Evaluation of mobile app paradigms

Ngu Phuc Huy
Norwegian University of Science and Technology
Dept. of Telematics, O.S. Bragstads plass 2A, Trondheim, Norway
Tel: +841265583872
phuchuy86@yahoo.com

Do vanThanh
Telenor & Norwegian University of Science and Technology – Dept of Telematics
Snarøyveien 30 1331 Fornebu, Norway
Tel: +47 909 77102
thanh-van.do@telenor.com

ABSTRACT
The explosion of mobile applications both in number and variety raises the need of shedding light on their architecture, composition and quality. Indeed, it is crucial to understand which mobile application paradigm fits better to what type of application and usage. Such understanding has direct consequences on the user experience, the development cost and sale revenues of mobile apps. In this paper, we identified four main mobile application paradigms and evaluated them from the developer, user and service provider viewpoints. To ensure objectivity and soundness we start by defining high level criteria and then breaking down in finer-grained criteria. After a theoretical evaluation an implementation was carried out as a practical verification. The selected application is object recognition app, which is both exciting and challenging to develop.

Categories and Subject Descriptors
H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *Web-based services, data sharing.* H.3.4 [**Information Storage and Retrieval**]: System and Software – *Performance evaluation (efficiency and effectiveness).* K.6.3 [**Management of computing and information system**]: Software Management – *Software development.*

General Terms
Measurement, Performance, Standardization, Experimentation, Human Factors.

Keywords
Mobile app, mobile application, HTML5, mobile widget, mobile Web application, object recognition, evaluation

1. INTRODUCTION
Recently, smart phones with iPhone in the lead experience a huge popularity and the number of smartphones in the market has increased considerably. The reason behind this popularity is the plurality of useful and fancy applications [6], also called Apps. Although apps may have the same functionality there are many ways of implementing them such JavaScript, HTML5, applets, widgets, etc. Seen from the developers, users and service providers it is both interesting and relevant to understand the differences in terms of architecture, underlying mechanisms and functionality. Further, it is crucial in the development and selection of mobile apps to know which paradigm is more suitable for a given type of applications or usage. This paper is aiming at shedding light on the current most popular mobile app paradigms and providing a fundament for appropriate selection of mobile app paradigms. The paper starts with reviewing the related works. Next, the different mobile app paradigms are explained in a comprehensive way. The core of the paper is the evaluation of the existing mobile app paradigms. To verify the evaluation a practical implementation is carried out. A mobile object recognition/visual search app is chosen and is developed using the two most promising paradigms, namely native app and HTML5 mobile app. Evaluation results are also thoroughly discussed.

MoMM2012, 3-5 December, 2012, Bali, Indonesia.

2. RELATED WORKS
Andre Charland and Brian LeRoux in their article discuss the strengths and weaknesses of mobile Web app and native app paradigms, with the aim to make a comparison between mobile Web apps and their native counterparts [1]. Richard Padley shows how publisher can reduce development cost, improve time to market and deliver a cross-platform mobile application to end users [2]. Marie-Claire Forgue and Dominique Hazaël-Massieux state in their paper that MobiWebApp project aims to enable European research on Web technologies to shorten the gap between mobile Web apps and native apps [3]. Tian-gang Xu, Wei Wang and Xia Jia in their research review the APIs of several platforms and propose a solution to cross-platform mobile widget development by using Mobile Widget Portable Development Library (MWPDL) [4]. Tommi Mikkonen and Antero Taivalsaari in their paper present the ongoing battle of mobile native app and HTML5-based Web apps, and describe two alternative scenarios for the future of the industry based on the possible outcomes of the battle between the two mobile app paradigms [5]. Our paper differs from all the explained related works since it provide both a formal evaluation and practical verification of the four mobile app paradigms.

3. THE MOBILE APP PARADIGMS
In this section, we identify four mobile app paradigms, consisting of native apps, mobile widgets, mobile Web apps and HTML5 mobile apps.

3.1. Mobile native applications
A mobile native application or native app is an application specifically developed to execute on a specific device platform

[7] and machine firmware, and cannot be used for other device platform without modifications. For example, apps developed for the iPhone run only on Apple devices. A native app could be a stand-alone app running on the mobile phone or consisting of a main component on the mobile phone communicating with network servers. To use native apps users must download them from app store and install them manually on their phones.

3.2. Mobile widgets

Mobile widgets represent lightweight, task-specific apps that leverage Web content [8]. Mobile widgets exploit web technologies, including HTML, CSS, JavaScript and XML. Widgets will be executed within a runtime environment known as widget engine (e.g. Opera, Nokia WRT, Samsung TouchWiz and Yahoo!Blueprint). Different types of widgets need different widgets engines to execute. Like native app, a mobile widget could be a stand-alone application running on the mobile phone or consisting of a main component on the mobile phone communicating with a specific network server.

3.3. Mobile Web applications

Mobile Web application is a good paradigm to deliver information and service to mobile phone. A mobile Web app enables information processing functions to be initiated remotely on Web server. The three-tiered architecture [9] is the most popular Web app architecture, which consists of thin client layer (mobile devices), application layer (Web server) and database.

3.4. HTML5 mobile applications

HTML5 is specified by W3C to create a standard consisting of a set of features that can handle all the tasks that the current technologies (e.g. Adobe System Flash, Apple Quick Time and Java Oracle FX) are doing in a mobile Web apps. In the same way as mobile Web apps, HTML5 web app can have a three-tiered architecture where demanding processing can be carried out remotely on the Web servers. Additionally, HTML5 supports newer mobile technologies, such as Geolocation [10] and Scalable Vector Graphic [16]. The new features of HTML5 that benefits mobile Web apps include: Canvas [11], video tags [12], location-based services [13], working offline [14] and Web workers [15].

4. EVALUATION CRITERIA

In order to really assert which mobile application paradigms are better it is necessary to consider the viewpoints of all the involved players, namely developer, user and service provider. We deduced the evaluation criteria as follows:

- **Developer's viewpoint**
  - **Ease of developing:**
    - **Programming language:** This sub-criterion tells both about the simplicity and the popularity of the programming language.
    - **Software Development Kit (SDK):**
      - **Applicability**: This sub-criterion relates the applicability and accessibility of the SDK.
      - **Specifications and tips**: This sub-criterion is about the quality of documents accompanied with the SDK.
      - **Download installation and configuration**: These sub-criterion shows how straightforward to download install and configure the SDK on our development environment.
    - **Support from community**: This sub-criterion is about how much help developers could receive from community (e.g. tutorials, development tools and troubleshooting).
  - **Ease of coding:**
    - **IDE capability:**
      - **Code editor**: This sub- criterion is about the qualification of the IDE adopted for the app development. A good code IDE will provide immediate feedback with error messaged and warnings, quick fix feature, content assistant and a good guidance document.
      - **User interface builder**: This sub-criterion shows how robust the adopted user interface builder is. A powerful user interface builder will have instantly viewable and drag-and-drop capabilities, and other features reducing the workload of developers to create an attractive user interface.
    - **User interface**: This sub-criterion shows how straightforward it is to build an attractive and adaptive user interface.
    - **Device's interaction:**
      - **Hardware**: This sub-criterion describes how easy it is to access device's hardware, such as camera, storage and WLAN card.
      - **Built-in apps**: This sub-criterion shows how effective it is to adopt different built-in apps on devices, such as camera, phonebook and photo gallery apps
    - **Web service interaction**: This sub-criterion is to evaluate the ease of using a Web service, such as Google Map, Facebook and IQEngine.
  - **Ease of debugging**: This sub-criterion tells about how powerful and applicable the debugging tools are. A robust debugging tool will help manage break point, trace object value through the code and identify unexpected bugs.
  - **Ease of testing:**
    - **Emulator**: This sub-criterion show how effective is the testing of apps on an emulator. A powerful emulator will run fast and support as many device's features as possible, such as GPS, camera, and accelerator.
    - **Real device**: This sub-criterion show how easy is the testing of apps on a real device.
  - **Ease of deploying and updating**: This criterion relates how straightforward it is to deploy and update an app onto a real device.
  - **Ease of distribution:**
    - **Compatibility**: This sub-criterion shows how easy it is to distribute an app to multiple platforms.
    - **With app store**: This criterion shows how easy it is to distribute an app via app store.
    - **Without app store**: This criterion relates the possibility of distributing an app without the app store.

- **Application types:**
  - **Application using device's capability**: This sub-criterion evaluates the possibility of apps to use device's hardware such as camera, keypad and GPS.
  - **Application using server's capability:** This sub-criterion describes the possibility of apps to use server's capability such as storage and processing.
- **Powerful APIs and libraries**: This criterion shows the robustness and the popularity of the APIs and libraries that developers can use to build mobile apps.
- **Payment possibilities**: This criterion relates the possibility to earn revenues from the sale of app.

- **User's viewpoint**
  - **Ease of use:**
    - **Performance**: This sub-criterion presents the response time of the app in milliseconds. The shorter time we use the better the app performs.
    - **User interface**: This sub criterion evaluates how attractive, adaptive and responsive the app is.
    - **Operation**: This sub-criterion shows the ease of use of app. For example, it evaluates how to start the app and how to navigate between the functionalities of the app.
  - **Functionality**
    - **Working offline**: This sub-criterion evaluates the capability of working offline of the app.
    - **Accessing device's hardware**: This sub-criterion describes how effectively the app can access device's hardware (e.g. camera, GPS and storage).
  - **Installation and update**
    - **Compatibility**: This sub-criterion shows the compatibility of the app and how easy to install it on different mobile platforms.
    - **Downloading, installing and updating**: This sub-criterion shows the simplicity of downloading, installing and updating the app on mobile phones.
- **Service/content provider viewpoint**
  - **Content management**
    - **Content presentation**: This sub-criterion shows how complicated it could be to present the content on a mobile phone due to the limitation on screen's size and computing resource.
    - **Content delivering**: This sub-criterion evaluates the ease of delivering the content to mobile phones.
  - **Administration**
    - **Security**: This sub-criterion evaluates the effort of content provider to secure their service (e.g. authentication, confidentiality and integrity).
    - **Maintenance**: This sub-criterion shows the ease of hosting, managing, updating and maintaining the app.
  - **Distribution**: This sub-criterion shows how easily service/provider distributes their apps to end users.

5. EVALUATION AND RESULT

We evaluate the identified paradigms based on the described criteria in the previous section. Points from 1 to 5 are given for each criterion in which 1 is the lowest and 5 is the highest. The points are given according to the information collected from community of mobile app developers (e.g. Google code forum, stackoverflow, etc) and several scientific articles (IEEE, ACM, ScienceDirect, etc).

Table 1: Evaluation on mobile app paradigms

Viewpoints	Criteria	Native app	Web app	Widget	HTML5
Developer	Ease of developing	4.67	3.67	5	4.67
	Ease of coding	4	2	3	5
	Ease of debugging	4.33	3	5	3
	Ease of testing	4	1.67	3	4.33
	Ease of deploying and updating	3	3	3	3
	Ease of distribution	3.33	5	4.5	5
	Application types	5	3	3.67	4.33
	Powerful libraries and APIs	5	3	5	5
	Payment possibilities	3	5	3	5
	Average points	4.04	3.26	3.91	4.37
User	Ease of use	5	1.33	3.33	3.67
	Functionality	4.33	1.67	3.33	3.33
	Install and update	2	4.5	3	4
	Average points	3.78	2.5	3.22	3.67
Service provider	Content management	4	2.5	4	3
	Administration	3.5	3.5	3.5	3.5
	Distribution	2	5	3	5
	Average points	3.17	3.67	3.5	3.83

For developers, native apps and HTML5 are the best selection to build a mobile app. If developers want to make an app that requires accelerated graphic processing (e.g. high-end gaming apps), they should develop a native app. Only in that way can the app truly tap in processing powers and hardware features of the device. On the other hand, for more straightforward content driven service apps, HTML5 is preferred. W3C and other third parties are developing new APIs and libraries to make HTML5

more powerful and seamlessly capable of interacting with mobile devices in the same manner in which native apps do. Widgets are valuable for developers when they want to make a lightweight, single functional and portable app on mobile phone. Mobile Web app paradigm obtains the lowest grading because it is very complicated to build a functional and powerful mobile Web app.

For mobile users, native apps are very robust, responsive and usable to create the best user experience. HTML5 mobile apps are the second choice of mobile users since they are lightweight and cross-platform. HTML5 mobile apps are also functional and perform well on mobile phones and they are coming closer to native apps. Mobile widgets come third because they are lightweight and quite convenient to use. Users are not interested in mobile Web apps because they are slow, low functional and unattractive.

For service/content provider, HTML5 mobile apps are the best choice. Service providers can build an HTML5 mobile app once and distribute it everywhere. They can hence seamlessly deliver their content and service to mobile users. Mobile Web apps work in the same way and get the second position in the race. Mobile widgets and native apps have the lowest grading because of platform fragmentation. Service provider must deploy widgets and native apps onto every device. It is a complicated process that considerably increases the cost and effort of service/content providers.

6. VERIFICATION

In order to verify the evaluation carried out in the previous section, we will build a **mobile object recognition/visual search app** using the two most promising paradigms, namely native app and HTML5 mobile app. The objective of the practical verification is to ensure that the performed evaluation is conformed to the reality and consequently usable. The same app will be developed using both paradigms and evaluated according to developer and user's viewpoints. Comparisons with the former evaluation will then be carried out.

The object recognition app in itself is a fascinating and useful app which provide to users information about any requested objects such as a glass, a car, a person to a building, a monument or a mountain. The app will make use all capability of devices and remote functionality provided by a Web service provider.

Figure 1 describes our technology architecture implementation. The frontend app running on the device is a native app, an HTML5 app or a PhoneGap app.

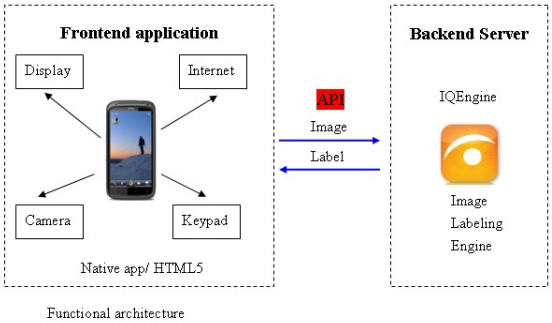

Figure 1: Technology architecture

The reason we choose IQEngine for the backend service is that IQEngine API is very usable, and the object recognition is quite fast (up to 1 minute per object). Moreover, we receive a lot of support from IQEngine developer center such as tutorials, guideline and troubleshooting, and 1000 free visual scans from the service. We can also create our own training database to reduce the time of visual search, which is an advantage over the other online visual discovery engines.

The frontend apps can access the camera and keypad, connect to the Internet and display the label on device screen. The user can capture image of unknown objects by using device camera and send the image and other parameters (e.g. the timestamp of the image, the API key and the signature created by using HMAC-SHA1 hashing algorithm) to IQEngine server through the available APIs. The server subsequently analyzes, compares the image with the images in its image database, labels the objects in the image and responds the device. Finally, the frontend app will display the label on device's screen. The mobile phones adopted for the implementation is the Samsung Galaxy GTI-5500 with Android 2.1, iPhone 4 and HTC Desire HD with Android 2.3.

6.1. Native application

We develop the app using native app paradigm and deploy it onto Android platform. The IDE to code the app is Eclipse and the programming language is Java. Figure 2 show the user interface of the native app when a user capturing an image and receiving the result after all.

Figure 2: User interface of mobile recognition app

The app's architecture has five main classes, including *HomeInterface*, *CameraView*, *AndroidExplorer*, *Inquiry* and *IQEngine* classes. *HomeInterface* defines the home interface of the app. *CameraView* can be considered as the client for the Camera service, which manages the actual camera hardware. *AndroidExplorer* helps users select the available image file to upload to IQEngine server. *Inquiry* initiates an API object using API key and secret, queries the image, retrieves the result in JSON format and decode JSON object into String object. *IQEngine* creates the package, (e.g. the timestamp, image, API key and API signature) and sends the request to IQEngine server.

6.2. HTML5 application

We create an object recognition app by using HTML5 mobile app paradigm. Similar to the native app discussed in the previous section, our HTML5 app can interact with device's hardware and IQEngine server. The difference between the two is that users load the app onto their device and run in on browser. We use Aptana Studio as the IDE for coding the app and Opera

Dragonfly for debugging and testing the code, and 000Webhost service to host the app.

The architecture of the app hosted on 000WebHost.com includes HTML5 and JavaScript, PHP script and jQuery Mobile files. HTML5 provides us with the *<video>* element and *navigator.getUsermedia()* to use device's camera, and the *<input>* element to create the file dialog to access file storage. We also adopt several interface elements and features of jQuery Mobile such as of pages within pages, Ajax navigation, page transition, orientation on change and theming, releasing our headache to make the app look like a native app.

Unlike the native app, the device will interact with the app on 000WebHost server rather than directly communicating with IQEngine server. Users will receive the label after all as shown in Figure 3.

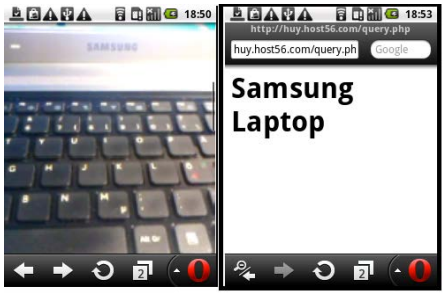

Figure 3: Users retrieve label from IQEngine server

6.3. PhoneGap application

The paradigm we employ in this section to build object recognition app is still HTML5 mobile app but we name it as PhoneGap app to distinguish it with HTML5 app. We create the app by using HTML5, CSS3 and JavaScript but we do not use *<video>* and *getUserMedia* API to access device's camera. Instead, we wrap the app with PhoneGap framework to enable the app to use the built-in apps (e.g. camera and photo gallery) and deploy the app to multiple platforms (e.g. Android, iOS, Windows Phone 7 and WebOS). We use Eclipse as the IDE and PhoneGap as the SDK to develop the PhoneGap app.

The app architecture includes HTML5, CSS3 and JavaScript files which stay on **mobile phone**, and PHP script which is hosted on 000WebHost server. The CSS3 will help to design the page with a range of media types such as *orientation*, *min-width* and *max-width* to make the app target different mobile devices with different screen's size and orientation.

From home interface users can capture an image or select an available image in their photo gallery. Then they call PHP script to upload the captured/chosen image onto 000WebHost server and calling IQEngine API to recognize the object as in Figure 4.

Figure 4: Users receive the result from IQEngine server

6.4. Result

The result in Table 2 is same that from the former evaluation. We find the grading for HTML5 mobile app paradigm by calculating the average points of the HTML5 app and PhoneGap app for every criterion.

Developers prefer HTML5 mobile app to native app paradigm even though there is sufficient support to hook into device's hardware features from both paradigms. The reason is that the HTML5/ PhoneGap code to access device's hardware and Web service is less verbose and complicated than Android Java code. Building an adaptive user interface on HTML5 mobile apps is also much more straightforward than on native apps. Furthermore, the capability of "write once and deploy many" is an advantage of HTML5/PhoneGap over the native app paradigm.

For users, it is understandable that native app is more preferable than the HTML5 mobile app paradigm. The native app is very robust and convenient to use while the HTML5/PhoneGap app has several limitations. The usability and the performance of the HTML5 app are lower than the native app. We perform the tests to measure the performance of the native app, HTML5 app and

Table 2: Evaluation on native app, HTML5 app and PhoneGap app

Viewpoint	Criteria	Native app	HTML5	PhoneGap	HTML5 mobile app
Developer	Ease of developing	4.83	4.83	4.83	4.83
	Ease of coding	3.21	4.23	4.17	4.2
	Ease of debugging	2	3	1.5	2.25
	Ease of testing	3	3	2	2.5
	Ease of deploying and updating	5	3	3.33	3.17
	Ease of distribution	3	3	3.56	3.28
	Application types	4.5	5	5	5
	Powerful libraries and APIs	5	5	5	5
	Payment possibilities	3	5	5	5
	Average points	3.73	4.01	3.82	3.91
User	Ease of use	4.67	3	4.67	3.84
	Functionality	3.67	3.33	4	3.67
	Installation and update	2.5	3	3	3
	Average points	3.61	3.11	3.89	3.5

PhoneGap app on Samsung Galaxy GTI-5500 phone. Figure 5 shows the time needed to use the apps in comparison.

We create three timers and integrate them into the apps to measure the time needed to use the apps in milliseconds. The timer created by using Android Java code is for calculating the time using the native app and the others are in JavaScript to measure the HTML5 mobile app and the PhoneGap app. We get the system time of starting the apps and the system time of receiving the result from the IQEngine server for each test. The native app requires the least amount of time to use. Meanwhile, it takes longer to use the HTML5 app than the native app and the PhoneGap app. The reason is that we must run Opera browser and load the app onto the browser before using it. PhoneGap app takes longer to use than the native app because it takes much more time to start the camera and capture a picture.

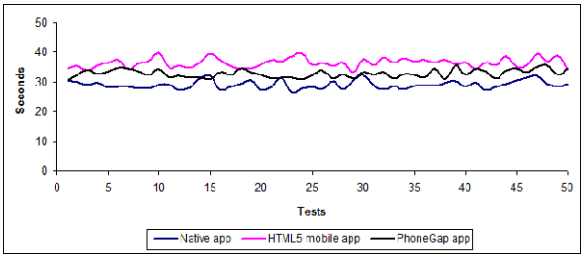

Figure 5: The amount of time to use the native app, HTML5 mobile app and PhoneGap app in comparison

6.5. Key finding and recommendation

W3C announces that HTML5 is a cross-platform solution that can hook into device's hardware (e.g. camera) to create a powerful mobile app. In fact, developers face a fragmentation in mobile platforms and browsers to build an HTML5 app to access hardware. For example, HTML media capture is the first API in 2011 to standardize media capture on Web. It works by overloading the *<input type="file">* to capture snapshot with device's camera. However, the API is too limited to use and only works on Android 3.0 browser. Therefore, we adopt the implementation of *getUserMedia* API belonging to W3C WEBRTC (Web Real-Time Communication) working group to access device's camera. However, only Google and Opera currently have developer builds that include this API, and only Opera Lab on Android supports developers to use it in their apps.

Native apps are well-known for their fast and responsive user interface, and the seamless capability to access hardware features. However, creating such apps is complicated and requires much effort from developers on any platform. Meanwhile, jQuery Mobile simplifies the code to build an attractive and adaptive user interface for HTML5 mobile apps. PhoneGap framework also lets developers make the HTML5 app access built-in apps and run on multiple platforms effectively. PhoneGap uses the same native APIs with native apps but abstract them so that developers can simply write apps in HTML and JavaScript. Therefore, we claim that the most effective solution now to build the apps that are capable of using device hardware (GPS, accelerator and camera) and working cross-platform is to wrap HTML5, jQuery Mobile and CSS3 with PhoneGap framework. However, PhoneGap apps cannot replace native apps because they perform slower than native apps due to the overhead from an abstraction and HTML render in addition to the time to execute the native processes.

7. CONCLUSION

The goal of our work is to give some guidelines about which mobile app paradigms are more suitable than other. After the analysis and evaluation, we conclude that native apps and HTML5 mobile apps keep their first places in the race of mobile paradigms. Mobile widgets are still valuable but their role is no longer so important on mobile devices. Mobile Web apps will become a history and they are soon replaced by HTML5 mobile apps. Although we strived to ensure objectivity it is worth noting that the results of our work are only indicative because the ultimate choice of a mobile app paradigm will always relies on the taste of the developer and the context of the application

8. REFERENCES

[1] A. Charland and B. Loux, "Mobile application development: web vs. native," *Commun. ACM,* vol. 54, no. 5, pp. 49-53, May 2011.

[2] R. Padley, "HTML5-bridging the mobile platform gap: mobile technologies in scholarly communication," *Serials*, vol. 24, pp. 32-39, 2011.

[3] M. Forgue and D. Hazaël-Massieux, "Mobile web applications: bringing mobile apps and web together," In *Proceedings of the 21st international conference companion on World Wide Web* (WWW '12 Companion), ACM. Newyork. USA, pp. 255-258, 2012.

[4] B. Zhang, T. Xu, W. Wang and X. Jia, "Research and implementation of cross-platform development of mobile widget," *Communication Software and Networks (ICCSN), 2011 IEEE conference* , vol., no., pp.146-150, 27-29 May 2011.

[5] T. Mikkonen and A. Taivalsaari, "Apps vs. Open Web: The Battle of the Decade," In *Proceedings of the 2nd Workshop on Software Engineering for Mobile Application Development* (MSE'2011,Santa Monica, California, USA) , pp. 22-26, October 27, 2011.

[6] S. Tarkoma, and E. Lagerspetz , "Arching over the Mobile Computing Chasm: Platforms and Runtimes," *Computer* , vol.44, no.4, pp.22-28, April 2011.

[7] S. Tarkoma, Ed. Mobile Middleware-Architectures, Patterns, and Practice, Wiley, pp. 2-3, 2009.

[8] C. Raibulet and D. Cammareri, "Automatic generation of mobile widgets", *International Journal of Pervasive Computing and Communications*, vol. 7, no. 2, pp.132 – 146.

[9] S. Helal, J. Hammer, J. Zhang, A. Khushraj , "A three-tier architecture for ubiquitous data access," *Computer Systems and Applications, ACS/IEEE International Conference on. 2001* , vol., no., pp.177-180, 2001.

[10] B. Pejiü, A. Pejiü and Z. Čović , "Uses of W3C's Geolocation API," *Computational Intelligence and Informatics (CINTI), 2010 11th International Symposium on* , vol., no., pp.319-322, 18-20 Nov. 2010.

[11] S.J. Vaughan-Nichols, "Will HTML 5 Restandardize the Web?," *Computer* , vol.43, no.4, pp.13-15, April 2010.

[12] S. Pfeiffer and C. Parker, " Accessibility for the HTML5 <video> element,". In *Proceeding of 6th International Cross-Disciplinary Conference on Web Accessibililty* (W4A '09), *ACM. USA*, pp. 98-100, 2009

[13] Y. Liu and E. Wilde, " Personalized location-based services," In *Proceedings of the 2011 iConference* (iConference '11), ACM. New York. USA, pp. 496-502, 2011.

[14] P. Lubbers, B. Albers and F. Salim, *Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development*, pp. 243-257, Apress, 2010.

[15] F. Reynolds, "Web 2.0–In Your Hand," *Pervasive Computing, IEEE* , vol.8, no.1, pp.86-88, Jan-March 2009.

[16] A. Quint, "Scalable vector graphics," *Multimedia, IEEE* , vol.10, no.3, pp. 99- 102, July-Sept. 2003.

[17] W3C working draft. getUserMedia: Getting access to local devices that can generate multimedia stream. [Online] [Cited: May 2012] http://dev.w3.org/2011/webrtc/editor/getusermedia.html.