Using Mobile Device Detection Approaches to Augment the Accuracy of Web Delivery Content

Ricardo Queirós[1] and Mário Pinto[2]

[1] DI-ESEIG/IPP & KMILT, Porto, Portugal
ricardo.queiros@eu.ipp.pt
[2] DI-ESEIG/IPP & KMILT, Porto, Portugal
mariopinto@eu.ipp.pt

Abstract. Recent studies of mobile Web trends show a continuous explosion of mobile-friendly content. However, the increasing number and heterogeneity of mobile devices poses several challenges for Web programmers who want to automatically get the delivery context and adapt the content to mobile devices. In this process, the devices' detection phase assumes an important role where an inaccurate detection could result in a poor mobile experience for the end-user. In this paper we compare the most promising approaches for mobile device detection. Based on this study, we present an architecture for a system to detect and deliver uniform m-Learning content to students in a Higher School. We focus mainly on the devices' capabilities repository manageable and accessible through an API. We detail the structure of the capabilities XML Schema that formalizes the data within the devices' capabilities XML repository and the REST Web Service API for selecting the correspondent devices' capabilities data according to a specific request. Finally, we validate our approach by presenting the access and usage statistics of the mobile web interface of the proposed system such as hits and new visitors, mobile platforms, average time on site and rejection rate.

Keywords: Device detection; XML repositories; m-learning.

1 Introduction

In a recent survey [1] at our school - ESEIG (Escola Superior de Estudos Industriais e de Gestão) - we state that a large number of our students use, on a regular basis, mobile devices. The survey shows us that they are already engaged with mobile technology and are eager to use their devices in several scenarios from accessing to the ESEIG's web site (e.g. to consult news and events), to accessing the Learning Management System for course content, assignments and grades. Moreover, we also noticed that our students use different mobile devices with different characteristics that hinder the user mobile experience. These issues have sparked the creation of a web solution to enable the delivery of uniform web content on particular devices. This solution is composed by two sequential phases: device detection and content adaptation.

In the former, the proposed solution should determine the client characteristics and compare them with a devices' capabilities storage system. On this phase we should obtain a fully and accurate X–ray of the client device. In the latter, the Web content (text, images, audio and video) must be selected/changed based on the characteristics previously obtained to suit the user's computing environment and usage context.

In this paper we explore the use of device detection techniques to provide mobile users with a more rich experience. The outcomes of this study were the basis for the design of a system architecture [1] – called ESEIG Mobile –to detect and deliver uniform m-Learning content to ESEIG students. In this architecture we highlight the devices' capabilities XML repository and the REST API Web Service. The repository aims to store a meaningful number of characteristics of mobile devices (e.g. number of colors, resolution). The REST Web Service is used to obtain these characteristics based on the client's HTTP header request.

We validate this approach by presenting the access and usage statistics on the proposed system. This statistics were collected through the Google Analytics service, in order to better understand the adherence to a mobile web interface (e.g. average time on site, rejection rate).

The remainder of this paper is organized as follows: Section 2 enumerates and compares several technologies for the devices' detection. In the following section we present the architecture of ESEIG Mobile and the design of its internal components. The next section we validate the ESEIG-Mobile prototype system based analyzing its usage data. Finally, we conclude with a summary of the main contributions of this work and a perspective of future research.

2 Device Detection techniques

Mobile content quality requires a full and demanding awareness of the special limitations and benefits of mobile devices [2]. Some examples of these constraints are the limited computational power, small screen size, constrained keyboard functionality and media content types supported. Due to those constraints the mobile content must be adapted to suit the mobile device characteristics. Adaptation means a process of selection, generation or modification of content (text, images, audio and video) to suit to the user's computing environment and usage context [3]. In order to provide content adaptation, one must acknowledge the characteristics of the client device. Several approaches appeared in the last years to address this issue.

One approach is to use the common capabilities of the mobile devices and ignore the rest. Finding the Lowest Common Denominator (LCD) of the capabilities of target devices, will allow you to design a site that will work fairly well in all devices. In order to allow content providers to share a consistent view of a default mobile experience the W3C Mobile Web Best Practices Working Group (BPWG) has defined the Default Delivery Context (DDC) as a universal LCD [4]. This purpose is commonly adopt, however it limits the devices with better capabilities than LCD and decreases the use of a wider and heterogeneous mobile audience.

The most used approach is the one that obtains context information through the HTTP headers. These headers can be used to obtain the capabilities of a requesting

device such as MIME types, character sets, preferred reply encoding and natural languages. In addition to the accepted headers, the User-Agent header includes non-standard information about the device and the browser being used. This lack of standardization affects the data interpretation and extension [5]. To overcome these difficulties the device profiling concept emerged in recent years as a definition of the profile data structure that is being covered by several standards, such as CC/PP [6], User Agent PROFile (UAProf) [7] and Wireless Universal Resource FiLe (WURFL) [8].

The W3C CC/PP specification defines how client devices express their capabilities and preferences (the user agent profile) to the server that originates content (the origin server).

The UAProf is a standard created by the Open Mobile Alliance (formerly the WAP Forum) to represent a concrete CC/PP vocabulary for mobile phones and defines an effective transmission of the CC/PP descriptions over wireless networks. Mobile phones that are conformant with the UAProf specification provide CC/PP descriptions of their capabilities to servers that use this information to optimize the content. The information is formatted in XML containing several attributes (e.g. screen size, color and audio capabilities, operating system and browser info, encoding).

WURFL is a repository describing the capabilities of mobile devices worldwide. It uses an XML configuration file which contains a comprehensive list of device capabilities and features. A huge community of developers contributes with device information feeding the WURFL file and reflecting the new mobile devices coming on the market. Nowadays, WURFL shares the hegemony on the device detection market with other products such as DeviceAtlas [9] and Mobile Device Detect [10].

DeviceAtlas is a commercial database for device detection created by dotMobi. DeviceAtlas incorporates many device databases and sources such as WURFL and UAProf and retrieve accurate JSON results. Recently, the project was updated by the the DeviceAtlas Personal - a SOA aware version. It works as follows: a user visits a Web site on his mobile device. Then the server forwards the User-Agent HTTP request header to the DA Personal service, and receives a response containing information about the user's device.

The Mobile Device Detect (MDD) project is a PHP solution for device detection. It is free for non-commercial sites. Rather than using a comprehensive user agent database, this project is based on a script that seeks for specific string fragments in the user agent string.

The following table presents a mobile device concurrency test [11].

Table 1. Mobile device concurrency test.

Method	Time (seconds)	Mobile	Non-Mobile
WURFL API	20,8	1090	482
DeviceAtlas API	1,2	527	1045
Mobile Device Detect	1,3	684	888

The data set includes 1,572 unique user agents. We can say that accuracy and performance are the two most important features to take into consideration when

selecting a device detection mechanism. Device detection is not guaranteed to be 100% accurate since user agent strings are highly variable and non-standardized. At the same time, DeviceAtlas and MDD present smaller processing times, but more inaccurate results than WURFL.

3 ESEIG-Mobile

In this section we present the architecture of ESEIG-Mobile as a new layer on the top of the existent ESEIG infrastructure. This project aims to standardize the delivery of learning content produced at our School (ESEIG) to the diversity of mobile devices used by our students. In the following subsections we present the overall architecture of the ESEIG-Mobile and its main components.

3.1 Architecture

The architecture of the ESEIG-Mobile system is described by the component diagram shown in Figure 1.

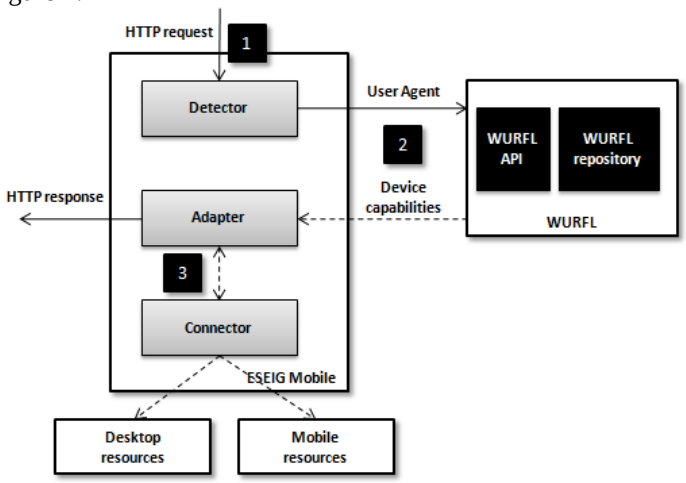

Fig. 1. Component's diagram of the ESEIG-Mobile system.

A typical execution flow will be as follows: the client device makes (1) an HTTP request; the Detector component at server-side invokes (2) a REST service with the user agent as parameter (collected in the HTTP user-agent header of the client request). The service seeks on the WURFL database and returns the respective capabilities to the Adapter component. The Adapter component based on the previously achieved characteristics of the device interacts (3) with the Connector component to select the more suitable content to compose the HTTP response back to the client.

3.2 Devices' Capabilities Repository

The Devices' Capabilities Repository contains a file with a large list of device features based on WURFL. WURFL is an open source database (file called *wurfl.xml*) of wireless device capabilities. The WURFL structure is currently formalized in a Document Type Definition (DTD) file. The following figure shows an overall view of the respective WURFL XML Schema.

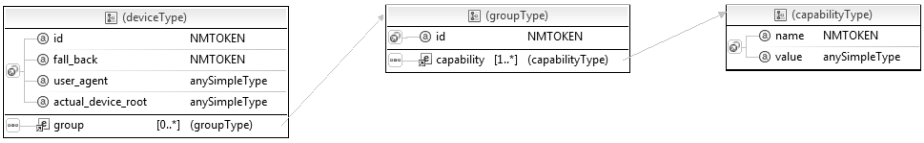

Fig. 2. The WURFL schema.

The schema has two top-level elements: the `version` and the `devices` elements. The `version` element is composed by a set of sub-elements: `ver` – the version of the WURFL database; `last_update` – the date of the last update of the database; `maintainers` – a set of maintainer elements related with the person(s) responsible by maintaining the database; `authors` – a set of author elements related with the person(s) responsible by creating the database;

The `devices` element contains one or more `device` sub-elements that model a certain device. This element contains the `user_agent` attribute, the device `id` attribute (created by the WURFL maintainer), the `fall_back` attribute (gives a way to infer more information about the device) and the `actual_device_root` attribute to signal that the current device element may be chosen as the representative for all devices by the same brand and model name.

In addition to this data, a device element may carry information about device features commonly referred to as capabilities. A device capability is an XML fragment which contains information about a specific feature of a given device. The device capabilities are organized in groups. Groups are used to improve the readability of the WURFL XML database by humans. For instance, Nokia phones support tables because *fall_back* is defined as generic (WURFL default) as described in the following piece of code.

```
<device user_agent="Nokia" fall_back="generic"
id="nokia_generic">
 <group id="ui">
  <capability
      name="break_list_of_links_with_br_element_recommended"
      value="false" />
 </group>
</device>
```

The WURFL is based on the concept of family of devices. All devices are descendent of a generic device, but they may also descend from more specialized families. This mechanism, called *'fall_back'*, lets programmers derive the capabilities of a given

phone by looking at the capabilities of its family, unless a certain feature is specifically different for that phone [8].

The WURFL repository can be either installed locally and be synchronized with a WURFL public repository where the developers' community makes updates regularly or accessed remotely through the use of a REST Web Service.

3.3 REST Web Service

A Web browser, when requesting a web page, sends a set of HTTP headers to the server. One of these headers is the User-Agent header that contains information about the user agent originating the request. The field can contain multiple product tokens and comments identifying the agent and any sub-products which form a significant part of the user agent as stated in the RFC 2616 [12]:

```
User-Agent = "User-Agent" ":" 1*( product | comment )
```

For instance, an Android mobile device may send the following user agent string:

```
User-Agent: Mozilla/5.0 (Linux; U; Android 2.2; pt-pt; GT-I9000 Build/FROYO) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1
```

The **Detector** component receives and uses it to query the WURFL device repository through the WURFL Web Service. This service provides a RESTful interface to the WURFL database. The use of this approach rather than a local implementation of WURFL avoids the maintenance of a local storage liable to the typical synchronization issues.

The following table details the WURFL API interface.

Table 2. WURFL REST Web Service.

Parameter	Description	Example
ua	User-Agent	http://api.wurflws.com/wurflws?ua=[UA]
search	Capabilities filter	http://api.wurflws.com/wurflws?search=[F^1|F^2|...|F^n]

The API's endpoint is http://api.wurflws.com/wurflws. The API has two parameters: `ua` and `search`. The `ua` parameter defines the User-Agent string that identifies the device. If not sent then the original User-Agent header is used to find the corresponding device. The `search` parameter represents the Capabilities filter. Only these capabilities (`is_wireless_device`, `brand_name`, `model_name`, `resolution_width`, `resolution_height`, `full_flash_support`, `flash_lite_version`, `mobile_browser`, `device_os`, `ajax_xhr_type`, `ajax_support_javascript`) should be returned if these parameters are sent. The capabilities should be separated by a pipe.

For instance, calling the following URL will return the capabilities of the Nokia 6630 with two filtered capabilities: resolution width and height:

```
http://api.wurflws.com/wurflws?ua=Nokia6630/1.0(2.3.129)%20SymbianOS/8.0%20Series60/2.6%20Profile/MIDP-2.0%20Configuration/CLDC-1.1&search=resolution_width|resolution_height
```

The response is a JSON string that will be parsed by the Adapter:

```
{"resolution_height": "208", "resolution_width": "176"}
```

4 Validation

In this section we validate the usage of the ESEIG-Mobile web interface, characterizing the access and rejection levels based on Google Analytics service data, such as hit counters, rejected requests, new visitors, traffic and mobile operating systems used to access the ESEIG-Mobile interface. The data was collected from November 2010 to February 2011.

Regarding the access rate (Figure 3), one can consider that although the access rate is relatively low, it has increased significantly. The amounts collected can result from the fact that the platform is very recent, and therefore still unknown by most students. Moreover, the high rates of new visitors may indicate that the ESEIG-Mobile web interface starts to be increasingly popular. This is reinforced by the high rate of new visitors, always above the 60%. This clearly shows that the service is gradually being known by students and teachers.

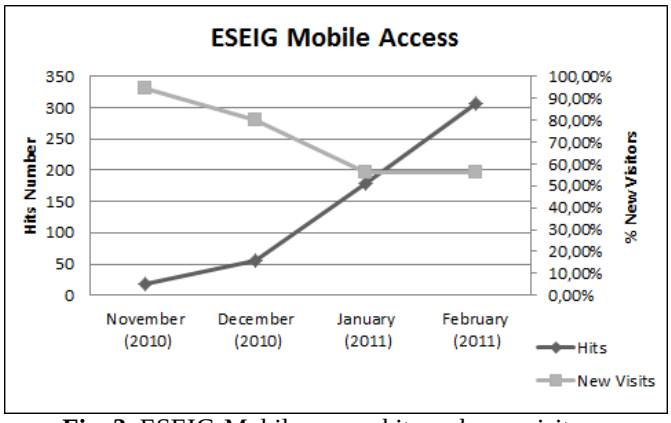

Fig. 3. ESEIG-Mobile usage: hits and new visitors.

Figure 4 shows the average time spent on the ESEIG-Mobile web interface and the number of pages visited (average) by access. This data are useful to evaluate the degree of interaction of each user and how it relates with the mobile platform. In fact, the average time on site is between one and two minutes, and the number of pages visited around two, by access. This data is in compliance with that obtained in ESEIG-desktop web interface, also collected through Google Analytics in the same period of time: average time on site around two minutes and number of pages retrieved 2.5. This is an interesting issue, since it shows that users remain interested

with the contents offered, at least to a similar degree to what happens with the desktop web interface. Other important conclusion is related with the average time on site that lies between 1 and 2 minutes. This fact indicates that the accuracy of the detector component is acceptable otherwise an incorrect approach would considerably decrease the current value.

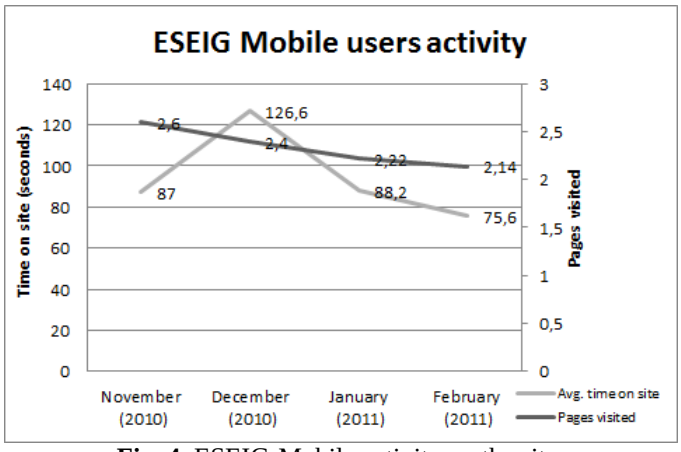

Fig. 4. ESEIG-Mobile activity on the site.

Another important issue that arises from the data analysis is the diversity and heterogeneity of the client devices. Symbian, Android, iPhone and iPad are the leading mobile devices, but there is a large number of other devices that ESEIG-Mobile system should respond. The support offered for different platforms and mobile operating systems is, certainly, a critical success factor for the ESEIG-Mobile web interface.

Figure 5 illustrates each of the mobile access platforms used by students, as well as its incidence. This in an important data, since it allows us to understand which are the mobile platforms most commonly used, and it returns some important feedback regarding the efficiency and effectiveness of the proposed approach detailed in this paper. In fact, Symbian and Android are the main platforms used to access the ESEIG-Mobile interface, followed by iPhone, iPad and iPod. A surprising fact is the lower number of devices with the Windows Mobile operating system.

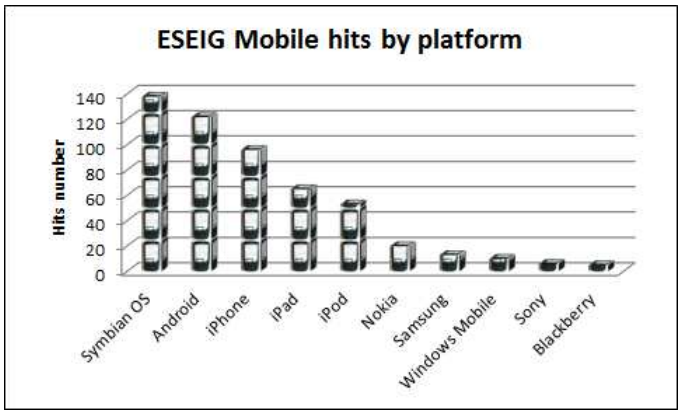

Fig. 5. ESEIG-Mobile access by platform.

5 Conclusions

In this paper, we present and compare several approaches for defining delivery context. Based on a previous survey and aided by this comparative study we present the design of ESEIG-Mobile - an open system for the delivery of suitable and uniform e-Learning content to the mobile devices of ESEIG students. The ESEIG-Mobile system relies on a devices' capabilities repository to store a meaningful number of characteristics of mobile devices and on a REST Web Service to obtain these characteristics based on the client's HTTP header request.

To validate our approach we present the access and usage statistics of the ESEIG-Mobile project based on the Google Analytics data. The analysis of this data is very important since it helps us to confirm and understand the heterogeneity of the students' mobile devices and their usage habits and preferences. It also helps to identify and find the best approaches to improve the ESEIG-Mobile system.

In this moment ESEIG-Mobile is in early development as we are only detecting if the HTTP request is made from a mobile device and query some device capabilities from the WURFL device repository. We expect some challenges in the prototype implementation process regarding, for instance, the transformation of the Web resources in the WNG format [13]. For this task we are considering using Extensible Stylesheet Language for Transformation (XSLT) to formally describe the transformations. Other ongoing work is related with increasing the device repository performance migrating from the WURFL XML database to a relational database (e.g. MySQL) using the Tera-WURFL project [14].

References

1. Queirós, R. and Pinto, M.: ESEIG Mobile: an m-Learning approach in a Superior School. CENTERIS'2010 - Conference on ENTERprise Information Systems, Viana do Castelo, Portugal, October, 2010.
2. Parsons D. & Ryu, H.: Software Architectures for Mobile Learning. In Mobile Learning Technologies and Applications, eds. David Parsons and Hokyoung Ryu. ISBN 978-0-473-11947-8. Massey University 2007.
3. Parupalli, R. : Dynamic Content Adaptation to Mobile Devices. In 3rd National Seminar on e-Learning and e-Learning Technologies, India, 2009.
4. Rabin, J. & McCathieNevile, C.: Mobile Web Best Practices 1.0" – basic guidelines, from http://www.w3.org/TR/mobile-bp/#ddc
5. Gimson, R., Lewis, R. & Sathish, S. (2006): Delivery Context Overview for Device Independence - W3C Working Group Note, from http://www.w3.org/TR/di-dco
6. Kiss, C. (2010). Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0 - W3C Working Group Note, from http://www.w3.org/TR/CCPP-struct-vocab2/
7. Wireless Application Protocol Forum, Ltd. User Agent Profile (UAProf), from http://www.openmobilealliance.org/tech/affiliates/wap/wap-248-uaprof-20011020-a.pdf
8. Passani, L. Wireless Universal Resource FiLe (WURFL), from http://wurfl.sourceforge.net/
9. DeviceAtlas – official website - http://deviceatlas.com/
10. Mobile Device Detect – official website - http://detectmobilebrowsers.mobi/
11. Keith, J.: Mobile Device Detection Results. In Cloud Four Blog, April 2009.
12. Fielding, R., Berners-Lee, T & et.al. HTTP/1.1 – RFC 2616 - www.w3.org/Protocols/rfc2616/rfc2616.html
13. Passani, L. (2010). Introducing WALL: a Library to Multiserve Applications on the Wireless, from http://wurfl.sourceforge.net/java/tutorial.php
14. Tera-WURFL – official website - http://www.tera-wurfl.com/