UMD Event Finder, A Context-Aware Event Finder Application on Mobile Platform*

Samet Ayhan
University of Maryland, Dept. of Computer Science
College Park, Maryland 20740
sayhan@cs.umd.edu

Snigdha Chaturvedi
University of Maryland, Dept. of Computer Science
College Park, Maryland 20740
snigdhac@cs.umd.edu

Adil Yalcin
University of Maryland, Dept. of Computer Science
College Park, Maryland 20740
adil.yalcin@gmail.com

ABSTRACT

Context-aware event finding system is expected to make use of not only user preferences but also information surrounding the specific contextual situation that is relevant. For instance, a user in search of an event within a certain distance should be presented with a set by taking her current location into account. In fact, not only her current location, but also the distance specified, her preferences, and any other factors that help define the relevant context. This contextual information gathering ought to occur pervasively in an ever changing environment where the process is encapsulated from the user.

In this paper, we describe a new mobile application that presents events on University of Maryland (UMD) College Park Campus to the user based on contextual information in addition to choices the user made in the past. The obvious challenge with this approach is determining and making use of the relevant contextual information per use-case.

Categories and Subject Descriptors

H.m [**Information Systems**]: [Miscellaneous], H.3.4Information Storage and Retrieval[Systems and SoftwareâĂŤDistributed systems], H.3.5Information Storage and Retrieval[Online Information ServicesâĂŤWeb-based services]

General Terms

Design, Human Factors

1. INTRODUCTION

According to the International Telecommunication Union (ITU), the number of cell phone subscribers has reached six billion in 2011, and mobile broadband subscriptions have exceeded one billion globally [3]. Recent mobile phones provide users with a number of features such as Wi-Fi connectivity, bluetooth and GPS localization, camera and video capture devices and, additionally, the capacity for users to program the mobile devices with more applications. Among the most popular applications there are location-based services (LBSs), in which knowledge of the end userâĂŹs location is used to deliver relevant, timely, and engaging content and information [12]. As it is clear that location is not the only information defining the context. Time and any other information pertaining to the situation is considered part of contextual information.

Most web applications today expose themselves with static interfaces in a segregated manner, where search options are presented to the user requiring her to make selections over and over again per session, without taking advantage of surrounding contextual information. Whereas ubiquitous and pervasive computing teaches better integration, systems to make use of contextual information. This way, users are presented with more accurate and timely information without requiring them to provide information that sensors already provide.

With these facts in mind, we have built a system that utilizes contextual information to serve for the purposes of event finding application on agnostic mobile platforms. Some of the key features are:

- Provided with internet connectivity (WiFi, 3G, 4G, cable, etc), any device (mobile or not) with a browser to invoke our service.
- No matter what type of mobile device is used, all capabilities offered by the service.
- In addition to regular event search options, distance proximity and temporal queries are supported.

Furthermore, our application implements Model View Controller (MVC) architecture pattern, where the representation of information is segregated from the user's interaction with it. With this architecture approach, the model that consists of event data and the controlling business logic and the graphical user interface are all separated from each other so that loose coupling, better reusability and more scalable solution is attained.

Among other tools and technologies, our implementation made use of HTML5, JQuery, Python, MySQL, and Apache HTTP Server.

*This effort is part of the CMSC818G, Information-Centric Design of Context-Aware Systems, class project.

The rest of this paper is organized as follows: In Section 2, we present related work, in Section 3, we explain the architecture in addition to design and implementation of three main components, front-end, back-end, and the controlling business logic. In Section 4, we discuss some major use-cases, limitations and optimizations. The final section contains concluding remarks and future work.

2. RELATED WORK

Plenty of research has been done in the area of context-aware computing in the past. Among these efforts, context-awareness has been mainly studied in the fields of ubiquitous, pervasive or mobile computing. Furthermore, a significant number of applications have been successfully developed and deployed.

Peddemors et al. introduce the PLIM framework which provides an infrastructure for the distribution and retrieval of location information of mobile devices using a publish/subscribe mechanism [11].

Harter et al. describe a system that builds a dynamic model of the environment where location information for the objects are updated based on data received from sensors. Also, an event-based monitoring system is provided allowing applications to detect location changes [7].

Indulska et al. present a context model for pervasive systems based on the CC/PP standard and points out some limitations of this standard [8].

In the Web service area, a number of work dealing with context has been done. The CB-SeC framework is an agent-based architecture that provides service selection based on a rating for Web services. The ratings are calculated using context of interest functions [10].

Aura is an architectural framework that models user tasks as coalitions of abstract services. Aura migrates such tasks from one environment to another one when the user changes location. Also, tasks can be adjusted when the environment changes [15]

In the information systems area, preferences are attracting noticeable attention due to fact that they are a means to support personalization within Web services, particularly within information services that often use DBMS as back-end. Preferences, also called soft constraints, are considered as wishes: The result of a query should be a perfect match, but a best possible match is also acceptable [9, 4, **?**].

Among the very first ones, with "The Insider" web application, students were able to keep up with happenings at the Arizona State University campus. Similar to our system, yet missing very key features, this system brought Mesa's Williams Campus together on PC as well as mobile devices [16, 6].

In the area of mobile computing, a number of papers discuss the effect of context on mobile usability, propose an expanded model of mobile application context, and conduct an empirical study to test a number of hypotheses concerning the use of software implementation technology and location context in mobile applications [14].

In their work, Alberto et al. analyse the prerequisites and enablers for context-aware mobile services. According to them, the user perspective, the business perspective, as well as the required technological enablers need to be understood in order to make context-aware services a reality [5].

With their research, Raptis et al. review mobile applications used in museum environments, focusing on the notion of context and its constituent dimensions [13].

3. ARCHITECTURE

In this section, we introduce the system architecture in detail. Although subsequent sections provide more detail regarding each component, here we will provide 10000 feet view.

Illustrated in Figure 1, UMD Event Finder Service utilizes a persistent storage, where events and other relevant information are stored. The data is periodically pulled from disparate data sources in various formats by invoking live feed consumption modules. The service is composed of a number of query modules such as geo-location, temporal, etc. The service makes use of these additional information by contacting corresponding sensors pervasively and incorporating the returned results in its query process. This process is completely encapsulated from the user. Upon search invocation from the end user, the service pulls all necessary pieces together and presents the results back to the user in an efficient and effective manner. The returned results can be viewed on any mobile platform or PC environment with the same look and feel.

3.1 Front-end Module

For our system, we decided to build a web based frontend (using HTML5, CSS3 and Javascript). The web technology has come to a point where rich interactive applications can be developed cross-platform easily. We chose JQuery and JQuery Mobile as the cross-platform interface and support libraries. The other option that was considered was building platform specific apps (android/ iphone/ windows phone). However, their development environments and languages are harder to use and so is deploying such apps to mobile devices, because of (developer and user) permissions required for installation. Our current prototype shows that this has been a reasonable approach given the easy access to our service and interactive interface. We should note that the only disadvantage (from user interaction perspective) is the lack of platform-integrated push notification. Such notifications could be used to alert users of events happening in real time near them. Such an app would ruin in the background, either to notify of the events that the user liked before, or suggest new events happening soon near the user's current location. However, there are also additional issues with such notifications, including the strict guidelines on how often the notifications can be sent, and that inappropriate use would be more annoying than helpful.

Now, let's focus on our user interface approach. First, we identified three main components, as shown in Figure 2: search, results and details. Search and results are presented in the same page (HTML), while details have their own page.

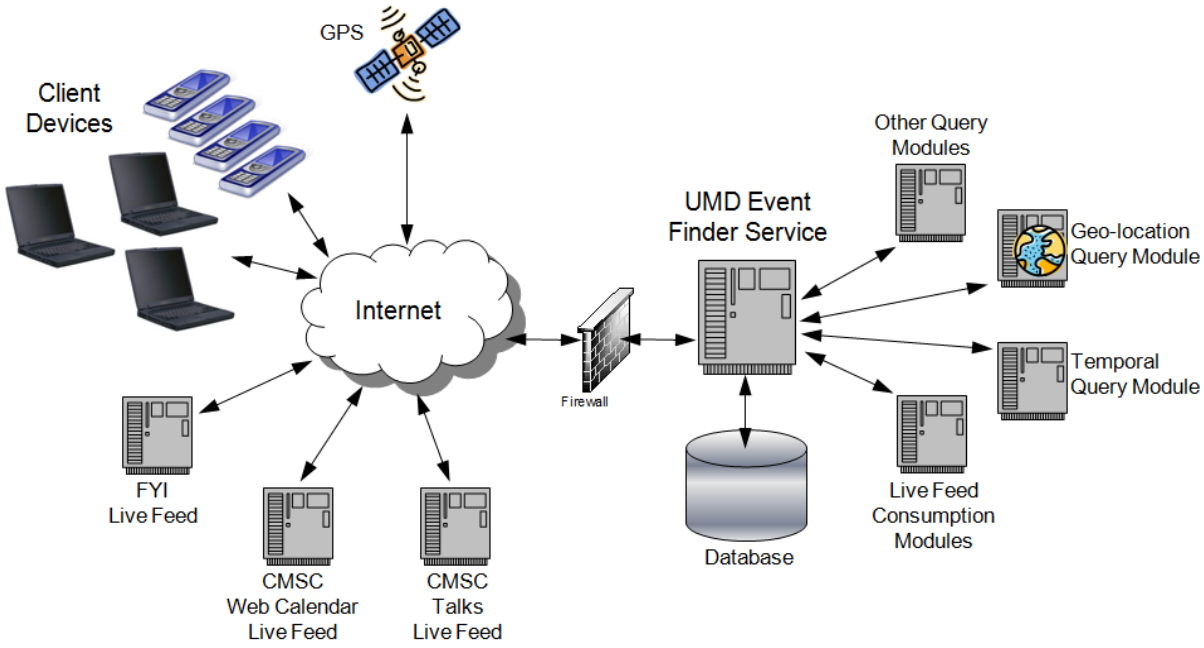


Figure 1: UMD Event Finder - Physical Architecture.

The specific events can be hot-linked using the specific event page and the unique event id.

Basic search options, where, when and what(categories) have multiple pre-defined options. These options may require more details to be presented, such as "'how many hours from now"' and "'which building and how far"'. This presents a 2-level search option, and is shown to be practical and easy to use in our prototype. Yet, this approach forces the options for search settings to a two-level simple hierarchy, and any new option added needs to follows this basic guideline. We use internal pages in JQuery mobile to show these options, which are communicated back to the main search page easily. Some of the search options are presented in Figure 3. We should note that these options are set according to our predicted use cases of an event browsing system for a campus community. Further surveys and user studies need to performed to understand how people would generally use an event query system.

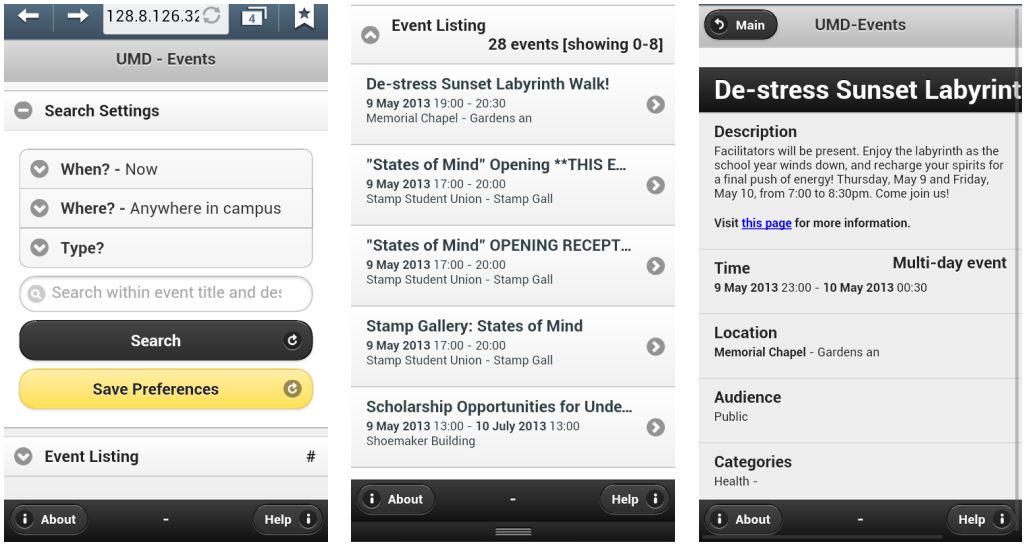
Another feature, which we assume is important for users, is the ability to store personal preferences. Our current solution is local, we store search options in cookie when the user wants to use the settings for later. This is the initial take on implementing profile storage. All the search options are saved, and re-loaded when the page is loaded again. Thus, preferences act as personalized "'default"' settings. The primary use would be filtering based on categories, time as now or future, and location as a specific building or anywhere in campus. The limitation of our current approach is that it only works on a single device. Another limitation is that only a single profile is supported. We predict that an extensive use will result in multiple user profiles.

We propose that a final system should include a user authentication and more detailed user profile management, while still maintaining the simplicity of our current prototype. Regarding including user profiles, we acknowledge that log-in screens and text entry on mobile devices can be deterring users from using the system if the system is only available when logged in. The data is already retrieved from public resources, so should be available to public. The user-account protected data should be restricted to only user-specific features, such as multiple preferences, and access to user liked events. Number of "'likes"' of an event may still be available to the public, as it exposes no personal information. The question to be answered is if it provides additional value to public.

Another minor extension would be configurable sorting of the events. Currently, sorting is based on the event start time, and latest event is shown first. This works well if "'now"' is selected as the time option. However, if "'future"' option is selected, the search is likely to present events happening months later. And, some events may have started a few weeks ago (generally promotions, but some might be valid events that the user may keep up with). Therefore, time based sorting can be extended to list events that is "'closer"' to current time. One other event list sorting option we considered was distance, but we should point that this option should only be available when a location-specific search was made (near building/near me).

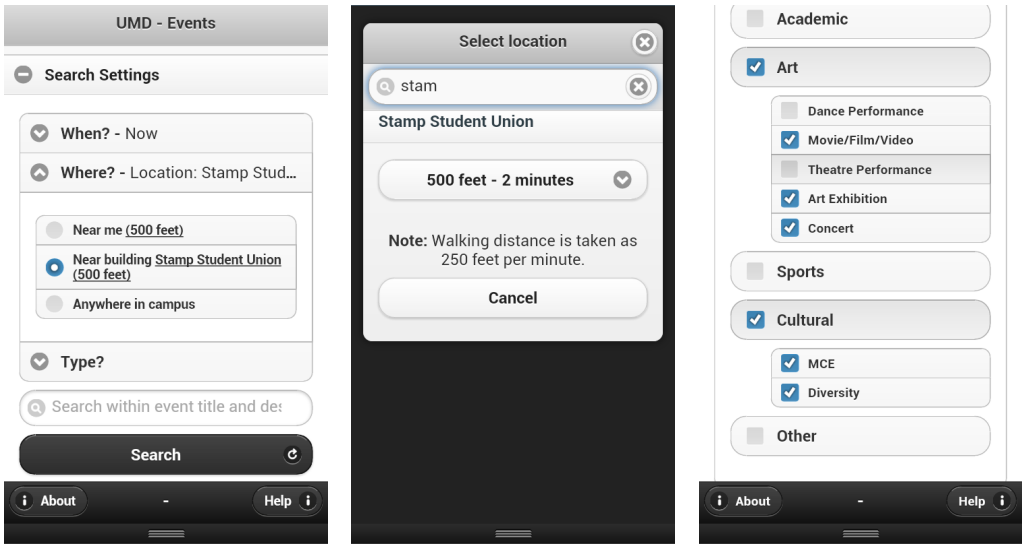
3.2 Back-end Module

In this sub section, we elaborate the design and implementation of back-end module for the UMD Event Finder application.



(a) Main search screen (b) Search results listing (c) Details about a single event

Figure 2: User interface is divided into 3 main sections shown above: Search, Results and Details



(a) Location related options (b) Search & select building (c) Hierarchical categories

Figure 3: Some of the additional search options

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
title	varchar(255)	YES		NULL	
description	text	YES		NULL	
startDateTime	datetime	YES		NULL	
endDateTime	datetime	YES		NULL	
audience	varchar(50)	YES		NULL	
locationName	varchar(100)	YES		NULL	
locationRoomNumber	varchar(10)	YES		NULL	
categories	varchar(100)	YES		NULL	
eventWebsite	varchar(100)	YES		NULL	
announcementDate	date	YES		NULL	
facilId	int(11)	YES	MUL	NULL	
buildingAbbreviation	varchar(5)	YES		NULL	
buildingUrl	varchar(50)	YES		NULL	
bio	text	YES		NULL	
abstract	text	YES		NULL	
speaker	varchar(50)	YES		NULL	
speakerAffiliation	varchar(50)	YES		NULL	
speakerUrl	varchar(100)	YES		NULL	
summary	varchar(255)	YES		NULL	
liked	int(11)	YES		NULL	
sequence	int(11)	YES		NULL	
modDt	datetime	YES		NULL	
allDay	varchar(5)	YES		NULL	
timeStampNow	datetime	YES		NULL	

25 rows in set (0.03 sec)

Figure 4: UMD Event Finder - Schema for the EVENTS table.

Field	Type	Null	Key	Default	Extra
bldgId	varchar(3)	YES		NULL	
bldgName	varchar(100)	YES		NULL	
bldgNn	varchar(100)	YES		NULL	
facilityId	int(11)	NO	PRI	NULL	
bldg3Code	varchar(3)	YES		NULL	
xCentCoord	float(10,4)	YES		NULL	
yCentCoord	float(10,4)	YES		NULL	

7 rows in set (0.00 sec)

Figure 5: UMD Event Finder - Schema for the BUILDINGS table.

Defined as Model in the MVC architecture pattern, back-end module is responsible for building and managing the data storage. Using stored data, the application is able to respond to users when the request is made.

3.2.1 *Database Design*

The back-end module contains a relational database, UMDEVENTS with two tables:

- EVENTS and
- BUILDINGS

Events are stored in EVENTS table and facilities on campus are stored in BUILDINGS table. The tables are connected using `facilityId`. Figure 2 illustrates the schema for the EVENTS table.

`id` is the primary key and `facilId` is the foreign key in the EVENTS table.

Figure 3 illustrates the schema for the BUILDINGS table.

`facilityId` is the primary key for the BUILDINGS table.

UMD Event Finder application uses MySQL for relational database.

BUILDINGS table has been created and populated by transforming an Excel spreadsheet containing facilities data for the entire campus, provided by the Facilities Management. Spatial and non-spatial attributes for the facilities such as facility name, id, latitude, and longitude values are included in the original data.

EVENTS table has been designed and implemented by taking available live data sources into account. The following on campus live data feed were acquired:

- For Your Information (FYI) live data feed available at [**?**].
- Computer Science Talks live data feed available at [1].
- Computer Science Web Calendar live data feed available at [2].

The back-end module is basically composed of a number of a number CRON jobs that perform the following scheduled tasks:

- Provided the URLs, subscribe to the live events feeds
- Parse the incoming data based on schema and format (XML, JSON)
- Check if the unique id per event exists in the database
- Push the incoming data into database, if not
- Repeat the above steps periodically

The largest events data source on campus, FYI is exposed through a live RSS feed in addition to a static web page. Per our request, the IT Department made the live data feed available in XML. Upon a new event entry by an authorized person, the FYI service accumulates all the events and publishes them periodically in XML form. The UMD Event Finder application back-end module subscribes to the feed on a daily basis, parses the incoming data using Document Object Model (DOM) and pushes it to the internal database.

In addition to the live FYI feed, the back-end module subscribes to Computer Science departmental feeds. Unlike the FYI, these feeds are provided in JSON. Similar to the FYI subscription, the back-end module subscribes to these departmental feeds, parses the incoming data and pushes it to the internal database.

Events with duplicate identifiers are not allowed in the database. The current subscription frequency is 86400 seconds for all three live data feeds and this frequency is configurable. The current implementation language for the back-end module is Python. The module uses PyMySQL interface to make connection from subscriber to the MySQL database.

3.2.2 *Querying the Database*

This section describes the module for handling the queries from the database.

This component is also implemented in Python and uses CGI to interact with the web server and PyMySQL to connect to the MySQL database. As described in Section 3.1, the user selects time, location and category choices using pre-specified options. These options are passed to this module in form of CGI parameters to a python script which parses them and constructs appropriate MYSQL queries.

One of the front end features allows searching for events happening within a specific radius of a building or user's current position. While building's or user's locations are specified in terms of latitudes and longitudes, a natural way to compute radius would be using Euclidean distances between geographical locations. For this purpose, the latitude-longitude information is converted to Cartesian coordinates before computing distances. Given a points latitude, θ and longitude, ϕ, the Cartesian coordinates (x, y) are computed as:

$$\vec{r} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} R\cos\theta\cos\phi \\ R\cos\theta\sin\phi \end{pmatrix} \quad (1)$$

where $R \approx 6371$km is the radius of the Earth.

For simplicity, it was assumed that all points lie on the XY plane, ie, $z = 0$ for all points. Thereafter, the distance between two points, $\vec{r_1}, \vec{r_2}$ was calculated as:

$$d(\vec{r_1}, \vec{r_2}) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (2)$$

The rest of the queries are in form of SELECT commands returning rows (event details) from the EVENTS table. The retrieved rows are then returned to the front-end in form of json objects.

This concludes the back-end module.

4. USE-CASES

The primary use case of this service is to provide a context aware search of events happening on or around the UMD campus. The contextual information exploited in this system include time and user location. Specifically, the system automatically detects the current geographical location of the user and time and allows searching based on this information. Typical use cases of this system are:

1. Consider a user who is present on campus, is free at the moment and is wondering if there is an event happening nearby which he/she can attend. Such a user can select the 'now' and 'near me' option and the system will display the appropriate events.

2. Another use case is of a user who is busy presently but is interested in attending an event in next couple of hours. Such a user can select the 'in _ hours' option and look for events of interests.

3. Suppose the above user knows that he/she would be constrained to a tentative location on campus after a few hours and wants to look for events that he/she can attend at that time and location. Such a search can be executed using the 'in _ hours' option and also selecting the tentative location.

4. The system can also be useful for users who are planning their week schedule and are interested in events happening between specific dates.

5. The system is also caters to users with more specific needs. For example, some more 'sedentary' users might be interested in events happening in or around a specific building only. Such a search can be executed by specifying the name of the building of interest and the distance and selecting an appropriate time option. The system, in such a case displays only the events located near the specified building.

6. Another set of users might only be interested in certain categories of events such as academic and/or sports only. Such users can specify their interests using the event 'type' option.

7. For users whose interests are not limited to specific events types but to certain other constraints that can be determined by the event description or titles, the system allows searching the title or description for user-specified word(s).

The above use cases are examples of situations when a user wants to look for events under certain constraints. However, regular visitors to the service would have nearly constant constraints. The system saves such users from specifying the same options multiple times by saving the user preferences. The next time a user visits the service website, the search criteria will be predetermined based on his/her saved preferences. The user will be free to modify and overwrite these preferences at his discretion.

5. CONCLUSION AND FUTURE WORK

This paper introduced a novel system that presents UMD events on agnostic mobile platforms based on contextual information in addition to user preferences. Taking major contextual information into account such as location and time in an ever changing environments, our system retrieves relevant events and presents to the user in an efficient and effective way.

This very first version of our system delivers context-based services by contacting sensors pervasively and making use of every bit of relevant information for the events being sought. However, there is still room for improvement. Among others, authentication, persistent user profiles, integration with social media, and automated notifications using pub-sub pattern are all critical capabilities that may be developed.

6. ACKNOWLEDGMENTS

We would like to thank Brooke Supple for allowing us to connect with IT resources. We would also like to thank Skip Warnick and Brian SwartzfageâĂŃr for their help in providing us with access to the UMD FYI Events live feed. We thank all of the anonymous contributors. We would especially like to thank Dr. Agrawala for his advice and directions.

7. REFERENCES

[1] Cstalks. `https://talks.cs.umd.edu/talks.json`, 2013.

[2] Cswebcal. `http://www.cs.umd.edu/webcal/webcal_dept.json`, 2013.

[3] Itu. `http://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx`, 2013.

[4] R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. *SIGMOD Rec.*, 29(2):297–306, May 2000.

[5] P. Alberto. User-centric mobile services: context provisioning and user profiling. In *Proceedings of the 11th Annual International Digital Government Research Conference on Public Administration Online: Challenges and Opportunities*, dg.o '10, pages 122–130. Digital Government Society of North America, 2010.

[6] H. W. Gellersen, A. Schmidt, and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mob. Netw. Appl.*, 7(5):341–351, Oct. 2002.

[7] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. *WIRELESS NETWORKS, VOL*, 8:187–197, 1999.

[8] J. Indulska, R. Robinson, A. Rakotonirainy, and K. Henricksen. Experiences in using cc/pp in context-aware systems. In *Proceedings of the 4th International Conference on Mobile Data Management*, MDM '03, pages 247–261, London, UK, UK, 2003. Springer-Verlag.

[9] W. KieÃ§ling and B. Hafenrichter. Optimizing preference queries for personalized web services. In *INSTITUTE OF COMPUTER SCIENCE, UNIVERSITY OF AUGSBURG*, pages 461–466, 2002.

[10] Z. Maamar, S. Kouadri Mostefaoui, and H. Yahyaoui. Toward an agent-based and context-oriented approach for web services composition. *IEEE Trans. on Knowl. and Data Eng.*, 17(5):686–697, May 2005.

[11] A. J. H. Peddemors, M. M. Lankhorst, and J. d. Heer. Presence, location, and instant messaging in a context-aware application framework. In *Proceedings of the 4th International Conference on Mobile Data Management*, MDM '03, pages 325–330, London, UK, UK, 2003. Springer-Verlag.

[12] B. Rao and L. Minakakis. Evolution of mobile location-based services. *Commun. ACM*, 46(12):61–65, Dec. 2003.

[13] D. Raptis, N. Tselios, and N. Avouris. Context-based design of mobile applications for museums: a survey of existing practices. In *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, MobileHCI '05, pages 153–160, New York, NY, USA, 2005. ACM.

[14] C. Ryan and A. Gonsalves. The effect of context and application type on mobile usability: an empirical study. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38*, ACSC '05, pages 115–124, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[15] J. a. P. Sousa and D. Garlan. Aura: an architectural framework for user mobility in ubiquitous computing environments. In *Proceedings of the IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture: System Design, Development and Maintenance*, WICSA 3, pages 29–43, Deventer, The Netherlands, The Netherlands, 2002. Kluwer, B.V.

[16] B. Valenzuela. The insider: the web site that brought mesa's williams campus together. In *Proceedings of the 31st annual ACM SIGUCCS fall conference*, SIGUCCS '03, pages 103–107, New York, NY, USA, 2003. ACM.