The Mobile Context Framework: providing context to mobile applications[*]

Luís Oliveira, António Nestor Ribeiro and José Creissac Campos

Departamento de Informática/Universidade do Minho & HASLab/INESC TEC
Braga, Portugal

Abstract. The spread of mobile devices in modern societies has forced the industry to create software paradigms to meet the new challenges it faces. Some of these challenges are the huge heterogeneity of devices or the quick changes of users' context. In this scenario, context becomes a key element, enabling mobile applications to be user centric and adapt to user requirements. The Mobile Context Framework, proposed in this paper, is a contribution to solve some of these challenges. Using Web servers running on the devices, context data can be provided to web applications. Besides the framework's architecture, a prototype is presented as proof of concept of the platform's potential.

Keywords: Context, mobile devices, mobile web servers, RIA, Web 2.0

1 Introduction

Many authors have already defined the concept of context. Dey et al. say that context is any information that can be used to characterize the situation of an entity [1]. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves. The same authors state that a system is context-aware if it uses context to provide relevant information and/or services to the user where relevancy depends on the user's task.

Mobile computing constitutes an excellent area where these definitions can be properly applied. In fact, as mobile devices become very common in modern societies, and with the significant improvements of their capabilities (in terms of software and hardware), these devices may store an important percentage of user context data. Besides, as the users carry the mobile devices with them, the context is always updated. The main problem is figuring out an efficient way to allow mobile applications to access and use this information. This paper proposes the Mobile Contextual Framework (MCF) as a solution to this problem.

[*] Published in *Distributed, Ambient and Pervasive Interactions*, volume 8028 of Lecture Notes in Computer Science, pages 144-153. Springer. 2013.
The final publication is available at link.springer.com.

2 Development of mobile applications

There are different approaches when considering the development of applications to mobile devices (see Figure 1). A first approach is developing applications that run natively in a selected platform. The advantage of this approach is that we can access the APIs of mobile operating systems and optimize the user experience for the target platform. However, one drawback of this approach is the tight coupling between applications and mobile platforms, resulting in high costs to maintain the same application for different operating systems.

A different approach is using web technologies (HTML5, CSS and JavaScript) to build a site/application that is accessed using a mobile browser. This results in applications that are generic – platform independent – and provide a good user experience with reduced effort both for developers and for final users. This approach has been gaining popularity mainly due to the fast improvement of mobile browsers and the currently numerous community of web developers.

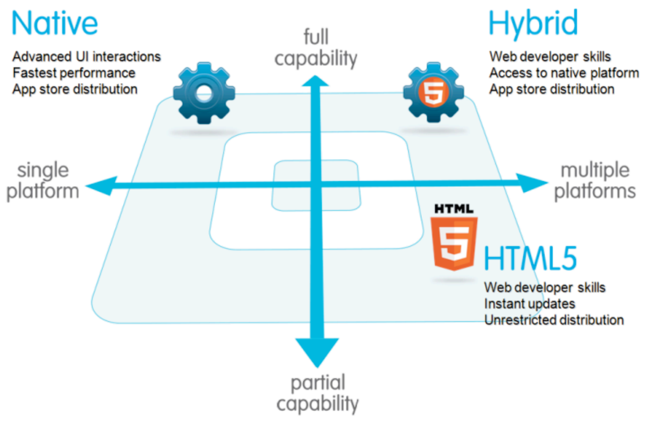


Fig. 1. Mobile development approaches (from [2])

A third approach – known as **hybrid** - results from the mix of native and web paradigms. In this paradigm, which we will adopt, the code is built using web technologies and is wrapped in a generic container, which is, in fact, a native application. This wrapper also exposes some of native APIs through a JavaScript (or similar) abstract layer allowing the developer to take advantage of some of features of mobile device. The main drawback of this approach is the, sometimes, poor performance of mobile applications and the lack of available frameworks. The most popular is Apache Cordova[1].

[1] http://cordova.apache.org/ (last accessed: 28/02/2013).

3 Contextual information in mobile devices

Applying the definition of context referenced above to the mobile computing world, we can consider context to encompass both static and dynamic properties. The **static** ones are the properties that do not change throughout time. Mainly, device characteristics such as model, capabilities, supported formats, among others. In fact, static properties represent only a part of the contextual data (not the most important). We should also be concerned with **dynamic** properties such as location, temperature, who's near, user PIM, user calendar, mood, financial status, etc.

Recently, the World Wide Web Consortium (W3C) created groups to work on integration of web applications with mobile devices such as **a) Device APIs WG** [3] which main goal is to create client-side APIs that enable the development of Web Applications that interact with device hardware, services and applications such as the camera, microphone, system sensors, native address books, calendars and native messaging applications; **b) Geolocation WG** [4] which mission is to define a secure and privacy-sensitive interface for using client-side location information in location-aware Web applications and **c) Web Applications WG** [5] that is chartered to develop specifications for web applications, including standard APIs for client-side development, and a packaging format for installable web applications. This group is also responsible to specify the APIs to deal with devices file systems. The main problem of these groups is the lack of implementation by industry on the specifications already published. Despite some exceptions, like the Geolocation API [6], which was already been successfully adopted by all the modern browsers[2], there is no standard way to get this information from devices (using local APIs) so current web applications have a reduced awareness of context. This constitutes an important drawback for the web paradigm.

This paper is a proposal to resolve this problem. Using the Mobile Contextual Framework (MCF), described in the next sections, it is expected that mobile applications should have three properties: (a) **Universality** – in the sense that the application should be independent of the device hardware and software; (b) **Context awareness** – in the sense that this factor is crucial for the success of the solution (in fact, today, users are constantly in movement so the context changes quickly, and, as the mobile phones become an essential tool for users, the application should know in detail one of these context in order to assist the users in an effective way), (c) **Low Cost** – in the sense that the effort to design, implement and maintain the solution should be small.

The solution implemented by the Mobile Context Framework – MCF (see Fig. 2) – is based on Web standards like HTTP and JSON. Instead of adding new capabilities to browsers, MCF uses local web servers. In order to get contextual data, the web pages need only to make local HTTP requests and process the JSON responses.

[2] http://caniuse.com/geolocation (last accessed 28/02/2103).

4 Mobile Web Servers

Mobile Web Servers are HTTP servers – like Apache or IIS – that run in mobile platforms. One of first versions was designed to run on top of .NET Compact Framework [7]. Later, Nokia launched a port of Apache – Raccoon[3] – targeted to Symbian devices. Other implementations are available for iOS[4] and Android[5]. Use-cases scenarios for the Mobile Web Servers, can be found in [8] and include sharing personal contents with others (e.g., contacts, calendar events, photos, videos, etc.); remote use via a web user interface (e.g., to locate the device); or the creation of context dependent content (e.g. obtaining information from people nearby).One of the issues when using mobile web servers in real scenarios is related to mobile operator's security policies. In fact, mobile operators networks are configured only to allow IP traffic originated from mobile devices. This means that an HTTP request done by an Internet host will not reach the target due to lack of **connectivity**. Additionally, it is not common that mobile devices connected to the Internet have fixed IP address. This brings another problem – **addressability.** Both issues can be solved using a gateway, outside the mobile operator network, in order to maintain HTTP connections from each device. These connections will be used whenever someone on the Internet makes a request to a certain device. Besides, this gateway guarantees the correct name resolution for mobile devices. This gateway-based solution is better explained in [9].

Upon the mobile web servers, developers are able to deploy web applications (like in other web servers). In order to allow running of web applications and dynamic pages, Raccoon implements a Python module - Python S60[6] – as well as the Personal Apache MySQL PHP module – PAMP[7]. Both modules provide APIs to guarantee direct access to operation system calls and, using this strategy, to get information about user PIM data or interact with GPS, camera, etc.

---

[3] http://sourceforge.net/projects/raccoon/ (last accessed 28/02/2013).
[4] https://github.com/robbiehanson/CocoaHTTPServer (last accessed 28/02/2013).
[5] http://code.google.com/p/i-jetty/ (last accessed 28/02/2013).
[6] http://sourceforge.net/projects/pys60/ (last accessed 28/02/2013).
[7] http://sourceforge.net/projects/pamp/ (last accessed 28/02/2013).

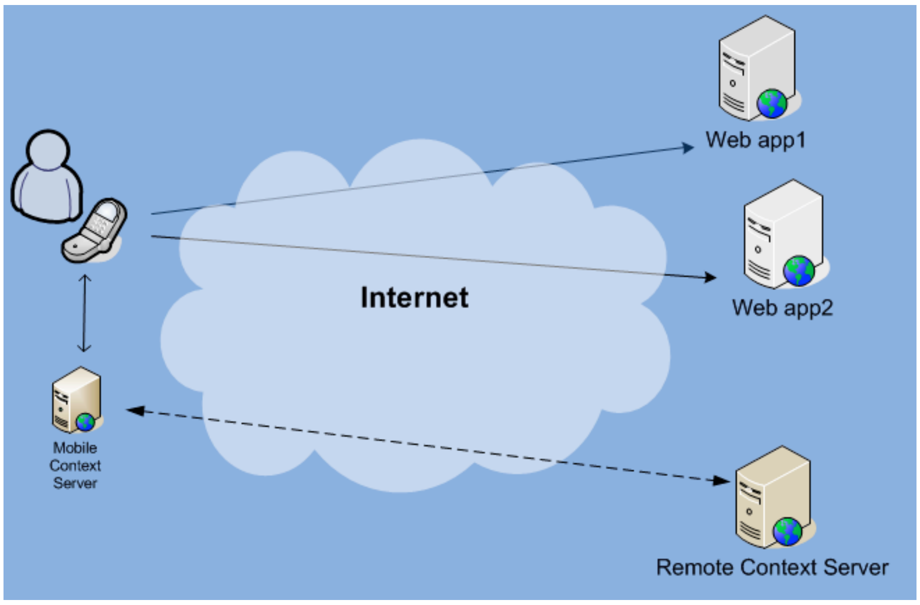


Fig. 2. MCF solution

5 MCF Architecture

The Mobile Context Framework (MCF) architecture encompasses two main components (see Figure 2):

1. **MCS** – Mobile Context Server
2. **RCS** – Remote Context Server

MCS runs on the mobile device as an application of the mobile web server. This component does not implement any logic dependent of a specific web application. It acts like a lightweight device API accepting HTTP requests and sending back responses in the JSON format.

In this way, the web application can access the device API, and gather contextual information just doing some more HTTP requests (to the localhost interface). From a user point of view, this solution is transparent since it does not require any additional device configuration or software installation in order to work properly (assuming that the mobile web server is installed by default in mobile devices). From a technical perspective, this solution joins both **remote HTTP requests**, in order to retrieve content from a remote web application, as well as **local HTTP requests**, to allow access to user context.

The MCS reference implementation was developed on top of the Raccoon server and using the Python module. The API is divided into two scripts: a main script – API-MCS.py - to manage metadata and textual responses (PIM, location) and an auxiliary script to fetch multimedia files such as images or videos in a streaming mode (Figure 3).

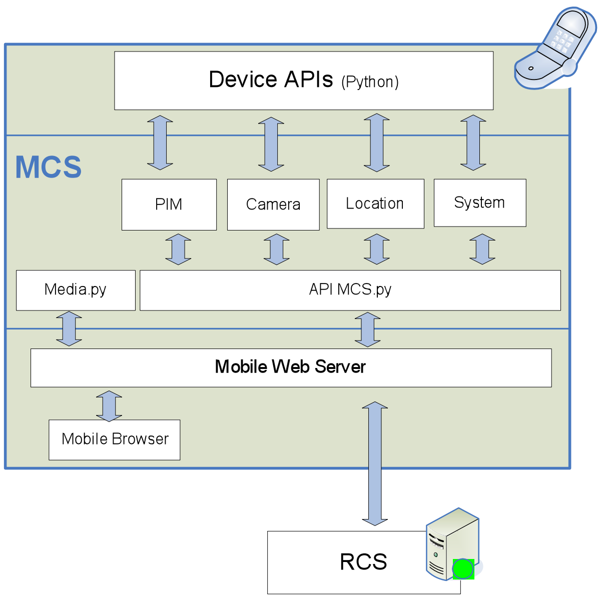


Fig. 3. MCS architecture

The contextual data, stored in mobile devices, is grouped in 4 categories: PIM, Camera, Location and System. Each of these categories has a correspondent class to manage, which acts like a bridge to device APIs. These APIs allow developers not only to grant access to context data but also to update it explicitly (e.g. to create a new calendar event). In what concerns to MCS API invocation, it is done asynchronously using callbacks to get the results.

As an example, let us consider a cinema tickets application, which contains a page listing the rooms near the user. To get the user current location (assuming that the user has a GPS device), the developer would need to add a script element which source URL should be http://localhost/mcs/api-mcs.py?op=getDeviceLocation. Besides that, it is required to implement the **mcf_callback** function in order to get the corresponding response (in this particular case, the location). In this example, the coordinates returned by MCF are used to make an Ajax request to a cinema service in order to get the rooms nearby. The full example code is listed in Figure 4.

```
<html>
<head>
<title>Location Test</title>
<script type="text/javascript">
var lat, longit;
function mcf_callback(obj) {
   lat = obj.location.lat;
   longit = obj.location.longit;
}
function updateRoomsDiv() { … }
</script>
<script type='text/javascript'
          src='http://127.0.0.1/mcs/api-mcs.py?op=getDeviceLocation'>
</script>
</head>
<body>
   <h1>Location Test</h1>
  <div id="result" style="display: none">
     <span id="status">Searching cinemas...</span>
     <img src="ajax-loader.gif">
  </div>
<script type="text/javascript">updateRoomsDiv();</script>
</body></html>
```

Fig. 4. MCS invocation example

Figure 5 depicts a sequence diagram of the interactions expected to occur in this example. First, the device browser performs a request to the cinemas web site hosted in the Internet. The response includes a reference to a localhost resource, which makes the browser perform an HTTP request to this local URL. This request is afterwards received by the mobile web server. Internally, the request is processed by MCS, which will perform a call to the operating system in order to get the device location. Once the result is obtained, it is sent back to the device browser. At the same time, the browser makes an Ajax request to get the cinema's set of rooms giving the location as parameter. When the final result is sent back to the browser, it is dynamically included in the page. All these steps are done transparently for the user.

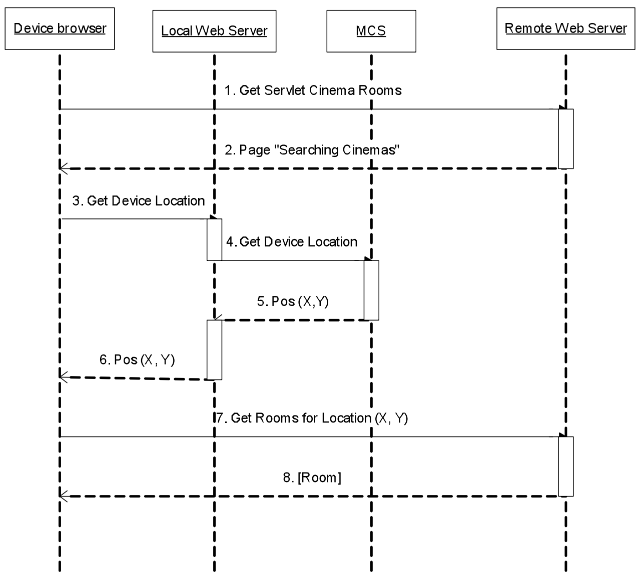


Fig. 5. Sequence diagram of a MCS invocation

The second component – RCS – is also a web application, and runs on a remote host. It includes an **Utils** sub-component, which helps MCS in more complex tasks. Consider, for instance, the processing of a 2D code. When a web page requests the MCS to read a 2D code, the component captures a photograph using the camera device API (assuming that user is pointing the device to a 2D code). Then, instead of doing the image processing in the mobile device it makes a request to the RCS. Besides this sub-component, the RCS also includes a **Trigger** sub-component, implementing rule based user notifications. For instance, if the user has a meeting in location X and, some minutes before the scheduled meeting time, he is far from it, the RCS would notify the user.

Internally, the trigger has a list of rules configured for each user. If the requirements of a rule occur, the corresponding event (in most cases, a notification to the user) is fired. In order to check the rules requirements, the RCS component makes periodical requests to the MCS, in order to get updated context information (for each user who subscribes some RCS rule). The mechanism that manages how rules are run is generic, independent of each specific rule.

While the MCS component has a **passive** behavior (it just responds to requests that are made when user is browsing in context aware web applications), the RCS component has an **active** behavior, mainly the trigger. This constitutes a great advantage when compared to solutions designed to work only in a passive mode. In the case of MCF, the MCS API is remotely called by the trigger component.

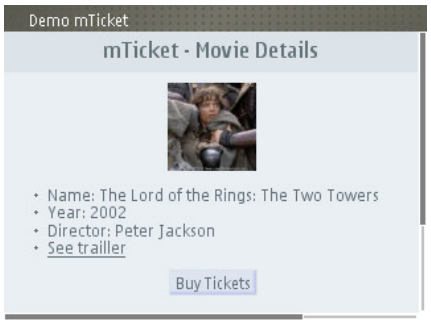
Regarding security and data privacy, the solution can run in a **local only** mode in order to guarantee that there's no remote requests performed to device web server. In this mode, the trigger component of RCS is disabled since it cannot communicate with MCS.

6 An example

To illustrate the potential of MCF, a prototype was implemented. The prototype was based on the use case described next.

Mary is a university student in London and a cinema fan. One day, while waiting for the bus, she sees an advertising spot about a new movie. Curious about it, Mary picks her phone, opens her operator mobile portal page and points the phone to the 2D code next to the advertisement. The phone camera captures the code and the data stored in it is sent to the RCS in order to be processed.

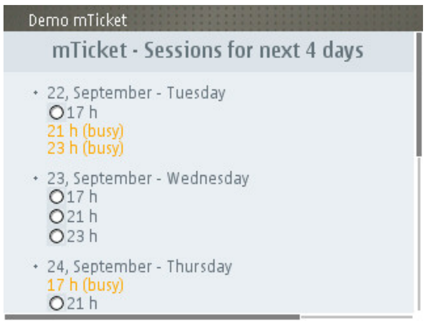
Next, Mary is presented a page with the movie details, including an option to see a trailer (Figure 6-a). After watching the trailer, she decides to buy two tickets, one for her and another for her boyfriend.


Demo mTicket

mTicket - Movie Details

- Name: The Lord of the Rings: The Two Towers
- Year: 2002
- Director: Peter Jackson
- See trailler

(a) Movie details page


Demo mTicket

mTicket - Sessions for next 4 days

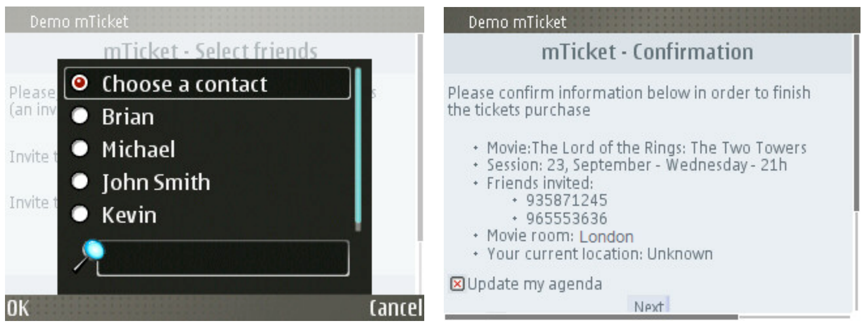
- 22, September - Tuesday
  - ○ 17 h
  - 21 h (busy)
  - 23 h (busy)
- 23, September - Wednesday
  - ○ 17 h
  - ○ 21 h
  - ○ 23 h
- 24, September - Thursday
  - 17 h (busy)
  - ○ 21 h

(b) Sessions list screen

Fig. 6. Choosing a session

In the screen to choose the date and session of the movie, Mary notices that some sessions are unavailable for selection (Figure 6-b). She then understands that those sessions are in conflict with events in her agenda. Mary chooses the Saturday night session.

After Mary has bought the second ticket, the application asks her for whom the second ticket is. Mary chooses her boyfriend contact from the phone contact list (Figure 7-a). Next, since this cinema chain has rooms in many cities in the UK, the application asks Mary to choose the room's location. John realizes that the default cinema room is already in London (but other options are also available). In fact, the application has made a location request to the device and it has chosen the nearest room as the default one. Then, Mary finalizes the tickets purchase (Figure 7-b) and, immediately, the application adds an event in Mary's agenda.



Please confirm information below in order to finish the tickets purchase

- Movie:The Lord of the Rings: The Two Towers
- Session: 23, September - Wednesday - 21h
- Friends invited:
  - 935871245
  - 965553636
- Movie room: London
- Your current location: Unknown

☒ Update my agenda

(a) Select friends screen (b) Confirmation screen

Fig. 7. Inviting friends

Some days later, Mary has forgotten about the tickets she bought and is travelling to Manchester that weekend. One hour before the movie starts, the application realized that Mary is far away from the room's location. Thus, it sends a short message to her, suggesting a change of movie location to Manchester since there are rooms from the cinema company there.

When considering the implementation of this use case, we could think that the cinema company implemented a native mobile application and that shared it to the customers. However, the company directors did not agree on paying a high quantity to develop and maintain versions of this application for the 4 main mobile platforms.

An application supporting the use case was implemented using MCF to provide context adaptation. The screenshots in Figures 6 and 7 are taken from the application. The application is provided as a Web application written in PHP. The code snippets presented throughout the paper, to illustrate MCF's usage, are taken from this particular application.

As illustrated by the use case description, the application is capable of accessing the device's camera and contact list, change the agenda, and request services from the server, in order to provide a contectualised usage experience to its users.

7 Conclusions

Enabling applications to access contextual information, allows for services with greater added value, and an improved usage experience. The mobile web is gaining an increased relevance in the information society age. It is foreseeable that in a few years mobile devices will be the primary means of accessing the web [10, 11]. At the same time, Web programming technologies evolution means that it is now possible to build complex applications that are largely device independent, and that can be made immediately available to anyone with a web enabled device. Finally, mobile web servers make it possible for those applications to have access to the user's context, without the need to install special purpose plug-ins in the browser.

This paper put forward MCF – the Mobile Context Framework – as a generic mechanism to enable mobile Web applications to interact with the mobile devices they will be *running* on. The proposed solution is invisible to users, assuming that devices will be delivered pre-installed with a mobile Web server, as argued for by Wikman et al. [9]. Besides the MCF framework, the paper presents a prototype application: Cinema Mobile Tickets. This application is a context sensitive application, built on top of MCF. The application is open source and demonstrates the feasibility of using the framework in a real life scenario.

Acknowledgments

This work is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - *Fundação para a Ciência e a Tecnologia* (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-015095.

References

1. Dey, A. and Abowd, G. Towards a Better Understanding of Context and Context-Awareness. In *Proceedings of the CHI 2000 Workshop on The What, Who, Where, When, and How of Context-Awareness*, Netherlands, April 2000.
2. Korf, M. and Oksman E. Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options. developer*f*orce Tech. Library, May 2012.
3. Hazael-Massieux, D. (Ed.) W3C Device APIs Working Group Charter. W3C, 2011. (available at: http://www.w3.org/2011/07/DeviceAPICharter.html)
4. Womer, M. W3C Geolocation Working Group Charter. W3C, 2010. (available at: http://www.w3.org/2008/geolocation/)
5. Schepers, D. and Barstow, A. and McCathieNevile, C. W3C Web Applications (WebApps) Working Group Charter, W3C, 2012.
6. W3C, "Geolocation API Specification", 2012
7. Pratistha, I. and Nicoloudis, N. and Cuce, S. A Micro-Services Framework on Mobile Devices. In *Proceedings of the International Conference on Web Services, ICWS '03*, pages 320-325. CSREA Press, 2003.
8. Wikman, J., Dosa, F. Personal Website on a Mobile Phone, Nokia Research Center, May 2006.
9. Wikman, J. and Dosa, F. Providing HTTP Access to Web Servers Running on Mobile Phones, Nokia Research Center, May 2006.
10. Huynh, S. Mobile Internet Users Will Soon Surpass PC Internet Users Globally. Forrester Blogs, February 21, 2012.
11. Lee, M. and Wong, J. Web access via mobile phone trumps PC in China: report. Reuters, Shanghai, July 19, 2012.