walking navigator based on Google Map API.

We have considered developing a GPS navigator for two reasons: 1) this application type requires the use of two hardware elements; a GPS location chip and magnetic field sensor, although this sensor would not be absolutely necessary and 2) there are several GPS navigators, which offer a good opportunity to compare some features with those of the applications developed using context information XML tags.

The web application included two context information XML tags on the index.php page. The first tag requests the location of the device, the second tag requests the value of the magnetic field sensor; this sensor type can be used in some scenarios to simulate a compass.

In this application the **<location/>** includes several attributes that are used to configure the behavior of the associated location task. The application needs to confirm the location of the device every three seconds (timeinterval=”3”). We also use other attributes (accuacy=”10”, sensitive=”true”) similar to those included in the native GPS APIs to optimize the task performance and to avoid unnecessary delivery of information. The location is sent to the web service URL specified in the attribute tag.

In a very similar way to the **<location/>** context information XML tag used the **<magneticfield/>,** every three seconds the device sends the value obtained by the magnetic field sensor. Below is the index.php snippet of code:

```
…
    <div id="map_canvas"
        style="width:100%; height:100%"></div>
    <location
        timeinterval="3" method="GPS"
        sensitive="true" accuacy="10"
        receiver="/services/location" />

    <magneticfield
```

```
        timeinterval="3"
        sensitive="true" accuacy="10"
        receiver="/services/magneticfield" />
…
```

To interpret this context-aware web application we have developed a context-aware web browser by following the proposed specification. This version of the context-aware web browser has been developed as an Android application using the Android SDK. This context aware web browser contains an Android implementation of several context information scheduler tasks, including those associated with the <location/> and <magneticfield/> context information XML tags.

The context-aware web browser represents the context information XML tags as buttons (Fig.4). These tags will not be visible in a common web browser. When the user clicks on a button the browser will start executing the associated context information scheduler task.

When the user clicks on the location or magneticfield button the device starts a loop task, these tasks obtain the context information required and send the data encapsulated in a POST request to the web service URL, the process is repeated every three seconds. The button style changes to notify that the task is running (Fig.5).

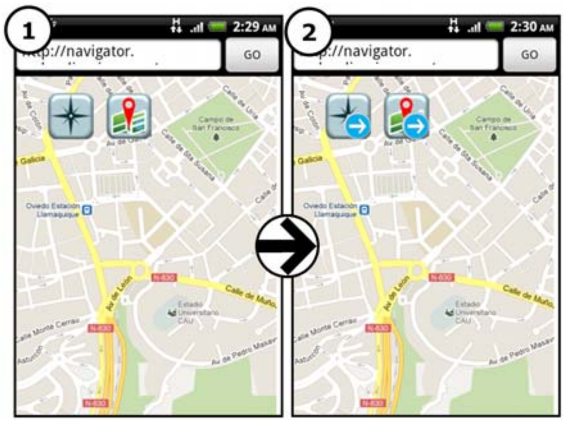

Figure 5: (1) Buttons notify the user all the requests for context information.(2) Location and magneticfield buttons indicate that the task is running.

The web application will have to implement the "/services/location" and "/services/magneticfield" web services, in this prototype we have implemented two PHP services that get the context information sent in the in the POST requests and stored in a server cookie. Each request contains the DSK header, that attribute is used to identify the device and the session. This application uses the Google Maps API, the index.php page contains a rich client based on html and javascript code. Commonly, web applications with rich clients call web services to obtain the necessary information to run their processes and update their graphical interfaces. In this case we have implemented two very simple web services that send the corresponding information stored in the server cookie (Fig.6).

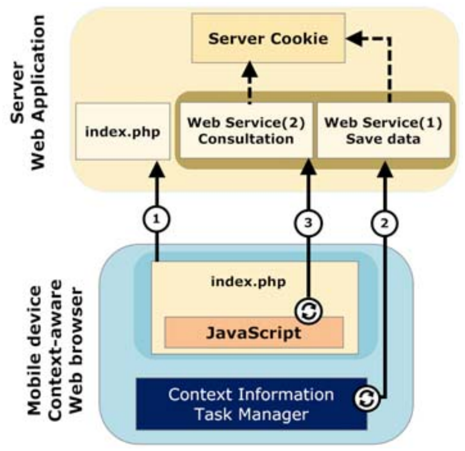

Figure 6: Functional GPS web navigator diagram. (1) The context-aware web browser loads the main page. (2) The page contains XML tags that execute a Context information task; this task invokes a web service that saves the context information in a cookie. (3) The index.php script calls a web service to update client information, location and orientation.

When the user initiates the context information tasks the web GPS navigator starts working, the location and orientation of the marker is updated every three seconds. The context information buttons are in a different interface layer to HTML so context-aware web browser can be configured to hide them (Fig.7).

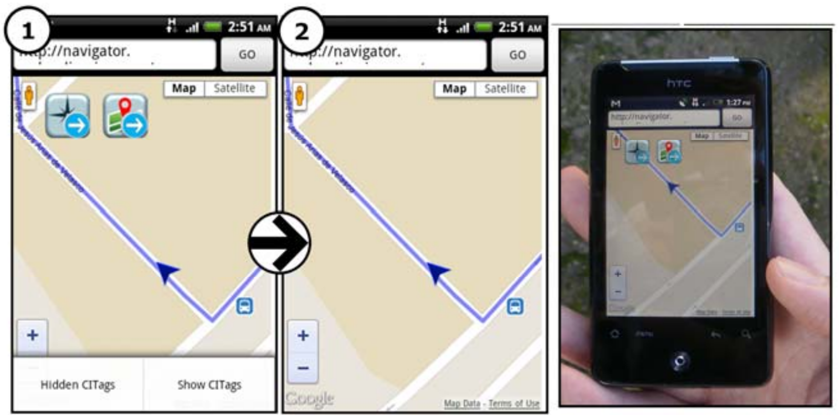

Figure 7: Screenshot of the web GPS navigator running.

6 Proposal evaluation

This section presents two evaluation processes: the first process involves the evaluation of a real application for it using the application developed in the previous section. The second process evaluates the network traffic generated by the use of context information XML tags in an individually scenario.

The web GPS navigator application has been developed using context information XML tags in this section we compare its performance with one of the most popular GPS navigators; the Google navigator for android.

For this testing process, we have used a HTC Aria Smartphones with the following features: Processor 600 MHz 384 MB RAM, 512 MB ROM, WiFi, Android OS 2.3. To obtain performance parameters in the simulations we have used three analysis tools: Traffic Monitor, Advance Task Manager, and TrafficStarts App / Task Manager.

For this simulation process a walking route (about 11 minutes) has been established. We have walked the same route using the two GPS navigation systems: 1) Web application: GPS navigator that uses context tags and 2) Native application: Google navigator. This procedure was done with each of the navigators 5 times. During the simulations we obtained some important parameters for the device performance: system memory usage (Fig.8), CPU usage (Fig.8), network traffic (Fig.9).

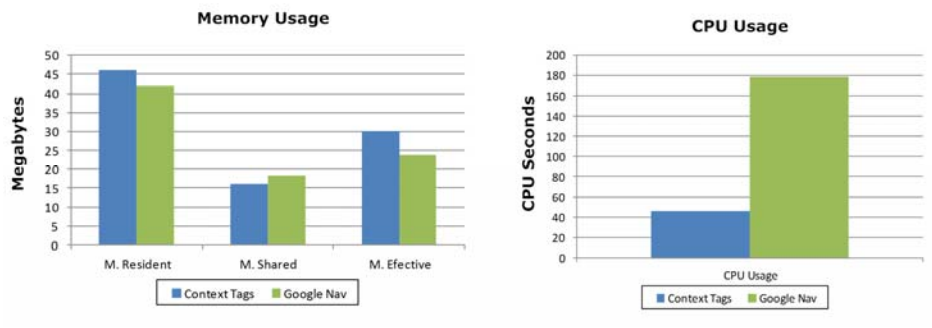

Figure 8: (1) Graphic representation of system memory usage during the execution of evaluated applications. (2) Graphic representation of CPU usage during the execution of evaluated applications.

The device performance expressed in memory usage gives similar values in two navigators. The native application Google navigator has a significantly lower memory usage to the web application navigator; however there is a big difference in the use of the device CPU. In this case, the functional complexity of the native application is greater than the web application, which may partially explain the difference in CPU usage. It is quite possible that the native application running multiple operations using the device's CPU and that some of these same operations are running on the web server for the web application.

The network traffic statistics generated by the two navigators are quite different even though the systems use the same maps. As expected, the web application has a high network traffic compared to the native application. The difference about sent and received data is around 50%, the main cause of this divergence is the constant exchange of data between the web application and the web server. Although it seems a big difference, the network traffic of the application is less than 1MB sent and 0.4MB received in 12 minutes of use, these values are probably lower than those generated by the use of the web browser in the same period of time.

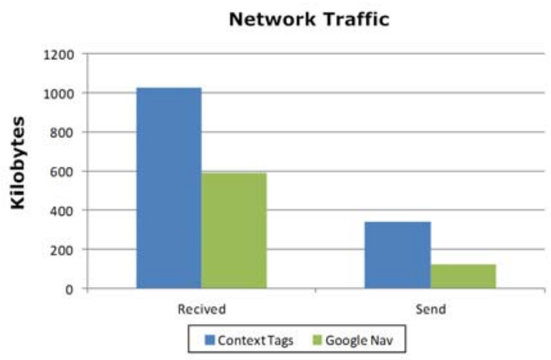

Figure 9: Graphic representation of network traffic received/sent during the execution of evaluated applications.

Comparing the resulting set obtained in the simulations, we can conclude that in the evaluated scenario the web application that use XML tags generates more traffic network, but instead it can run on many different platforms because it uses little CPU resources and runs on a context-aware web browser.

Previously we compared the performance of a native application with a web application developed using context information XML tags, in this simulation we analyze the performance of context information XML tags and compared it with the traditional data sending mechanism on the web: the form.

In this evaluation we have analyzed two important issues 1) the network traffic generated with the sending of data (Fig.10) and 2) the time spent since the information is sent until the user perceives the application response (Fig.10). For these comparisons we use twenty-five web applications divided into three types. Each of the web applications types sends to the web server the same data fields using: a HTML form, a group of context information XML tags and a variable number of individual (no group) context information XML tags. The number of sent data fields is increasing progressively from 1 to 5. For example when n=3, the first application contains a HTML consisting on 3 input fields, the second application it's a HTML that contains 3 context information XML tags (<audiocapture speech='true' group='info'/>) from the same group and the last application it's a HTML that contains 3 individual context information XML tags (<audiocapture speech='true' />).

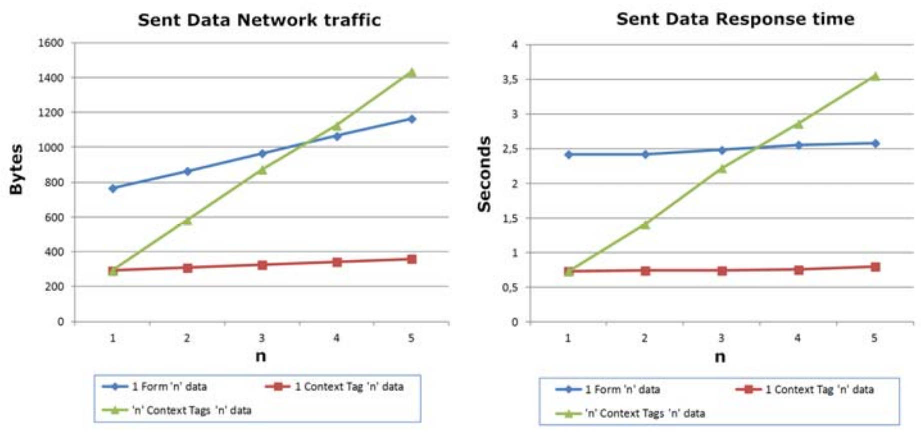

Figure 10: (1) The graph shows the volume of data transferred when the user sends data using three different approaches for sending data from the web browser to the server. (2) The graph shows the waiting times since the data is sent until the user perceives the response using three different approaches for sending data from the web browser to the server

Comparing the results obtained in the simulations we can conclude that in some scenarios the use of context information XML tags generates less network traffic and is a faster form the user perspective than traditional HTML forms. It appears from the analysis that the proposal is more efficient in scenarios that use one or two groups of context information XML tags for each HTML form. It should be noted that in some cases a context information XML tag could replace many form input fields as for example some web applications can use the GPS coordinates of the current location <location/> to infer the country, state and street.

The use of context information in web applications can improve the user experience in many scenarios. In general, mobile devices have several limitations in their interfaces: digital keyboard, small screens, the necessity to scroll to center the view on the page elements, etc. All the information that can be sent to the web application simply and without forcing the user to interact with the device interface is a step towards improving the user experience. The functionality of the application developed in section 6 could also have been implemented without the use of this proposal, but context information XML tags significantly reduce the number of user interactions with the device's interface. Therefore the application can be handled more quickly, decreasing the chances of the user causing errors and increasing the usability.

The following evaluation has analyzed the behavior of nineteen regular smartphone users. Users first used a traditional web application and secondly they used a web application that included two context information XML tags. The two web applications had exactly the same functionality (Fig.11). The use of context information XML tags allowed the second web application to capture the context information and prevent the user from having to introduce part of the required information manually.

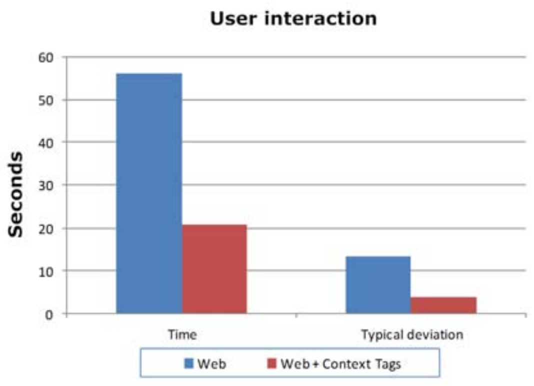

Figure 11: The graph shows the average time-consuming by users to complete the same assignment in a traditional web application and a web application that uses context-information XML tags.

In this scenario the context information has been used to re-implement the functionality of an existing web application. This web application allowed users to search for a product in nearby supermarkets, so this application was very appropriate to use context information. The result of this evaluation shows that in some cases the proposal may significantly improve user interaction times even up to 50% .This proposal may contribute to significantly improve the user experience in some web applications.

7 Conclusions

This proposal includes a normalized and extensible mechanism for web applications that can use most of the context information that mobile devices are capable of capturing. The use of this context information allows the implementation of features that were not possible before. Besides some types of context information can be useful to improve the functionality and user experience of some of the current web applications.

The proposed system has several interesting features:

From the point of view of developers, it is a simple and lightweight mechanism based on general purpose technologies (XML/HTML, and HTTP). The mechanism to request context information is easy to integrate into new web developments and into already developed web applications, without altering the application structure or conditioning their development using specific technologies, libraries, plugins or other components. The use of context information XML tags can be combined with traditional controls such as HTML forms, allowing the user to choose between using context-aware controls or traditional controls, a traditional web browser simply ignores the context information XML. The proposal also offers flexibility for web applications to collect context information; it only requires an element that is able to process POST requests. Context information XML tags simplify the part of the web application implementation that requests the context information, it only requires the use of a XML tag, while in most native applications (Android, iOS, Windows mobile, etc) is not so simple. This proposal has some aspects that could be improved. The development of web applications can be complicated if they use many types of context information, as it would be necessary to implement a web service for collecting each of the context information types.

This proposal introduces an uncommon approach in the context-aware web browsers and context-aware web applications: the system gives the user control over their context information. The requests and sending of context information are conceptually similar to any other information on the web; text in an input form, upload a file, select an option in a checkbox, etc. When a web application requests context information the user must decide whether to accept the request or not.

The proposal is not as efficient as native applications, due to the increased use of device resources and the generation of more network traffic, but the proposal brings other advantages over native applications. Web applications have not been developed for a specific mobile platform, which greatly simplifies the development process of context-aware applications that can run on different mobile platforms.

The system includes mechanisms to obtain performance and user experience that are very similar to traditional web applications that do not use context information, but could be even more effective in many scenarios from the point of view of data traffic and response times.

8 Future work

Some areas of future investigation:

- Conduct a study on political and commercial interest to focus innovation in native applications and how this affects distributed applications.
- Design tools or processes that offer support for the development of received context information systems, such as automatic deployment of web services that store the received context information in the session, cookies or databases.
- Development tools for the automatic creation of web services responsible for receiving context information.
- Additional proposal evaluations focusing on the impact on technological and commercial sectors
- Expanding the context information types that the tasks are able to capture, including support for external hardware components and other accessories can be connected to the devices, such as: external RFID readers, fingerprint readers, etc.
- Consider including new context information XML tags and context information scheduler tasks that use other types of context information.
- Improving the architecture of the context-aware web browser for a better performance and optimize the data traffic.

References

[Barnes, 08] Barnes, M.: HTTP Enabled Location Delivery (HELD), 2008, http://tools.ietf.org/html/draft-ietf-geopriv-http-location-delivery-08/ 2008.

[Biegel, 04] Biegel, G., Cahill, V.: A framework for developing mobile, context-aware applications. Paper presented at the Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on, 2004, 14-17 March 2004.

[Boehm, 08] Boehm, S., Koolwaaij, J., Luther, M., Souville, B., Wagner, M., Wibbels, M.: Introducing IYOUIT. In A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin & K. Thirunarayan (Eds.), The Semantic Web - ISWC 2008 Vol. 5318, pp 804-817: Springer Berlin / Heidelberg, 2008.

[Challiol, 07] Challiol, C., Fortier, A., Gordillo, S., Rossi, G.: A Flexible Architecture for Context-Aware Physical Hypermedia. Paper presented at the Database and Expert Systems Applications, 2007. DEXA '07. 18th International Workshop on, 2007, 3-7 Sept. 2007.

[Chua, 11] Chua, A. Y. K., Balkunje, R. S., & Dion Hoe-Lian, G.: The Influence of User-Context on Mobile Information Needs. Paper presented at the Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on, 22-25 March 2011.

[Coppola, 10] Coppola, P., Della Mea, V., Di Gaspero, L., Menegon, D., Mischis, D., Mizzaro, S., Vassena, L.: The Context-Aware Browser. Intelligent Systems, IEEE, 25(1), pp 38-47, 2010.

[De, 09] De, S., Moessner, K.: A framework for mobile, context-aware applications. Paper presented at the Telecommunications, 2009. ICT '09. International Conference on. 25-27 May 2009.

[Dey, 01] Dey, A. K., Abowd, G. D., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum.-Comput. Interact., 16(2), pp 97-166, 2001.

[Ennai, 08] Ennai, A., Bose, S.:MobileSOA: A Service Oriented Web 2.0 Framework for Context-Aware, Lightweight and Flexible Mobile Applications. Paper presented at the Enterprise Distributed Object Computing Conference Workshops, 2008 12th, 16 Sept. 2008

[Fielding, 00] Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures. Dissertation for Doctor of Philosophy, University of California Irvine, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm/, 2000.

[Gossweiler, 11] Gossweiler, R., McDonough, C., Lin, J., Want, R.: Argos: Building a Web-Centric Application Platform on Top of Android. Pervasive Computing, IEEE, 10(4), pp 10-14, 2011.

[Goze, 08] Goze, T., Bayrak, O., Barut, M., Sunay, M. O.: Secure User-Plane Location (SUPL) Architecture For Assisted GPS (A-GPS). Paper presented at the Advanced Satellite Mobile Systems, 2008. ASMS 2008. 4th, 26-28 Aug. 2008.

[Hernandez, 09] Hernandez, E. A.: War of the Mobile Browsers. Pervasive Computing, IEEE, 8(1), pp 82-85, 2009.

[Hickson, 11] HTML 5 W3C working draft, 2011, http://www.w3.org/TR/html5

[ITU, 11] The world in 2011 ITC Facts and figures, 2011, http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf/

[Johnson, 07] Johnson, S.: A framework for mobile context-aware applications. BT Technology Journal, 25(2), pp 106-111, 2007.

[Kapitsaki, 08] Kapitsaki, G. M., Kateros, D. A., Venieris, I. S.: Architecture for provision of context-aware web applications based on web services. Paper presented at the Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on, pp 15-18 Sept. 2008.

[Kim, 09] Kim, N., Lee, H. S., Oh, K. J., Choi, J. Y.: Context-aware mobile service for routing the fastest subway path. Expert Systems with Applications, 36(2, Part 2), pp 3319-3326, 2009.

[Lahti, 06] Lahti, J., Palola, M., Korva, J., Westermann, U., Pentikousis, K., Pietarila, P.: A mobile phone-based context-aware video management application. Paper presented at the Multimedia on Mobile Devices II. Edited by Creutzburg, Reiner; Takala, Jarmo H.; Chen, Chang Wen. Proceedings of the SPIE, Volume 6074, pp. 204-215, 2006.

[Lamsfus, 10] Lamsfus, C., Alzua-Sorzabal, A., Martin, D., Lopez de Ipiña, D.: Digital broadcasting for context-aware services in tourism. Paper presented at the Telecommunications Conference (HISTELCON), 2010 Second IEEE Region 8 Conference on the History of, pp 3-5 Nov. 2010.

[Miluzzo, 08] Miluzzo, E., Lane, N. D., Kristóf, F., Peterson, R., Campbell, A. T: Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application. Paper presented at the Proceedings of the 6th ACM conference on Embedded network sensor systems, Raleigh, NC, USA, 2008.

[Mobile, 02] Mobile Location protocol Specification, 2002, http://www.openmobilealliance.org/tech/affiliates/lif/lifindex.html/

[Noltes, 08] Noltes, J.: An Architecture For Context-aware Mobile Web Browsing , 9TH TSCONIT, university of twente, 2008.

[PhoneGap, 11] PhoneGap, 2001, http://phonegap.com/.

[Raento, 05] Raento, M., Oulasvirta, A., Petit, R., Toivonen, H.: ContextPhone: a prototyping platform for context-aware mobile applications. Pervasive Computing, IEEE, 4(2), pp 51-59, 2005.

[Schilit, 94] Schilit, B. N., & Theimer, M. M.: Disseminating active map information to mobile hosts. Network, IEEE, 8(5), pp 22-32. 2004.

[Schmidt, 99] Schmidt, A.: There is more to context than location. Computers & Graphics, 23(6), pp 893-901, 1999.

[Shirazi, 10] Shirazi, A. S., Winkler, C. Schmidt, A.: SENSE-SATION: An extensible platform for integration of phones into the Web. 2010 Internet of Things IOT, pp 1-8, 2010.

[Sugano, 04] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., Peterson, J.: Presence information data format (PIDF). Internet Engineering Task Force, 2004.

[Webkit, 11] Webkit, 2011, http://www.webkit.org/

[Yong, 08] Yong, L., Connelly, K.: Realizing an Open Ubiquitous Environment in a RESTful Way. Paper presented at the Web Services, 2008. ICWS '08. IEEE International Conference on, pp 23-26 Sept. 2008.