Crowdsourcing Mobile Web Applications

Cecilia Challiol[1,2], Sergio Firmenich[1,2], Gabriela Alejandra Bosetti[1,2], Silvia E. Gordillo[1,3], and Gustavo Rossi[1,2]

[1] LIFIA, Facultad de Informática, UNLP. La Plata, Argentina
{ceciliac,sergio.firmenich,gabriela.bosetti,gordillo, gustavo}@lifia.info.unlp.edu.ar
[2] CONICET
[3] CIC

Abstract. Building Mobile Web or Hypermedia Applications is usually difficult since there is a myriad of issues to take into account. Moreover adding support for personalized or context-aware behaviors goes far beyond the possibilities of many kinds of organizations that intend to build this kind of software (museums, city halls, etc). In this article we present a novel approach to delegate part of the effort in building mobile Web software to developers outside those organizations or even to final users. We show that this approach is feasible, light and practical and present a set of experiments we developed to verify our claims.

Keywords: Mobile Web Applications, Client-Side Adaptation, Crowdsoursing, Mobile Hypermedia.

1 Introduction

Mobile Web applications cover a great variety of domains (tourist, marketing, entertainment, etc.) to provide different services to the users, ideally considering his/her actual location. These applications must be “evolvable” and “adaptable” due to their particular characteristics. These characteristics may be classified from the point of view of the software: intensive use of users’ preferences, great variety of services, etc. or the hardware: wide variety of platforms and availability of devices. Many authors have researched on these topics [12], most of them consider the task of building this kind of software very complex.

Web applications have been evolving to provide services with different purposes, for example, for shopping, banks, museums, etc… The majority of these applications are not built considering the user’s location to provide services or content and of course they do not consider more sophisticated services such as providing mobile guides [7] and much less other well known features of Mobile Hypermedia [2] such as for example walking links [11].

To solve this problem, at least partially, many organizations, such as museums, tend to use their own existing websites to provide location-based information, for example, through barcodes which once read by the mobile device point the user to the correspondingly defined url. Another popular strategy is to use QRpedia[1] codes to provide Wikipedia pages associated to that code (which is in a particular location). This mechanism does not consider mobility aspects since the Web application (in this case, Wikipedia) is not designed to assist the mobile user.

One possibility, to solve this problem, such as is mentioned in [1], is to extend crowdsourcing technology to enrich mobile applications. So, each user can develop new services associated with a specific physical location, allowing the crowd to share the solution.

Inspired in part by the mentioned research we propose an approach to augment Web applications with mobile features, specifically implementing most interesting mobile hypermedia concepts such as mobile guides, walking links, etc. Our approach allows users to create their own tours according to their interest and eventually share them with others. The approach only requires an existing Web application and a mechanism to map locations to application objects (such as barcode or other kind of sensing strategy).

To create these augmentations, we use a client-side approach in the typical style of crowdsourcing; users create their personalized scripts that once run in the mobile device, enrich Web applications with mobile hypermedia functionality. Since the task of scripting might prove to be difficult we have develop a set of tools to improve and ease the process.

The structure of the paper is the following: Section 2 introduces a background of both mobile hypermedia and Web augmentation. In Section 3 we compare our approach with other related works. Section 4 explains our approach and presents our framework. An evaluation is presented in Section 5 in order to validate our approach. Finally, in Section 6 we present the conclusions and mention some further works.

2 Background

2.1 Mobile Hypermedia

Mobile Hypermedia Applications [2] are a particular kind of Mobile Application that extends the concept of navigation (used in the traditional Hypermedia) to the real world. In these applications, there are two ways to access to the information: digitally (as in typical hypermedia applications) and physically, based on the users' location. Corresponding with these types of information access we have digital links (the usual hypermedia links) and "walking" links, those links which connect two different physical objects (or real world object) and require the user to walk to reach the target object. Figure 1 shows digital and physical information access, and links in two levels and the association between them; real world objects are related with their hypermedia information, that we call digital counterpart (other author call as "digital-physical link" [10] or "physical hyperlink" [16]). The "mechanics" of mobile hypermedia is the following; when users are sensed in front of a physical object, its digital counterpart is shown; it might exhibit digital links which are navigated as usual or walking links (which point to others physical objects). When a walking link (or

[1] QRpedia page: `http://qrpedia.org/`

real-world link as mentioned in [13]) is triggered the application assumes the user intention to walk to this target; at this moment a map may be displayed in order to guide the user to the selected object; when he arrives, the user will be sensed to be in front of the object, and show information about it.

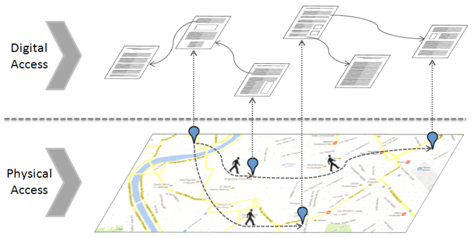


Fig. 1. Levels related with Mobile Hypermedia

Digital and walking links trigger different kind of navigation: the first one has a defined semantic for concepts like back and forward. The second one implies the user movement and, in this context, the user may change his decision to go to the target, she/he may get lost, etc; back and forward are more difficult to establish.

We have been working on different aspects related with Mobile Hypermedia Applications. These aspects cover different areas, such as, for example, context-aware assistance to the traveler [15], browsing behavior in these kinds of applications [5], modeling aspects [6]. In these works we mainly discuss issues related with the application of software engineering principles to the development of mobile systems.

We have been using separation of concerns to create Mobile Hypermedia applications in a model-based approach. The separation between pure hypermedia and mobility aspects allowed us to think in other ways to create mobile hypermedia applications, by using existing Web applications and adding the physical concern, e.g. positioning information. In this paper we are interested in light and user-centered ways of enriching Web applications to create mobile applications with those features of mobile hypermedia that improves the user experience.

2.2 Web Augmentation

Web augmentation [3] is a well-known technique for manipulating existing Web applications in order to improve them with new features. These approaches manipulate Web pages DOM (which is an object-based representation of HTML pages), since this is the only resource which is perceivable from outside of the application server. By changing the DOM we can alter what users perceive of the applications. We have successfully used Web augmentation for supporting users' concerns in traditional desktop Web applications [9]. In this paper we show a new approach used for building mobile applications running on mobile clients. As usual in the context of Web augmentation, our approach follows the idea of crowdsourcing: it allows users to define their own applications (or applications extensions) by developing specific scripts which respond to their own requirements; they may eventually share these scripts with the community.

3 Related Work

[8] describes an approach allowing Web applications to access context information in an easy and fast way. To do that, they create their own context-aware Web browser and a list of specific XML tags which are used to indicate that some particular page needs some context information, for example, location. In addition, each page requires implementing some mechanism to refresh the page with the current context. Their context-aware Web browser is designed in three layers: the standard Web browser, the context tag manager and the context information scheduled task manager. The context tag manager analyzes each requested page and if it has a specific context-aware XML tag, it activates the corresponding task to obtain the required context. The context information scheduled task manager, defines different tasks to obtain contexts of the user or of the mobile devices (using, for example, sensor information). When a specific task is called, the context information is encapsulated and it is sent using a web server to the web application (through some mechanism, the web page is updated with the current context). The authors present in [8] a context-aware Web browser defined specific for Android platform.

[17] presents a generic approach to enhance existing Websites with context-aware features on-the-fly. This work enriches a priori unknown, third-party Websites by using a semantic mechanism to extract information from these sites. Semantic information is used to enhance (through DOM manipulation) each page with the according context. This approach uses three methods of adaptation: context-aware recommendations, injection of contextual information and aids, and Websites user guides. The authors present in [17] a concrete application, using Android based on the SCOUT framework (which offers support to context-aware mobile application development). In this Android application each adaptation and the mechanism to enhance each page are defined.

Both approaches, [8] and [17] require users to install native applications in their devices; meanwhile in our proposal the user “just” installs browser plugins. Furthermore, our approach allows creating complex adaptations by only merging simple scripts; this users can combine these scripts easily (only activating them) or they can create their own scripts. Providing the same functionality with native applications is more complex, because these applications would need to provide a platform to create new adaptations.

Additionally [8] and [17] have defined specific adaptations types while in our approach each user may define his/her own personalized adaptations.

In [14], the authors present a crowdsouring approach to adapt mobile Web interfaces. The authors provide a toolkit for deployment adaptations, and then these adaptations are shared through a specific server. The adaptation are related only with those aspects associated with viewing context as, for example, size, position of websites elements, etc. This approach captures the context in which each adaptation takes place to increase the quality of adaptation scenarios. Then, this information is used to recommend the best-matching adaptations for a given context. This approach allows user to personalize (adapt) mobile web interfaces; when these adaptations are shared with the crowd, these can be used in a particular context. Adaptations are reviewed and rated by others users.

Our approach shares the same philosophy of [14], i.e. encouraging users to create their own adaptations, but while [14] focuses on interfaces, we focus on mobile features.

4 Our Approach

Our aim is to augment existing Web applications with mobile features, in particular with mobile hypermedia features such as walking links, active guided tours, etc. Our approach is client-side oriented with the typical style of crowdsourcing; in this sense we envision two different user roles: developers and end-users. Developers are JavaScript programmers who can create their own personalized scripts (which run in mobile devices) to enrich existing Web applications. End-users install and use scripts generated by some developer. In this section, we focus on the role of the developers and how our approach simplifies the task of augmenting Web applications, particularly, with mobile hypermedia features.

As previously indicated, we only assume the existence of Web pages and some mechanism (such as barcodes or other kind of sensing strategy) to map locations to those pages. One possible scenario is a museum, which uses QRpedia to provide Wikipedia pages in specifics locations to visitors. Figure 2 shows the user's view of an object in La Plata Museum.

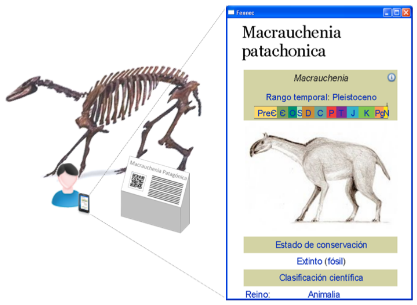


Fig. 2. User reads a QRpedia in a museum

While the Wikipedia page might have links to explore, they may not relate the object with other physical objects in the museum, nor give the user any cue about mobility through the museum rooms.

The same museum, meanwhile, may provide in its site a *Darwin tour*[2] which connects chronologically different animals of the museum providing information about each animal from the point of view of *Darwin*. An interesting augmentation of the current QRpedia-based "guide" results from implementing a mobile *Darwin tour*. A script developer may take the information from the *Darwin tour* site and augment, for each animal, the corresponding Wikipedia article (by manipulating the DOM in the client). Additionally, he may add walking links among museum's specimens to

[2] Darwin tour: `http://www.fcnym.unlp.edu.ar/museo/educativa/darwin/darwinenmuseo.html`

follow them chronologically; eventually he can add mechanisms for orienting the users in case they get lost or a warning if they reach a different target (not the one in the tour). For example, when such a link is selected a map with the path to the target object is shown. Figure 3 shows a general scheme of the relation between the script, the existing information and how it is connected with QRpedia (as a layer that is only active when the script is running).

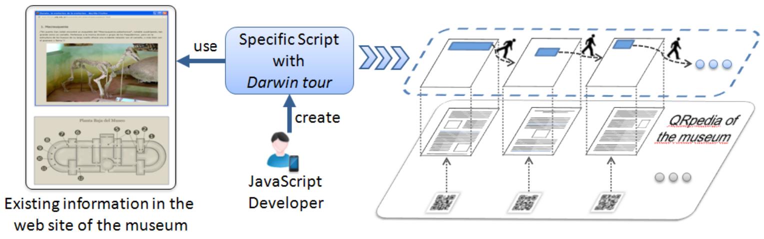
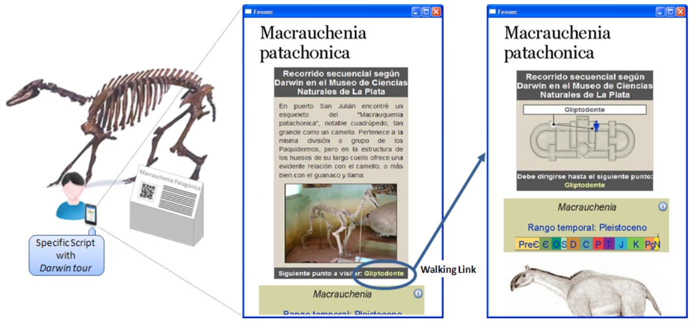

use
Specific Script with *Darwin tour*
create
JavaScript Developer
Existing information in the web site of the museum
QRpedia of the museum

Fig. 3. A developer creates a script with the *Darwin tour*

Figure 4 shows the results as perceived by the user. It can be appreciated that the user not only receives the Wikipedia page but also information related with the *Darwin tour*. In particular, a walking link is shown; when it is selected a map is displayed.


Macrauchenia patachonica

Recorrido secuencial según Darwin en el Museo de Ciencias Naturales de La Plata

En puerto San Julián encontré un esqueleto del "Macrauquenia patachonica", notable cuadrúpedo, tan grande como un camello. Pertenece a la misma división o grupo de los Paquidermos, pero en la estructura de los huesos de su largo cuello ofrece una evidente relación con el camello, o más bien con el guanaco y llama.

Siguiente punto a visitar: Gliptodonte

Debe dirigirse hasta el siguiente punto: Gliptodonte

Rango temporal: Pleistoceno
Specific Script with *Darwin tour*
Fennec
Macrauchenia
Gliptodonte
Walking Link
PreЄ Є O S D C P T J K Pg N

Fig. 4. The script with the *Darwin tour* is running

Different developers may create scripts implementing varied mobile functionality, e.g. personalized for different profiles (young students, disabled people, etc); the approach has two interesting benefits: first, it relieves the host institution (e.g. the Museum) from a task that might be outside its scope, and also allows arbitrary customization of the user experience with minimum effort.

However, in the example described above the developer needed to create all the mechanisms related with mobile hypermedia features as part of the script. The complexity might grow when, for example, there are other sensing mechanisms (e.g. in outdoor settings). For example, a script which uses GPS, needs to access the internal information of the mobile device in order to obtain the location. Additionally (though not further discussed in this paper), developers would want to generate visual adaptations of the original pages if they do not fit well with mobile devices (See [14]).

As part of our approach and to simplify the developers' tasks we have built a framework providing script skeletons for most mobile hypermedia features. Thus developers only need to instantiate the framework with the specific augmentations, e.g. concrete data and walking links. In Section 4.1 we summarize the characteristics of our framework and in Section 4.2 we briefly detail how users create scripts that use the framework. To run scripts generated by our framework, we provide a plugin, which is described in Section 4.3.

4.1 The Development Framework

Our framework, called MoWA (***Mo**bile **W**eb **A**ugmentation*), addresses the general functionality (or mechanisms) of mobile hypermedia applications. In addition, MoWA provides concrete functionally for specific domains. In this paper, we will focus on presenting the more general mobile hypermedia features. When developers use MoWA they only need to create a script which extends one of the framework's extension points. A further goal is helping developers to create the script in an easy way.

In Figure 5, we present a reduced schema of MoWA to understand its principal concepts. In the figure we use modules to group functionalities to show a more abstract overview.

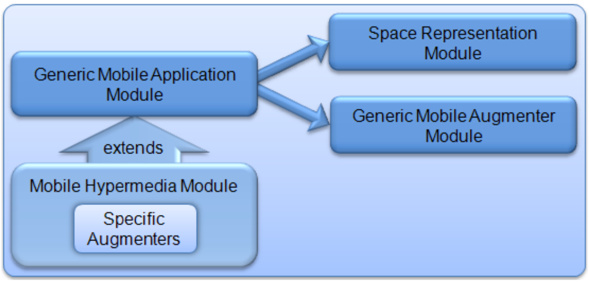

Space Representation Module
Generic Mobile Application Module
Generic Mobile Augmenter Module
extends
Mobile Hypermedia Module
Specific Augmenters

Fig. 5. Reduced schema of MoWA framework

We next present a more detailed explanation of the core functionalities provided by each module presented in Figure 5:

- *Generic Mobile Application Module.* This module implements generic concepts existing in all mobile hypermedia applications (in particular, we focus on those that consider the user's position to offer information). Some of these concepts are, for example, the information related with the user or with the points of interest.
    - For the user, it is important to represent his/her current position (sometimes the history of these positions is also important). Another aspect, as we focus on mobile web applications, is to save the navigation history.
    - For each point of interest its location and properties (which are used to augment the point of interest) are represented. These properties are

defined by each developer according to each specific augmentation. In MoWA they are represented in a general way. In addition, for each point of interest, a URL associated to it is defined (the URL can be expressed, for example, as a regular expression). This URL is used to resolve which page needs to be enhanced when the user is in front of the target physical object.

- *Space Representation Module.* In this module concepts related with the space are represented; this information is useful to eventually show a map or, for example, to find a specific path to an object of interest.
- *Generic Mobile Augmenter Module.* In this module generic augmenters, which help developers to enhance each point of interest defined in their scripts, are defined. For example, we provide augmenters to:
    - Show information about the point of interest in the corresponding page.
    - Show a map with the current location of the user.
    - Show some properties defined for the point of interest (in the *Generic Mobile Application Module*).
- *Mobile Hypermedia Module.* In this module the specific mobile hypermedia features are defined. The concepts represented in this module extend those defined in the *Generic Mobile Application Module*. For example:
    - The concept of user is extended to save specific information related with mobile hypermedia features, as the target (in case that a walking link is selected) or the history of navigation related with the walking links.
    - The concept of points of interest is extended to be able to represent the walking link between them.

  These extensions can be seen as wrappers of the general concept of user or point of interest respectively.

  In addition, as part of this module, we have defined specific augmenters oriented to adapt Web pages with mobile hypermedia features. For example, one of the augmenters we developed shows the path to the target in the map when a walking link is selected, which corresponds to the functionality shown in Figure 4.

MoWA can be used not only by developers who want to create specific scripts but also by the institutions, for example a museum, to provide suitable space representations. These space representations can be later used by developers in the scripts. In this case, developers create scripts using not only the “bare” framework but also the additional information offered by the institution (which is created using MoWA too).

In the following sub-section, we detail the steps that developers need to follow to create their scripts.

4.2 Creating Scripts Using MoWA

Developers with programming skills may use MoWA by programming JavaScript scripts. For doing it, developers need to follow these guidelines:

- Create a new concrete application, for example, one with the mobile hypermedia features. To do that, he/she only needs to create a new class which extends on, for example, *AbstractMobileHypermedia*.
    - Define the mechanisms of sensing which are relevant for the application, for example, barcode, GPS, etc. This information is used afterwards when the script is running in the browser.
- Create the instances of each point of interest which will be considered by the application. For each point of interest it is required to define:
    - Its properties (which will be used to enhance the page).
    - Its associated url (used to resolve which page is enhanced).
    - Its position (used to handle when and where the augmentation should be triggered)
- Define the walking links related with each point of interest. These links define the mobile hypermedia which users (of the script) rely on in order to "navigate" the real world.
- Create the space representation which can consist of complex structures or a simple image map. The developer needs to indicate where each point of interest in the defined space is located.
- Define an augmentation method for each point of interest. This method will be run when the user is positioned in the location associated with the point of interest. It is in this method where DOM manipulations may be defined (for a particular point of interest). Developers can use the augmenters defined in MoWA; for example to show walking links associated to the point of interest. If all the Web pages corresponding with each point of interest are augmented in the same way, developers may use a unique method for all points of interest.

Following the mentioned guidelines, we show in Figure 6 a *Darwin tour* specifies as an extension of *AbstractMobileHypermedia*. Figure 7 shows a simplified schema for the *Darwin tour* script.

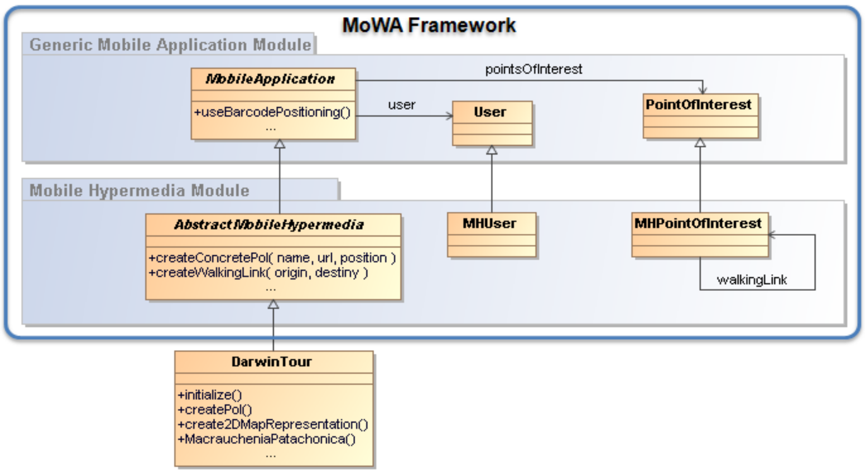
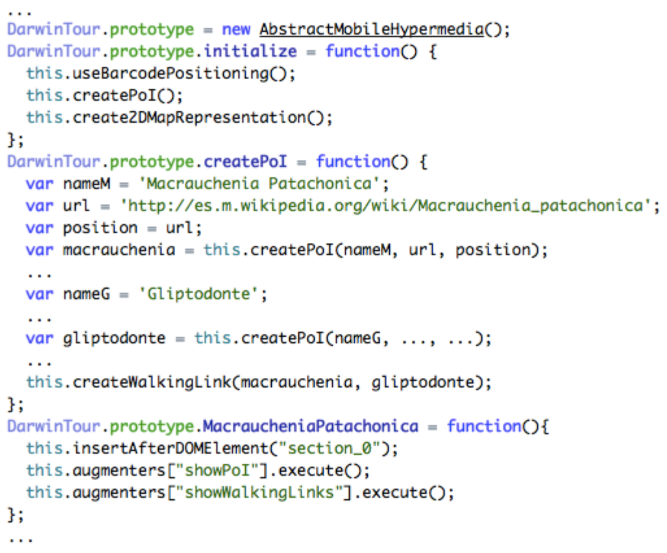


Fig. 6. Darwin *tour* application using MoWA



```
...
DarwinTour.prototype = new AbstractMobileHypermedia();
DarwinTour.prototype.initialize = function() {
  this.useBarcodePositioning();
  this.createPoI();
  this.create2DMapRepresentation();
};
DarwinTour.prototype.createPoI = function() {
  var nameM = 'Macrauchenia Patachonica';
  var url = 'http://es.m.wikipedia.org/wiki/Macrauchenia_patachonica';
  var position = url;
  var macrauchenia = this.createPoI(nameM, url, position);
  ...
  var nameG = 'Gliptodonte';
  ...
  var gliptodonte = this.createPoI(nameG, ..., ...);
  ...
  this.createWalkingLink(macrauchenia, gliptodonte);
};
DarwinTour.prototype.MacraucheniaPatachonica = function(){
  this.insertAfterDOMElement("section_0");
  this.augmenters["showPoI"].execute();
  this.augmenters["showWalkingLinks"].execute();
};
...
```

Fig. 7. Darwin *tour* script using MoWA

For creating the script related to the *Darwin tour*, developers do not need to define specific mechanisms to handle the involved objects; he should only define information related to each point of interest or related to the space; and in addition, he/she could use the augmentation methods provided by MoWA. So, his/her task to create scripts is easy. As said before the museum could offer its own space representation, so, the developer could use this representation, without the need to define this representation as part of his/her script.

To run these scripts, we provide a plugin to the mobile browser. A brief description of this plugin is detailed in the next sub-section.

4.3 Our Mobile Plugin

An important component of our approach is the mobile browser plugin in which we have included some MoWA components which are required to run scripts. In this section, we give a brief overview of this plugin to help the reader understand the mechanics of scripts created with MoWA. The following description is based on the Firefox plugin we developed.

We have divided our plugin in two modules. One of them, the *Scripting Module*, implements the mechanisms which are needed to run scripts created with MoWA. In this module, more than one application can be running at the same time; the module defines the order in which they execute – note that different applications may be defined to manipulate the same Web page -. The *Positioning Module* represents all the abstraction needed to handle the different location strategies, such as GPS listener or Barcode Reader listener. In the current version of the plugin we have created an implementation of the Barcode Reader listener. The *Scripting Module* is "listening" to the positioning events (triggered by the *Positioning Module*). Every time that the user's location changes, the *Scripting Module* is notified, and executes, for each application, the corresponding augmentation (if it is defined for this location).

Figure 8 shows a simplified sequence diagram of the *Darwin* tour script running our plugin. Suppose that this sequence occurs when the user reads the same barcode that has been showed in Figure 2. Then the *PositionManager* (which is included in the *Positioning Module*) detects that a barcode reader read a new code; it sends a notification to the *Scripting Module*, in particular to the *Darwin* tour application. As the url detected by the barcode reader matches with the url defined for a particular point of interest, then the augmentation method is executed and the page will be augmented with information related with this point of interest.

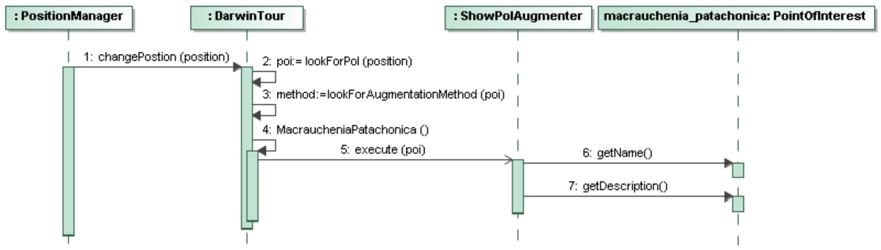

: PositionManager
: DarwinTour
: ShowPoIAugmenter
macrauchenia_patachonica: PointOfInterest
1: changePostion (position)
2: poi:= lookForPoI (position)
3: method:=lookForAugmentationMethod (poi)
4: MacraucheniaPatachonica ()
5: execute (poi)
6: getName()
7: getDescription()

Fig. 8. *Darwin tour* script running in our plugin

5 Evaluation

In order to evaluate our approach, we have conducted an experiment whose main goal was to explore how easy is to build mobile Web applications using our approach compared to a more traditional development approach (i.e. programming an entire application that runs on the server-side). Specifically, we wanted to determine if this kind of applications are developed faster and more effectively using client-side adaptation than with a traditional mobile Web development style.

With this purpose, we convened developers with different skills in Web development. A total of 16 developers participated in the experiment, all of them computer science students, aged between 20 and 39. We collected the programming experience of the participants with a preliminary digital questionnaire about: back-end technology experience (PHP, Spring, Python, etc.) and front-end experience (JavaScript, DOM manipulation, HTML, etc.). According to this information, we divided the participants into two groups of 8 participants each (to balance the knowledge in both groups): Group 1 (for developing a mobile Web application in a traditional way) and Group 2 (for developing a mobile Web application with our approach).

The rest of the experiment was performed during a week with two in-person meetings. This part of the experiment started with a meeting in which all 16 participants were asked to fill another questionnaire about Mobile Hypermedia background. Then, they were instructed about Mobile Hypermedia concepts, which took around 30 minutes. Afterwards we presented the task to be developed:

"La Plata Museum has several pieces exposed with an associated QRpedia code. We need a Mobile Web Application that supports a specific tour among specific pieces. The tour, called ***Darwin tour****, is based on the relationships Darwin found between these pieces. The application has to contemplate the current information provided for each piece plus new information about Darwin research (R1). Besides, the application must support users with walking links for showing them the following piece to be visited (R2) and correct them if they are in a wrong piece (R3). The application must allow users to visit the Museum in two ways (normally and with Darwin Tour) as well as allow users to cancel the Darwin tour. If the user finishes the Darwin tour, then a message must be shown informing this situation."*

After the general talk, we had two different talks (one for each Group); each separated talk took 20 minutes. Group 1 was instructed about tools for QR code generation and was advised about the freedom of developing the application using their preferred technology. On the other hand, Group 2 was trained about the use of MoWA and how it should be instantiated. We had a new meeting (with all participants) once they ran out of development time (a week later).

It is important to mention that all participants got the same artifacts: i) a QR code simulator running as a GreaseMonkey Script, ii) an HTML page showing the Museum map with an identified piece (from this HTML page participants could locate the pieces of the Museum and draw walking links between two pieces), iii) a full detail of the pieces contemplated by the *Darwin tour*: pieces, QRpedia codes, map location, etc. All of them were advised about the time available for developing the application: one week starting from the first in-person meeting. Besides, we asked them to register all their activity by using activity diaries, making a new entry for each work session (time used, tasks performed, problems founded, suggestions, etc.). Finally, we specified the concrete artifact to be delivered: the developed application, documentation about both, installation and use, the activity diary and a video showing the application.

At the beginning of the second meeting, we asked Group 1 participants to answer some questions about the development process; meanwhile Group 2 participants completed a SUS [4] questionnaire about the use of MoWA. Besides, we defined a general questionnaire (for all 16 participants) whose goal was to get the requirements satisfied for each participant, and the difficulty level found for the task.

Figure 9 summarizes the results in terms of effectiveness. From each requirement (R1, R2, R3) the figure shows how many participants of each group have satisfied it. Note that only four participants of the Group 1 could finish all requirements, meanwhile the rest of the same group did not finish any. Participants of the Group 2, however, were much more successful: seven of them finished the application completely, while the eighth finished R1, R2 but could not satisfied R3.

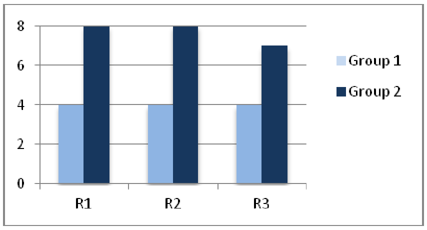


Fig. 9. Effectiveness of each group by requirements

Regarding the time took for developing the application: if we take into account only successful cases (4 from Group 1 and 7 from Group 2, where successful means all requirements satisfied), we found that Group 2 was faster. The average time of Group 1 was of 18 hours and 52 minutes (with a standard deviation of 7 hours 28 minutes), and Group 2’s average time was of 5 hours and 17 minutes (standard deviation = 3 hour).

Though this is a first evaluation of our approach, the first conclusion is that developing this kind of applications is much faster and effective with our approach than using traditional Web application development.

As a complementary result of our approach, the SUS questionnaire score was quite motivating for us, MoWA got a score of 81,3 which is an acceptable score for our first version of the tool.

6 Concluding Remarks and Further Work

In this paper we have proposed a novel approach for building Mobile Web Applications that reduce the intrinsic complexity involved in the development process of this kind of applications. Our approach takes into account personalization since users are able to install and use their preferred scripts. We made only two assumptions for applying our approach: i) a Web application exists, and ii) there is a mechanism to map locations to application objects.

Based on the concepts of crowdsourcing and client-side augmentation, our approach supports two roles for users: developers and end-users. For developers we provide a framework that can be used for specifying augmentations (for adapting Web pages in a particular way) and new Mobile Web applications (which coordinate these augmentations for making concrete the new applications over the original Web pages). For the two kinds of artifacts our MoWA framework provides extension points that make easier their development.

Meanwhile, end users are supported with a plugin to run these scripts inside the mobile browser plugin. Personalization is achieved by allowing end-users to install their preferred artifacts (scripts).

We performed an evaluation that shows that our approach is feasible. Moreover, according with the first results presented in this paper, we can figure out that developing using our approach is faster and more effective than using traditional Web development techniques.

Currently, we are working on visual tools for pre-building a specific application in order to simplify the development process. The main goal of these tools is to allow users with minimum programming skills to create personalized Mobile Web applications.

In the same line, we are extending MoWA with new features related with mobile Web application in order to support more functionality, for example, considering others sensing mechanisms (for example, the compass), providing more augmenters as part of MoWA, etc.

As part of our future work, we are planning to perform new experiments with users, but in contraposition with the experiment presented in this paper, we want to evaluate usability aspects of the Mobile Web application generated with our approach, which would involve the point of view of the end-users.

References

1. Alt, F., Shirazi, A.S., Schmidt, A., Kramer, U., Nawax, Z.: Location-based Crowdsourcing: Extending Crowdsourcing to the real world. In: 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries, pp. 16–20. ACM, New York (2010)
2. Bouvin, N.O., Christensen, B.G., Gronbaek, K., Hansen, F.A.: HyCon: a framework for context-aware mobile hypermedia. Journal of the New Review of Hypermedia and Multimedia 9(1), 59–88 (2003)
3. Bouvin, N.O.: Unifying Strategies for Web Augmentation. In: 10th ACM Conference on Hypertext and Hypermedia, pp. 91–100. ACM, New York (1999)
4. Brooke, J.: SUS: a 'quick and dirty' usability scale. In: Usability Evaluation in Industry, pp. 189–194. Taylor and Francis, London (1996)
5. Challiol, C., Muñoz, A., Rossi, G., Gordillo, S.E., Fortier, A., Laurini, R.: Browsing Semantics in Context-Aware Mobile Hypermedia. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2007 Workshop, Part I. LNCS, vol. 4805, pp. 211–221. Springer, Heidelberg (2007)
6. Challiol, C., Rossi, G., Gordillo, S.E., Fortier, A.: Separation of Concerns in Mobile Hypermedia: Architectural and Modeling Issues. In: Alencar, P., Cowan, D. (eds.) Handbook of Research on Mobile Software Engineering: Design, Implementation and Emergent Applications, vol. 1, pp. 211–233. IGI Global (2012)
7. Emmanouilidis, C., Koutsiamanis, R., Tasidou, A.: Mobile guides: Taxonomy of architectures, context awareness, technologies and applications. Journal of Network and Computer Applications 36(1), 103–125 (2013)
8. Espada, J.P., Crespo, R.G., Martínez, O.S., Pelayo, G., Bustelo, B.C., Lovelle, J.M.C.: Extensible architecture for context-aware mobile web applications. Journal Expert Systems with Applications 39(10), 9686–9694 (2012)
9. Firmenich, S., Winckler, M., Rossi, G., Gordillo, S.: A Framework for Concern-Sensitive, Client-Side Adaptation. In: Auer, S., Díaz, O., Papadopoulos, G.A. (eds.) ICWE 2011. LNCS, vol. 6757, pp. 198–213. Springer, Heidelberg (2011)
10. Hansen, F.A., Bouvin, N.O.: Mobile Learning in Context - Context-aware Hypermedia in the Wild. Journal of Interactive Mobile Technologies 3(1), 6–21 (2009)
11. Harper, S., Goble, C., Pettitt, S.: Proximity: Walking the Link. Journal of Digital Information (JODI) 5(1) (2004)

12. Hong, J.-Y., Suh, E.-H., Kim, S.-J.: Context-aware systems: A literature review and classification. Journal Expert Systems with Applications 36(4), 8509–8522 (2009)
13. Millard, D.E., De Roure, D.C., Michaelides, D.T., Thompson, M.K., Weal, M.J.: Navigational hypertext models for physical hypermedia environments. In: Fifteenth ACM Conference on Hypertext and Hypermedia, pp. 110–111. ACM, New York (2004)
14. Nebeling, M., Norrie, M.C.: Context-Aware and Adaptive Web Interfaces: A Crowdsourcing Approach. In: Harth, A., Koch, N. (eds.) ICWE 2011. LNCS, vol. 7059, pp. 167–170. Springer, Heidelberg (2012)
15. Rossi, G., Gordillo, S., Challiol, C., Fortier, A.: Context-Aware Services for Physical Hypermedia Applications. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1914–1923. Springer, Heidelberg (2006)
16. Schmalstieg, D., Schall, G., Wagner, D., Barakonyi, I., Reitmayr, G., Newman, J., Ledermann, F.: Managing Complex Augmented Reality Models. Journal of IEEE Computer Graphics and Applications 27(4), 48–57 (2007)
17. Van Woensel, W., Casteleyn, S., De Troyer, O.: A generic approach for on-the-fly adding of context-aware features to existing websites. In: 22nd ACM Conference on Hypertext and Hypermedia, pp. 143–152. ACM, New York (2011)