An Automation Component for Cross-Platform, Context-Aware Applications Development

Achilleas P. Achilleos(✉), Marita Thoma, Georgia M. Kapitsaki, Christos Mettouris, and George A. Papadopoulos

Department of Computer Science, University of Cyprus,
1 University Avenue, Nicosia, Cyprus
{achilleas,mthoma03,gkapi,mettour,george}@cs.ucy.ac.cy
http://www.cs.ucy.ac.cy

Abstract. Context-aware computing faces many challenges mainly due to the increasing number and heterogeneity of context sources, since the Internet of Things introduces billions of devices. The development of context-aware applications is thus becoming a complex and cumbersome process, which is also augmented by the availability of different mobile platforms. This requires a modular approach that aims to automate the development of these applications, by enabling developers to easily add context-aware functionality. In this paper, an automation component is presented that allows novice developers to select context plug-ins (e.g., Geolocation, Facebook profile, battery level) and generate a sample application that includes these context-aware functions. This application serves as a basis for the development of more complex cross-platform, context-aware applications. The code generation support of the automation component is demonstrated through a case study. Finally, a basic evaluation is performed to showcase the benefits, issues and identify potential future work.

Keywords: Context-aware applications · Separation of concerns · Mobile computing · Web development · Code generation

1 Introduction

The widespread use of mobile platforms has led to new business models and has urged organizations to provide applications for users "on the move". These mobile users are best served if their needs of mobility are fulfilled in all settings without overwhelming users with redundant information. Such a personalized technology view has been adopted by various researchers [1] and is currently intensified by the widespread presence of sensors in mobile devices, such as GPS receivers, accelerometers and compasses, as well as the vast volume of data available on the Internet in the form of Web Services, information from social networks and sensors and actuators typically connected to micro-controllers. This necessity to follow user needs to utilize efficiently software applications and services using different mobile devices at various places forms part of the general

H. Christiansen et al. (Eds.): CONTEXT 2015, LNAI 9405, pp. 63–76, 2015.
DOI: 10.1007/978-3-319-25591-0_5

term of context-awareness. Context-awareness defines a broad concept that is generally used to describe the process of acquiring and managing different pieces of context information to intelligently adapt the application behaviour. We adopt the definition from Dey and Abowd [2], referring to context as *"any information that can be used to characterize the situation of an entity, in which the entity can be a person, a place, or a physical or computational object that is considered relevant to the interaction between the entity and the application"*.

Mobile computing combined with context-awareness encompass various aspects spanning from sensing information at the hardware and network information level, to context-based recommendations at the application level, such as travel or music recommendations for mobile environments [3]. Context-acquisition from device components (e.g., device motion, battery level), from social networks (e.g., LinkedIn, Facebook), network information (e.g., based on Cell ID, or Wi-Fi), etc. can support and facilitate the mobile user in a variety of tasks. In fact, this information act as enablers of context-awareness, empowering applications to be adapted to end-user preferences and circumstances.

Although mobile devices offer clear-cut benefits to the user and context-awareness is a desirable feature, the diversity of platforms makes development of platform specific applications an uneconomical choice, since it requires manpower and additional resources from the provider's side to develop for each platform. The problem of platforms diversity needs to be addressed, since is important by developers to provide cross-platform applications, because even the same users own and use different devices.

Currently, a useful alternative to native applications is offered through the HTML5 standard, which allows accessing many device resources that were unavailable to web technologies in the past. This vision of pure web-based development offers many benefits to software engineers, where the key one refers to cross-platform development: develop once, deploy anywhere. Also recent developer surveys demonstrate a tendency to move towards pure HTML5 solutions[1], as mobile browsers implement the features offered by the HTML5 standard. Gartner also defines[2]: *"HTML5 is in the top 10 technologies and capabilities for 2015-16 and despite many challenges, HTML5 will be an essential technology for organizations delivering applications across multiple platforms"*.

An HTML5, context middleware was developed [4], which provides a modular approach that promotes the concept of separation of concerns and thus enables developers to reuse context plug-ins for the development of context-aware applications. This work develops a new component that assists developers with limited experience in developing context-aware applications, by automating a large part of the process. It allows selecting from an (extensible) pool of context plug-ins and generates the code for a sample context-aware application that can serve as the basis for further development. The key target is to reduce the complexity involved with developing context-aware functionality, simply by

[1] http://www.sencha.com/blog/the-state-of-html5-development-in-the-enterprise/ - Published: February 12, 2014.

[2] http://www.gartner.com/newsroom/id/2669915 - Published: February 24, 2014.

allowing developers to easily select and include context plug-ins that already provide this functionality. A further benefit is found in the production of compact code, having clearer separation of concerns (focus on application logic instead of context-aware functions).

The rest of the paper is structured as follows. Section 2 introduces related work, outlining the difference of our approach in comparison to existing frameworks. Section 3 provides an overview of the context-middleware. The following section introduces the developed automation component and the development pathways. Section 5 demonstrates the use of the automation component in two scenarios of a case study and presents the results of the evaluation performed. The final section outlines the conclusions and identifies potential future work.

2 Related Work

The provision of access to context sources to facilitate the development of mobile applications, has been the focus in research work. MUSIC (Self-Adapting Applications for Mobile Users in Ubiquitous Computing Environments) defines a context-management and adaptation middleware for Java [5]. MUSIC runs on top of the OSGi (Open Service Gateway initiative) framework and includes a number of context plug-ins available as OSGi components. Similarly to our solution, MUSIC includes the notion of context sensor and reasoner plug-ins, which receive and process context data from other plug-ins via the middleware in order to produce higher level data. The evolution of MUSIC, following the same rationale, but tailored to the Android platform is found in RSCM (Really-Simple Context Middleware) [6]. The RSCM architecture differentiates between context producing components (the plug-ins) and context consuming components (the applications). RSCM is used in the Professor2Student application that offers dynamic collaboration between supervisors and students [7].

A similar solution, which also focuses on web applications accessing sensor information on the Android platform, has been proposed in Ambient Dynamix [8]. Sensor access is handled through native plug-ins that can be installed on-demand on the users Android device, and are organized and managed by the Dynamix Service. Each plug-in provides its own set of context events and (optionally) an API for controlling its functionality. To support communication with web applications, Dynamix exposes two REST APIs using a customized web server embedded within the Dynamix Service. This way a web-based application is able to interact with the associated Android plug-in to trigger context sensing and/or remote device interactions.

Dynamix has characteristics that are closer to a “hybrid” approach (it does not follow a pure cross-platform approach), but it rather ties web application development to the native context plug-ins offered by the underlying Android platform. The use of hybrid technologies is very popular in mobile application development in an attempt to satisfy the cross-platform requirement. A variety of such technologies exists, such as PhoneGap, Apache Cordova (i.e., the open-source engine/version that runs PhoneGap) and AppBuilder. Hybrid technologies offer usually a set of uniform JavaScript libraries that can be invoked,

Table 1. Comparing HTML5 properties with hybrid and native frameworks

Feature	H5CM approach	Hybrid Technologies approach	Platform-specific frameworks approach
Native code use	x	✓ [PhoneGap]	✓ [RSCM, Dynamix]
Technologies used	Web Technologies	Web along with platform-specific, e.g., Android, Firefox OS, iOS [PhoneGap]	Web [Dynamix] and Android SDK [RSCM, Dynamix]
Required development knowledge	Web Technologies	Web technologies along with basic framework/platform specific understanding [PhoneGap]	Web technologies and platform specific [Dynamix)]or Platform specific [RSCM]
Pre-installation requirements	x	✓ [PhoneGap]	✓ [RSCM, Dynamix]
Access Device-specific features	Limited (Browser restricted)	Full (based on API availability per platform) [PhoneGap]	Full (based on plug-in availability) [RSCM, Dynamix]
Security and Privacy support	Browser based	Guidelines only [PhoneGap]	Via context firewall [Dynamix]

wrapping device-specific native backing code through provided JavaScript libraries. This process provides access to native device functions through JavaScript, such as the device camera or its accelerometer.

With the introduction of H5CM (HTML5 Context Middleware) [4], the vision is a pure web-based approach. Table 1 captures the key points of variability between our approach and other approaches. All cases have both advantages and disadvantages in terms of application variability and execution speed. The main criticisms for hybrid development is the learning curve, since developers need to learn how to use the native libraries for each platform, but most importantly that mobile devices are not able to smoothly run a hybrid application [9]. On the other hand, native applications offer benefits in terms of performance and API coverage, but lack in terms of instant worldwide deployment, manual installation or upgrades and flexibility to combine data from different resources [10].

Differentiating from both native and hybrid approaches, H5CM offers a pure HTML5 approach not bound to any platform or development environment employing solely web technologies [11]. Although access to device-specific capabilities is provided based on the browser support, vendors are continuously extending their support based on the popularity and evolution of HTML5. Moreover, in respect to pre-installation requirements our approach does not have

any prerequisites. Finally, a feature that is missing from all approaches, is an extended security and privacy support, since this feature is handled either by the browser (as in the case of H5CM) or by the underlying platform (in hybrid and platform specific frameworks).

3 H5CM Overview

3.1 Architectural Elements

As aforementioned the main requirements fulfilled with the creation of H5CM were to provide a framework that is modular, reusable, extensible, and that can be utilised for web applications on any mobile platform. H5CM has a hierarchical structure: at the lower level there are context-sensor plug-ins that allow acquiring and distributing low-level context data. These may refer to the location of the user, the orientation of the user's device and the results from the invocation of Web APIs. At the second level of the hierarchy there are context-reasoner plug-ins that accept low-level context from one or more sensor plug-ins and apply the appropriate reasoning, in order to create high-level context information. The application is at the top of the hierarchy and is able to communicate with the sensor and/or reasoner plug-ins to acquire context information that enables the adaptation of the application's logic. More details on the architecture of H5CM can be found in our earlier work [4].

3.2 The Context Repository

The H5CM functionality is empowered by an extensible and reusable *Context Repository*. The basic point of differentiation between sensor and reasoner plug-ins, is that the former provides access to "basic" context data in the form that these can be collected directly from context sources (e.g., device accelerometer, user geographical coordinates), whereas the latter gives access to sophisticated context information that are derived from the "basic" data. These plug-ins define an extensible and reusable repository. On the one hand, they can be reused by developers, since they are generic and can be invoked from any context-aware, web application. On the other hand, the set of plug-ins can be extended by technical users that need additional functionality, as more and more features of HTML5 are continuously being supported in mobile browsers. H5CM is currently offering different reusable plug-ins, some of which are described in Table 2. The current version of H5CM and the plug-ins are available on Google Code[3].

The context plug-ins repository has been also enriched with the addition of the SensoMan plug-in. This plug-in enables access to context data retrieved from sensors connected to Arduino micro-controller boards. The SensoMan plug-in is of particular importance, since it provides access to the context data coming mainly from the environment (see Fig. 1), which was not fully covered by other plug-ins that principally retrieve data from the mobile device (internal) and the

[3] https://code.google.com/p/h5cm/.

Table 2. The context repository: example sensor and reasoner plug-ins

Plug-in	Plug-in type	Functionality	Access type
SensoMan	Sensor	Enables connecting and retrieving data from sensors connected to microcontroller boards through the REST API provided by the SensoMan system [16].	External: Micro-Controllers
BatteryLevel	Sensor	Retrieves and monitors the battery level (e.g., 74 %).	Internal: Mobile Device
Geolocation	Sensor	Allows detecting and continues monitoring the position of the user. Returns the location in the form of GPS coordinates (i.e., latitude and longitude).	Internal: Mobile Device GPS Receiver
DeviceOrientation	Sensor	Monitors the physical orientation of the device (e.g., the user tilts or rotates it).	Internal: Mobile Device
RestfulService	Sensor	Allows connecting to RESTful services. It requires as a parameter the URL of the service including any parameters the service may require for its functionality.	External: Internet
FacebookConnect & LinkedInConnect	Sensor	Enables user authentication with Facebook/Linkedin and requests the user to grant access permissions for retrieving data (given in comma separated string values).	External: Social Networks
FacebookInformation & LinkedinInformation	Sensor	Provides the means to acquire Facebook/LinkedIn profile data of the user (i.e., public user data) and retrieving additional data provided that the user is authenticated (requires FacebookConnect/LinkedInConnect).	External: Social Networks
FacebookPosts	Sensor	Provides the way to obtain wall posts of the user, provided that the user is authenticated (requires FacebookConnect).	External: Social Networks
ActivityRecognizer	Reasoner	Recognizes the activity of the user (Jogging, Walking, Sitting, Upstairs or Downstairs) based on the results of a decision tree classifier.	Internal: Mobile Device
BatteryAnalyzer	Reasoner	Retrieves data from the existing BatteryLevel sensor and returns TRUE if the application is able to handle computational intensive tasks based on a developer specified cutoff value (e.g., ¿60 %) passed as a parameter to the reasoner or FALSE otherwise.	Internal: Mobile Device
FacebookRestaurants	Reasoner	Retrieves data from two sources: Google Places restaurants at a specific area and Facebook posts about user-visited restaurants. It computes the intersection of the two sets in order to give user preferred restaurants (requires FacebookConnect and FacebookPosts).	External: Social Networks

user (social networks). The SensoMan plug-in enables access to a diversity of sensors and delivers context-aware functions that further support the development of context-aware Web applications. The list of sensors currently supported by the SensoMan system is presented in [16].

Furthermore, reasoner plug-ins provide the capability to perform aggregation, analysis and reasoning on "basic" context data to derive higher level information that can be useful in taking proactive actions at the application level. Hence, a simple reasoner of H5CM can include the use of information on whether the device is charging and its acceleration to decide whether the user is walking, driving a car or sitting in a room. Such mechanism can be combined with machine learning techniques of clustering or classification for drawing useful conclusions on the user, such as activity recognition addressed in previous works [12]. Such techniques can be supported by H5CM through plug-ins that include advanced processing of context information.

In that respect we have implemented the ActivityRecognizer reasoner plug-in that performs activity recognition based on training performed over raw accelerometer data obtained from the raw dataset[4] presented in [13]. The dataset includes information collected from 29 users while performing various daily activities, such as Jogging, Walking, Sitting, Upstairs or Downstairs. We have used the dataset that includes these specific activities to train a C4.5 Decision tree classifier with a confidence factor of 0.25 [14]. For this step of the process we have employed the Weka machine learning software and its decision tree implementation indicated as J48 [15]. Subsequently, a pruned version of the tree that was created from the classification process was transferred to JavaScript code and was used in the formation of a new reasoner plug-in. The current version of the ActivityRecognizer is available on the Google code website of H5CM.

4 H5CM Automation Component

The main contribution of this work is the definition and implementation of the H5CM Automation Component (HAC), which refines and extends the middleware architecture. In specific, it automates the development process, so as to support mainly developers that are not experts in the implementation of context-aware applications. In this way a developer can easily kick-start the implementation of context-aware, web-based applications using the concepts and the reusable elements provided by the middleware. This section presents the HAC component that was developed through refinement of the H5CM architecture. The refined architecture presented in Fig. 1 enables storing all developed plug-ins in an XML-based repository, which is queried by the HAC component to identify the complete list of plug-ins that the developer can use to automatically generate the new (sample) context-aware application.

In specific, the HAC component gives the ability to the developer to select which sensor and reasoner plug-ins need to be included and used in the context-aware application to be developed. Apart from selecting the plug-ins, the HAC

[4] http://www.cis.fordham.edu/wisdm/dataset.php.

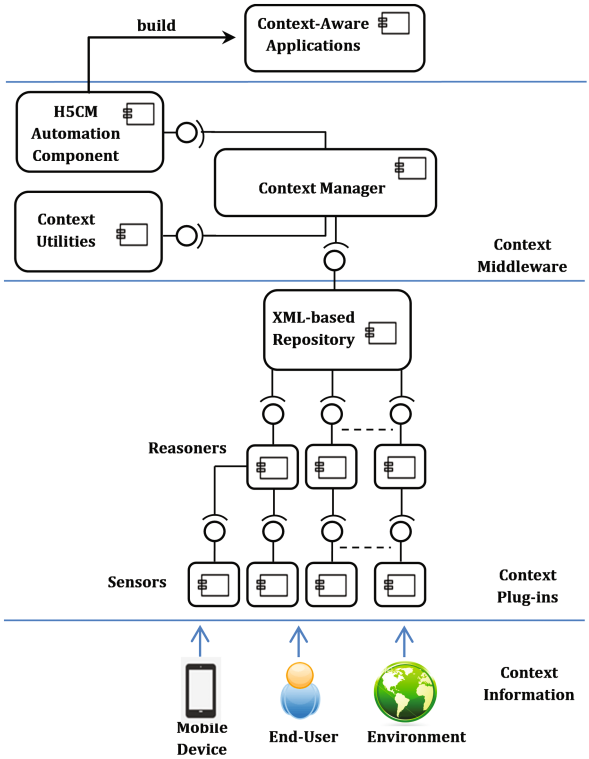

Fig. 1. H5CM Automation Component (HAC): the extended H5CM middleware.

component also queries these plug-ins so as to identify their context properties (e.g., name, birthday from FacebookInformation plug-in). This enables the developer to choose the properties needed by the application, in order to generate the code that composes the new context-aware application. A novice developer of context-aware applications, is thus able to easily include context-aware functionality in the application, simply by selecting the necessary plug-ins and properties. This allows the developer to focus on the implementation of the application logic and UIs, rather than on retrieving and analysing context information, e.g., GPS coordinates, Facebook profile, which is indeed a complex task.

The HAC component is implemented using the library JQuery Steps[5] In specific, JQuery Steps is a User Interface (UI) library that enables the developer to easily create wizard-like interfaces. The library basically groups content into sections for a more structured and orderly page view, while providing the capability for validation of steps and the information provided, so as to ensure smooth progress and quality of the generated code produced by the HAC component.

[5] GitHub wiki Online: http://www.jquery-steps.com/.

When a developer navigates to the first page, all available plug-ins and their properties, including an informative description in terms of the functionality, are queried, generated and presented in a visual form. The description of the plug-ins and their properties is defined in an XML file named "all-plugins.xml" (see description of Facebook plug-in in Listing 1.1), which forms the XML representation of the context repository. The HAC component was implemented using this modular and extensible approach, so as to adhere to the architecture of the H5CM. This allows any expert developer to implement a new plug-in and add it directly to the repository, simply by describing it using XML. In fact, the HAC component can query and visualise any newly implemented plug-in, and include its functionality as part of the generated application, simply by describing the new plug-in similarly to the example Facebook plug-in shown in Listing 1.1.

Furthermore, the HAC component provides the capability to the developer to select from two development pathways. The first development pathway, showcased in the form of a UML Activity diagram in Fig. 2 (A), allows developers to enter the application name, select all context plug-ins and their properties, and generate directly the new context-aware application. The generated application code can be downloaded and the application is executed and demo-ed before quitting the application. Using the generated code the developer can further modify the generated application.

Listing 1.1. XML Description of Facebook plug-in and properties.

```
<plugin>
    <name>FacebookConnect</name>
    <sources>
        <source>modules/sensors/FacebookConnect.js</source>
    </sources>
    <description>This plug-in is used to allow the user to access
        through a Facebook account </description>
    <properties>
        <property>
            <name>name</name>
            <description>This property retrieves the name of the
                Facebook user</description>
        </property>
        <property>
            <name>email</name>
            <description>This property retrieves the email of the
                Facebook user</description>
        </property>
        <property>
            <name>gender</name>
            <description>This property retrieves the gender of the
                Facebook user</description>
        </property>
        <property>
            <name>username</name>
            <description>This property retrieves the username of the
                Facebook user</description>
        </property>
        <property>
            <name>birthday</name>
            <description>This property retrieves the information about
                the birthday of the Facebook user</description>
        </property>
    </properties>
</plugin>
```

The second development pathway enables the developer to enter the name of the application and then select specific plug-ins. Afterwards, the developer is able to choose context properties available in specific context plug-ins (e.g., FacebookConnect, LinkedInConnect), which are required as context information in the context-aware application to be developed. For instance, as presented in

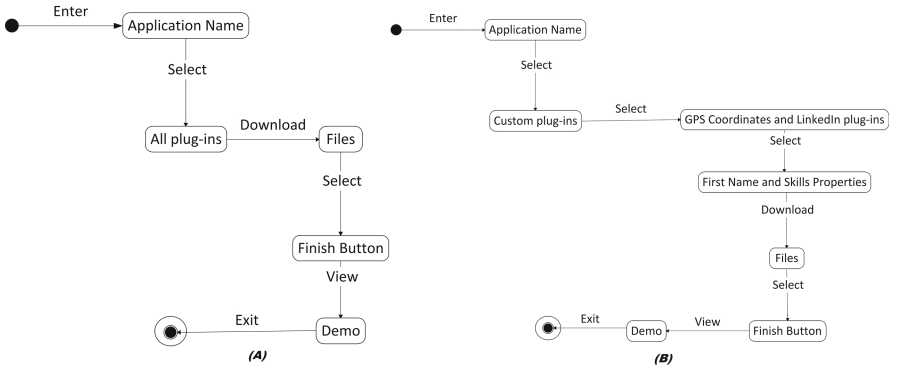

Fig. 2. UML activity diagram: the development pathways.

Fig. 2 (B), the developer has selected the Geolocation, LinkedInConnect and LinkedInInformation plug-ins. In the following step the developer chooses the first name and skills context properties associated with the LinkedIn related plug-ins. Finally, the developer is able to download the generated code and view a demo of the application prior to exiting. The developer can use and further modify the generated application based on the desired logic and UI for the final context-aware application.

5 Case Study and Evaluation of the HAC Component

This section describes the use of the HAC component for the generation of two example context-aware applications and the evaluation of the component. The evaluation is contacted by nine MSc students of the course EPL603: Advanced Software Engineering, offered at the Dept. of Computer Science, University of Cyprus. In the first scenario, all context plug-ins offered by the middleware are selected by the developer and the new context-aware application is generated. It is important to note that the SensoMan sensor plug-in and the ActivityRecognizer reasoner plug-in are still to be integrated with the HAC component, since their development was only recently completed [16]. The automatically generated application for this scenario where all the plug-ins are selected and executed using the desktop version of the Firefox Browser.

In the second scenario, the development pathway enables developers to select specific plug-ins and their associated properties. As presented in Fig. 3, during the third step the HAC component allows selecting the desired context-plug-ins. In the next step, the HAC component detects that the FacebookConnect plug-in has explicit context properties. This allows the developer to choose the necessary ones for supporting the desired functionality for the context-aware application. At the "Finish" step the summary of the process is presented to the developer, and as soon as the developer downloads the code files of the generated application, the wizard is completed. The final snapshot in Fig. 3, showcases the application running on the Android Firefox Mobile Browser.

Fig. 3. Scenario 2: selecting context plug-ins and context properties.

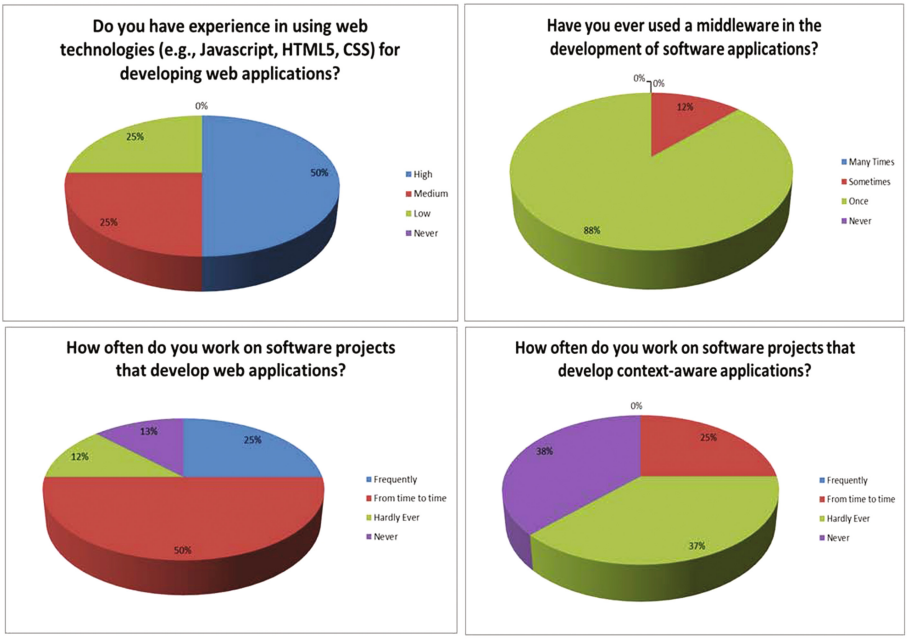

Fig. 4. Knowledge, experience and expertise on web technologies and context-aware applications development.

The HAC component was used by nine MSc students of the EPL603 course: Advanced Software Engineering. At first the evaluation gathered information on the knowledge, experience and expertise of the students in regards to prior usage of middleware technologies, use of web technologies and development of context-aware applications. As illustrated in Fig. 4, students have at 75 %, medium to high experience in the use of web technologies (e.g., Javascript, HTML5, CSS) for developing web applications. In terms of the use of a middleware for the development of software applications, 88 % indicated that they have only used such software once, which refers to the use of the first version of the middleware during the first evaluation session [4], while only 12 % have used a middleware sometimes. The third chart confirms that 75 % of the students have experience and have worked in the past on software projects for the development of web applications. Finally, in respect to the development of context-aware applications 3 out of 4 students indicated that they almost never developed such applications.

The students answered the provided questionnaire[6], which revealed that the opinion of the students for the HAC component was highly positive. In specific, the scores were high in terms of all the assessed factors: *Learnability, Memorability, Effectiveness and Functionality.* Moreover, reviewing the answers to the

[6] https://docs.google.com/forms/d/1DJ7FNarlpAq0FI6bqcAOsOb703NgNbRctTsSFUxhBgU/viewform, HAC Component Questionnaire.

Table 3. Evaluation results

Learnability, Memorability	Effectiveness	Functionality
4.78/5	4.63/5	4.63/5

questions dedicated to the HAC component it is evident that the students considered it as a strong point/extension for the middleware. Finally, the automation tool raised the interest of students during the evaluation session, since they actively tried it on different mobile devices to identify and assess if it was working for mobile-specific plug-ins (e.g., device motion, device orientation) and on different platforms. The results of the tests were encouraging.

The last two factors scored lower since the students indicated that they would appreciate access to additional sensors, which provided context data from the environment. The SensoMan plug-in can assist to overcome this limitation and as part of future work we will be adding this plug-in in the HAC component, so as to offer additional context-aware functions and increase effectiveness of the generated applications. Moreover, a complementary evaluation by expert developers can definitely provide additional insights in terms of functionality coverage and effectiveness in building context-aware applications. The evaluation was intentionally performed with novice developers (in terms of developing context-aware applications), since the component's key aim is to support developers that have limited experience with such kind of applications (Table 3).

6 Conclusions

Context awareness is a promising research field that refers to the acquisition, processing and reasoning on context information, so as to adapt the application mainly to the requirements of the user. Due to the diversity of context sources (i.e., mobile device, user, environment) the development of context-aware applications becomes inherently complex. The developed H5CM context middleware provides the ability to reduce complexity through the concept of separation of concerns. In specific, the middleware enables reusability of context plug-ins and favours extensibility for supporting the development of context-aware applications. Furthermore, the combination of HTML5 and context-awareness in the developed middleware addresses the issue of cross-platform development.

The work in this paper goes a step further and aims at further simplifying the development of context-aware applications. In specific, it extends the H5CM middleware by defining the HTML5 Automation Component. The HAC component is also implemented following a modular and extensible approach, which enables novice developers to define and automatically generate web-based, context-aware mobile applications. Future work aims to integrate the SensoMan context sensor and ActivityRecogniser context reasoner plug-ins with the HAC component, as there plug-ins are only recently developed [16].

References

1. Rodden, K., Hutchinson, H., Fu, X.: Measuring the user experience on a large scale: user-centered metrics for web applications. In: Proceedings SIGCHI Conference on Human Factors in Computing Systems, pp. 2395–2398 (2010)
2. Dey, A.K.D., Abowd, G.D.: Towards a better understanding of context and context awareness. In: Proceedings of Workshop: What, Who, Where, When, and How of Context Awareness, ACM Conference Human Factors in Computer Systems (2000)
3. Wang, X., Rosenblum, D., Wang, Y.: Context-aware mobile music recommendation for daily activities. In: Proceedings of ACM International Conference on Multimedia, pp. 99–108 (2012)
4. Achilleos, A., Kapitsaki, G.M.: Enabling cross-platform mobile application development: a context-aware middleware. In: Proceedings of the 15th International Conference on Web Information System Engineering (WISE 2014), pp. 304–318 (2014)
5. Floch, J., Fr, C., Fricke, F., Geihs, K., Wagner, M., Lorenzo, J., et al.: Playing USIC - building contextaware and selfadaptive mobile applications. Softw. Pract. Exp. **43**(3), 359–388 (2013)
6. Paspallis, N., Papadopoulos, G.A.: A pluggable middleware architecture for developing context-aware mobile applications. Pers. Ubiquit. Comput. **18**(5), 1099–1116 (2014)
7. Ioannides, F., Kapitsaki, G.M., Paspallis, N.: Professor2Student - connecting supervisors and students. In: 10th International Conference on Mobile Web Information Systems, pp. 288–291 (2013)
8. Carlson, D., Schrader. A.: Dynamix: An open plug-and-play context framework for android. In: 3rd International Conference on the Internet of Things, pp. 151–158 (2012)
9. Gai, D.: Hybrid VS Native Mobile Apps. http://www.gajotres.net/hybrid-vs-native-apps/. Accessed 26 Sept. 2014
10. Mikkonen, T., Taivalsaari, A.: Reports of the web's death are greatly exaggerated. IEEE Comput. **44**(5), 30–36 (2011a)
11. Mikkonen, T., Taivalsaari, A.: Apps vs. open web: the battle of the decade. In: Proceedings of 2nd Annual Workshop Software Engineering for Mobile Application Development, pp. 22–26 (2011b)
12. Abdullah, M.F.A., Negara, A.F.P., Sayeed, M.S., Choi, D.J., Muthu, K.S.: Classification algorithms in human activity recognition using smartphones. World Acad. Sci. Eng. Technol. **68**, 422–430 (2012)
13. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. SIGKDD Explor. Newsl. **12**(2), 74–82 (2011)
14. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
15. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, J.H.: The WEKA data mining software: an update. SIGKDD Exp. **11**(1), 10–18 (2009)
16. Paphitou, A.C., Constantinou, S., Kapitsaki, G.M.: SensoMan: remote management of context sensors. In: 5th International Conference on Web Intelligence, Mining and Semantics (WIMS 2015) (2015)