Context Inference for Mobile Applications in the UPCASE Project*

André C. Santos[1], Luís Tarrataca[1], João M. P. Cardoso[2], Diogo R. Ferreira[1], Pedro C. Diniz[1], and Paulo Chainho[3]

[1] IST – Technical University of Lisbon, Taguspark, Oeiras, Portugal
[2] FEUP – Faculty of Engineering, University of Porto, Portugal
[3] PT Inovação S.A., Portugal
Corresponding author: jmpc@acm.org

Abstract. The growing processing capabilities of mobile devices coupled with portable and wearable sensors have enabled the development of context-aware services tailored to the user environment and its daily activities. The problem of determining the user context at each particular point in time is one of the main challenges in this area. In this paper, we describe the approach pursued in the UPCASE project, which makes use of sensors available in the mobile device as well as sensors externally connected via *Bluetooth*. We describe the system architecture from raw data acquisition to feature extraction and context inference. As a proof of concept, the inference of contexts is based on a decision tree to learn and identify contexts automatically and dynamically at runtime. Preliminary results suggest that this is a promising approach for context inference in several application scenarios.

Key words: Context-aware services, context inference, smartphones, wearable sensors, decision trees.

1 Introduction

There is a growing desire of telecommunication operators to increase traffic volume even further by offering value-added services to customers in addition to traditional voice and data communication. These services can be enabled or disabled depending on the specific user context. For example, when caught in rush-hour traffic, a service could automatically estimate the delay for the user to reach a child's school. In case of excessive delay, it would notify an alternate adult for pick-up. Other examples include anti-theft or near-emergency services. Using sensors it might be possible to determine whether an elderly has fallen at home and has been immobile for some time thus triggering an emergency call.

To enable such kind of services, mobile devices must be able to clearly identify specific contexts the user goes through [12, 27]. For this purpose, mobile devices must include sensors that yield data such as position, lighting or sound

* This work was partially funded by PT Inovação S.A.

conditions from which user contexts can be determined. Accurate context inference, however, is notoriously difficult as there exist various sources of data signals with possibly very distinct patterns which need to be captured and processed in a timely fashion. Furthermore, the amount of raw sensor data can overwhelm the resources of even the most sophisticated mobile devices. A possible solution would require each mobile device to acquire and transmit sensor data to a centralized server for processing. Although conceptually simple, this centralized solution is infeasible. It would require constant communication with a centralized server as most sensors need to operate in real-time. This in turn would require excessive computing power for each device to constantly transmit a possible high volume of sensor data. On the server side fusing sensor data from millions of devices would require a tremendous computing power. Instead, each mobile unit should be able to infer user context by processing data originating from its sensors and possibly from communicating with network services to obtain additional information such as traffic or weather conditions.

In this paper, we describe the architecture and operation of a proof-of-concept system for context inference based on a smartphone augmented with an array of sensors connected via *Bluetooth*[2]. This system is part of the UPCASE project (User-Programmable Context-Aware Services), an industry-funded R&D project. The architecture of the system consists of three main layers: (1) the acquisition layer which is concerned with sensor data acquisition and preprocessing, (2) the feature extraction layer which assigns specific categories to the preprocessed sensor data, and (3) the context inference layer which uses decision-tree induction techniques to uncover the user context.

This paper is organized as follows. We begin with an overview of related work and describe the developed system. Next, we present the various sensors used in connection with the smartphone and the experimental results of context inference in a simple scenario of daily activities. Lastly, we describe two potential scenarios of application of the system developed in identifying meaningful contexts, namely in elderly care and emergency management scenarios.

2 Background and Related Work

Context identification as been recognized as an enabling technology for proactive applications and context-aware computing [4, 11]. Sensor networks can be used to capture intelligence (see, e.g., the e-SENSE[3] project [17]), providing sensing capabilities from the environment and opening opportunities for context-aware computing.

Early context-aware applications were predominantly based on user location defined as typical user places (e.g., "at home", "in a museum", "in a shopping center"). Projects such as GUIDE [3] and Cyberguide [1] addressed the use of information about location and situation to guide the user when visiting touristic

[2] The Official Bluetooth Technology Info Site (www.bluetooth.com/bluetooth).
[3] http://www.ist-esense.org/

city spots. Recently, researchers have studied techniques to identify a richer set of contexts or activities. These include simple user activities (e.g., "walking", "running", "standing"), environment characteristics (e.g., "cold", "warm"), or even emotional condition of the user (e.g., "happy", "sad", "nervous").

In the SenSay [23] project, researchers developed a smartphone prototype able to exploit the user context to improve its usability. For example, if the user is occupied and wishes not to be interrupted, the smartphone can answer/reply automatically using an SMS. The SenSay prototype uses a smartphone and a sensor unit consisting of a 3-axis accelerometer, two microphones (one to capture sound from the environment and the other to capture voice from the user), and a light sensor. The prototype makes use of simple techniques such as performing an average of sensor readings over a given window and applying a numeric threshold to identify each activity.

Generally, the identification of contexts is done in stages. Processing raw data from sensors may require a wide variety of techniques such as noise reduction, mean and variance calculation, time- and frequency-domain transformations, estimation of time series or sensor fusion. Data collected from sensors is catalogued (a process known as feature extraction) and the context-inference stage makes use of features rather than raw data. Context inference has been addressed using different techniques such as Kohonen Self-Organizing Maps (KSOMs) [14], k-Nearest Neighbor [15], Neural Networks [20], and Hidden Markov Models (HMMs) [24]. Some approaches even combine several of these techniques, as in [13].

Regarding the inference of user activities such as "walking" or "running", they use a plethora of approaches, ranging from simple processing steps and threshold operations [8, 22, 27] to the use of neural networks as the clustering algorithm [20]; or even using non-supervised time-series segmentation [9]. As an example, the work presented in [12] infers activities such as "walking", "running", "standing", and "sitting" with a single 3-axis accelerometer claiming an accuracy of 96%.

In our approach, we extract signal features using techniques similar to those described in [22, 27]. For context inference we combine signal-processing and machine-learning techniques, using decision trees [18] to fuse features and determine user activities. All data preprocessing and context inference is performed on the mobile device. The results are sent to a server in order to monitor activities, thus allowing for more advanced and possibly non-local context inferences.

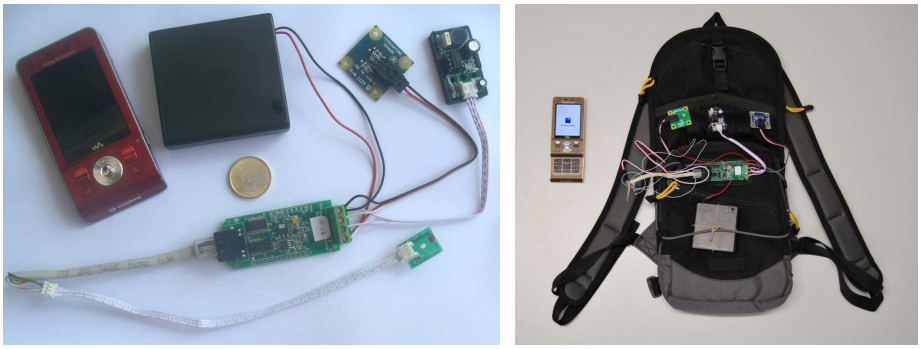
3 The UPCASE Project

The UPCASE project aims at uncovering user contexts using a set of sensors connected to the user's mobile phone. These sensors can be embedded into personal clothes or items such as backpacks or purses. Sensors include accelerometers, light, sound, and temperature sensors, and also *virtual* sensors to acquire information such as time of day or approximate location via external services.

A goal of the project includes the development of robust algorithmic approaches to accurately determine user context. Specifically, we include supervised-based learning techniques. During a training phase the system collects a sufficient number of data samples for context derivation using decision-tree based techniques. After this training phase the system operates autonomously and unobtrusively automatically deriving contexts.

3.1 System sensors and prototype

Figure 1(a) depicts the main system components used in the prototype we have developed: the mobile device, a sensor node, and a set of sensors. The black box contains the batteries (the 1-Euro coin is shown to provide an idea of scale). Figure 1(b) depicts an experimental setup where the components are embedded in a backpack used for testing purposes. In this early prototype, we have deliberately not concealed the sensors to experiment with sensor sensitivity to environment conditions. The prototype has been tested on a backpack and also on a vest, ensuring that the sensors experience the same conditions as the user. The only requirement is that some sensors must be exposed to allow for more correct sound, temperature and light measurements.



(a) (b)

Fig. 1. The system components (a) and an experimental setup in a backpack (b).

The system prototype comprises a Sony Ericsson W910i smartphone[4] and a BlueSentry external sensor node[5]. A sound sensor[6], a temperature sensor[7], and a light sensor[8] are wired to the sensor node. The BlueSentry sensor node communicates with the smartphone via Bluetooth to provide sensor readings, thus avoiding the need for physical connection between the two. In addition

[4] http://www.sonyericsson.com/cws/products/mobilephones/overview/w910i
[5] http://www.rovingnetworks.com/bluesentry.htm
[6] http://www.inexglobal.com/products.php?model=zxsound
[7] http://www.phidgets.com/products.php?product_id=1124
[8] http://www.phidgets.com/products.php?product_id=1127

to these, there are two other sensors being used: the internal accelerometer of the smartphone and a virtual time sensor to provide the time of day. It is also possible to connect a second accelerometer to the BlueSentry node.

3.2 System sensors and prototype

The overall system architecture is presented in figure 2. The application layer has been developed using Java ME platform[9], a technology that is widely used due to its recognized portability across many mobile phone devices. At the lowest level, the sensors gather data from the environment and provide it as raw analog signals to the sensor node, which in turn converts them to digital and transmits the digital representations to the mobile phone via Bluetooth. The mobile phone runs a proprietary operating system (OS) which supports the execution of Java code. We have developed a MIDP (Mobile Information Device Profile) application that acquires raw sensor data from the BlueSentry node and supports the extraction of features to be used in the upper layers of the system architecture.

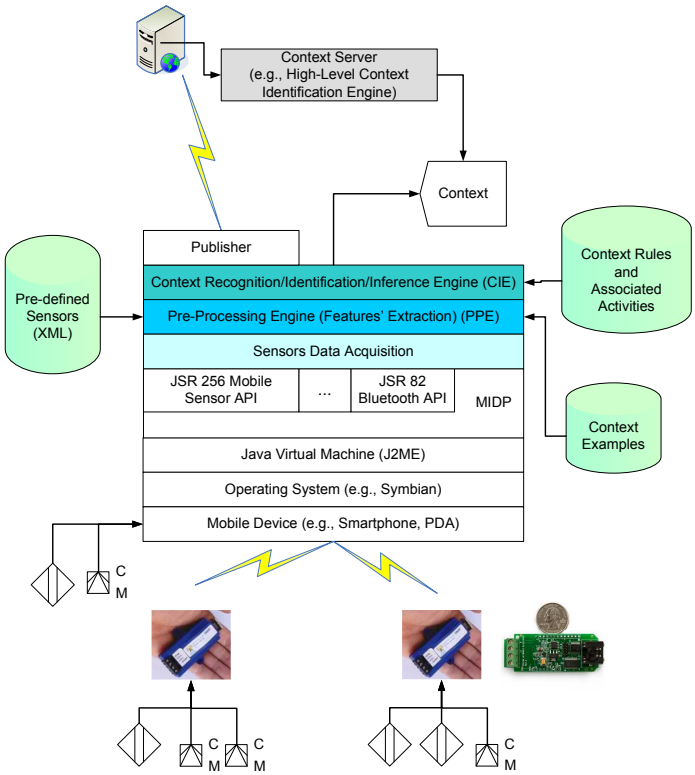


Fig. 2. The UPCASE system architecture.

[9] http://java.sun.com/javame/

At runtime, there are three main stages: a sensor acquisition stage, a preprocessing stage, and a context inference stage. Sensor data acquired from the available sensors are fed to the preprocessing stage, which is responsible for extracting signal features. The inference stage gathers these features and identifies the context that is associated with them according to a set of rules. These rules are either pre-defined or derived by a learning phase. This synergy between stages can be seen as a process that maps low-level sensor data into higher-level context information. Both the preprocessing and the context inference stage can be configured by specifying the available sensors and the default context rules. In addition, it is possible to add new sensors or replace existing ones by means of an XML configuration file, thus providing an easy and simple way of managing the set of system sensors.

The upper section of the figure depicts the context-inference stage and the system interaction with external resources, such as a centralized server. Each mobile device, upon having recognized a specific context, can (contingent on user permission) publish it on a server to trigger other actions or custom services, or to provide statistical data about user behavior that can be used to generate other services or improve the performance of existing ones. A server can take advantage of historical data to improve the accuracy of context information.

3.3 Sensor data acquisition

We make use of JSR-256 Mobile Sensor API[10] whenever it is supported by the mobile device. The JSR-256 API allows developers to retrieve data not only from embedded sensors but also from sensors connected via infrared, bluetooth and GPRS. When the JSR-256 is not supported, we use the JSR-82 Bluetooth API[11]. At the moment we are using JSR-256 for the internal accelerometer in the Sony Ericsson W910i device, and JSR-82 for all other data acquisition.

Sensor data are acquired at a fixed rate. At regular intervals, the device sends a request to the sensor node in order to retrieve data from the sensors connected to that node. The sensor readings are buffered in the smartphone to be fed later to the preprocessing stage. It is worth noting that sensors may have different acquisition rates. The difference in sampling frequency may force the acquisition to run at the slowest rate, or at individual rates for each sensor. Currently we are using the same acquisition rate for all sensors.

3.4 Preprocessing

The preprocessing stage converts raw sensor data into a set of features. This conversion is performed by means of threshold operations that map sensor data into a finite set of categories. Rather than using instant values, the system captures a sequence of sensor readings and then preprocesses them in order to minimize jitter and to provide a more accurate categorization. The preprocessing may

[10] http://jcp.org/en/jsr/detail?id=256
[11] http://jcp.org/en/jsr/detail?id=82

involve averaging, filtering or transforming values, depending on the particular sensor the data come from.

Table 1. Categorization of the sensor values acquired.

Sensor	Category	Value range
sound	very silent	0% - 20%
	silent	20% - 40%
	moderate	40% - 60%
	loud	60% - 80%
	very loud	80% - 100%
light	very dark	0 - 200
	dark	200 - 400
	normal	400 - 600
	bright	600 - 800
	very bright	800 - 1000

Sensor	Category	Value range
temperature	very cold	−50°- 0°
	cold	0°- 15°
	mild	15°- 25°
	hot	25°- 30°
	very hot	30°- 150°
time	dawn	0h - 5h
	morning	6h - 11h
	afternoon	12h - 17h
	night	18h - 23h
accelerometer	not moving	variance-based
	moving	variance-based
	moving fast	variance- and FFT-based

Table 1 presents the categories for each sensor, as well as the range of values for each category. Clearly, a limited number of categories places some limits on the total number of contexts that can be identified. However, a too large number of categories can also divide the perception of the environment in too many different states. According to the user contexts to be identified and according to the specific application domain, the categories of sensor values may have to be adjusted. In general, however, most applications will make use of only a limited set of contexts and the above categories, we believe, are enough to handle many practical situations.

For some sensors, the raw sensor value can be mapped directly to a category, while other sensors such as the accelerometer need more elaborated preprocessing. For each sensor there is a fixed buffer window. For the temperature, light and sound sensors, the category is found by applying threshold operations to the mean value inside the buffer, making context inference less prone to sensor noise.

For the 3-axis accelerometer, the system calculates the variance for each axis along a time-framed window (last 16 data samples). Those three variances obtained are compared to a threshold in order to identify "moving" and "not moving" activities. When movement is detected, the system performs an FFT over the last 32 data samples and the amplitudes of the harmonics representing frequencies within the range of "running" activities (currently 0.5 to 2 Hz) are added up. If the resulting value is greater than a specific threshold value a "running" context is signaled. Otherwise, a "walking" context is determined.

4 Context Inference in the UPCASE Project

At any given moment, the set of sensor readings will have some relationship with user activity of the surrounding environment. The purpose of context inference

is to discover this relationship so that when a similar set of readings occur, the device will recognize the same context. While different contexts can lead to the same set of sensor readings the converse is also true: different readings may correspond to the same context, as there will be naturally some variation in sensor values. Context inference must therefore identify the range of measurements that typically corresponds to each context.

4.1 Context inference with decision trees

Decision trees are structures that fit the purpose of induction and are fast to build and process, which makes them attractive for implementation on mobile devices. Provided with a set of training examples, decision tree induction attains a classification of contexts according to a set of sensor readings. Our implementation is based on the ID3 algorithm [19], which uses information entropy to build the smallest tree that can correctly identify all the branches.

The decision tree is first built from a set of default context rules. These initial rules are represented in XML so that they can be provided as a different starting point for different user profiles (e.g., student pack, sports pack, professional pack). The tree is then updated as the user trains the system with new contexts. As the system is being trained, the whole tree is recomputed and updated with the learned contexts.

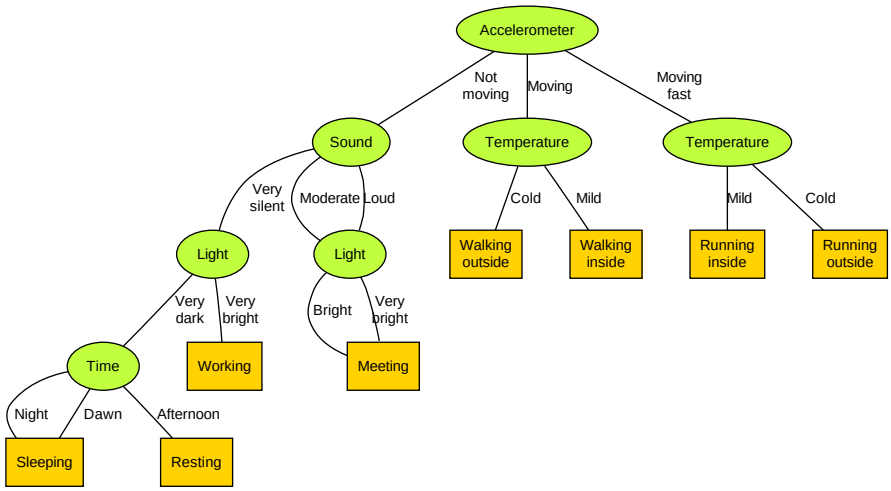


Fig. 3. Example of an induced decision tree.

Figure 3 depicts an example of an induced decision tree using five sensors, namely: time, sound, accelerometer, light and temperature. The tree has paths from root to leaf nodes that provide the rules for context identification based on identified features. In this example the accelerometer provides the most distinc-

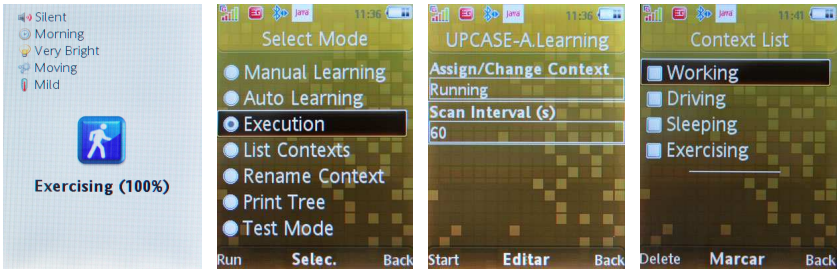
tive feature, followed by temperature and sounds sensors, and only then by light and time sensors.

Learned contexts can be renamed or deleted, as the system provides means for the user to manage existing contexts. This way the user can erase a learned context and train the system again, or simply remove an unused context. We are studying ways to allow the user to correct only some rules without having to completely re-train the system. In any case, recomputing the decision tree is sufficiently fast for not being noticeable by the user.

The context identified via the decision tree is stored in a buffer which gathers a finite number of contexts and returns the context that has been recorded more often within a certain time window. This avoids momentaneous conditions that would make the context change unexpectedly. Together with sensor data buffering, this provides another layer that minimizes errors due to faulty sensor readings or to activities that were detected for only a brief moment within a different context.

4.2 Application layer

Figure 4 provides some screenshots of the UPCASE application running on the smartphone. The application presents a simple yet effective user interface, allowing different modes chosen from a list of available options. It includes the possibility of editing existing contexts and printing the decision tree for debugging purposes. Figure 4(c) presents an example of the initial configuration for the continuous context-learning mode which acquires sensor data during a period of time when a certain context is active. In figure 4(a) we can see the different sensor readings and the identified context, as well as a confidence value calculated as the percentage of total records for the displayed context within the buffer window. A suggestive icon is also presented to the user.



(a) (b) (c) (d)

Fig. 4. The UPCASE application in operation mode (a), in selection mode (b), in learning mode (c), and in context-editing mode (d).

4.3 Learning mode

The decision tree shown in figure 3 can be induced from a set of examples collected during a training period. When the user sets a context and a learning period as in figure 4(c), the system collects sensor readings during that period and classifies the readings as examples for a specified context. It then uses these examples to update the decision tree. Figure 5 presents the sensor readings collected for four different contexts in a simple test scenario. The results are shown for a period of ten minutes, where the system acquired approximately 300 examples for each context.

(a) (b) (c) (d)

Fig. 5. Sensor readings for different contexts: sleeping (a), working (b), exercising (c) and driving (d).

As can be seen in figure 5, the sound sensor exhibits the highest variability which can be explained by the fact that this sensor is particularly sensitive. The light sensor shows no variation as the data for this experiment were captured in constant lighting conditions. As a result, light will not become a discriminator in the induced decision tree. A similar situation occurs with the temperature sensor, where the *Mild* category is present in all contexts. The accelerometer provides useful information, as some contexts can be distinguished in terms of movement. Lastly, the time sensor allows the derivation of the most distinctive

feature, as it provides a constant value within each context, and a different value for three out of four contexts. Figure 6 depicts the induced decision tree for this testing scenario.

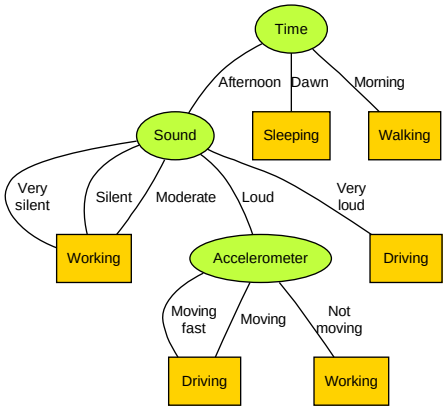


Fig. 6. Induced decision tree from the readings shown in figure 5.

4.4 Context server

The context can be uploaded, upon user permission, to a centralized server at the network operator over `http` and conforming to the REST API[5]. This has several advantages. First, it is possible to enable, disable or change the behavior of value-added services that depend on user context. Second, user context can be augmented with information available at the network level, such as traffic conditions or special services available at the user location. Last, the publication and availability of user contexts open the way for advanced social networking services and other applications.

5 Case-study Scenarios

In this section we describe two potential scenarios of the application of the UPCASE system for context detection and exploitation. Both cases require additional sensors while still relying on the context-detection techniques described in this paper. The applications exploit not only the ability of the system to identify user contexts, but also of making context information available to other users via a context server.

5.1 Elderly care

Devising support technologies for elderly care is an active area of research [16, 25] where applications have traditionally focused on monitoring the location

and risk-prone activities of elderly persons. These applications usually require multiple sensors [10], or the user to carry appropriate sensors, for example for automatic fall detection [7].

For elderly people that have an active life but still inspire some care, a general-purpose, inexpensive system such as the one developed in the UPCASE project can help a family member keep track of their daily activities. Such approach is not intended to limit the activity of an elderly person, but simply to provide an up-to-date account of the elderly person's activities. For example, if a person is resting or performing some activity at home, family members can be reassured of the person's well-being; on the other hand, knowing that the person is driving or inside a car makes them aware of activities that inspire more care. In general, such context-aware systems can enable family members to promptly respond to any situation that requires attention, while allowing elderly people the freedom they desire.

In this scenario, as well as in medical applications, there are additional sensors such as heart-rate monitors or blood-pressure sensors. Contexts can be published in the context server, and family members can retrieve them using any Web-enabled device. We are currently developing this scenario in connection with wearable sensor systems equipped with physiological sensors such as ECG, heart rate, oxygen, posture, and body temperature.

5.2 Emergency management

Emergency management is the discipline that deals with preparing for, preventing, responding to, and recovering from emergency situations [6]. Under extreme conditions, it is essential to be able to: make accurate decisions; coordinate a number of team members and to keep track and dynamically allocate available resources. These and other emergency-related challenges can be addressed by having appropriate information systems in place [21].

Sensor-based and context-aware systems can play a key role in preventing and also in responding to emergency situations. These systems usually make use of front-end and back-end components in order to coordinate team members on the field [2, 26]. A key requirement of any emergency response system is its ability to know the state of readiness of the allocated workforce. For a given situation, team members that are both on duty and off-duty may have to be called in. Knowing the context of each staff member can help determine which members are more or readily available to promptly respond to the situation. For example, it may be the case that only active, or free members, or members currently driving, or in any other specific context, should be called to the scene, thereby substantially reducing the emergency total response time.

The scenario can be supported by the UPCASE system, as sensors can be easily merged into the members outfit. In the background, a server receives context information from the smartphones and stores it in a database where all contexts are kept up-to-date. In an emergency situation, the database can be queried to quickly obtain a list of all team members in a specific context.

6 Conclusion

Being able to gather information about user context is a key enabling factor for a new generation of context-aware services and applications. In this paper we addressed the problem of distinguishing between a number of daily activity contexts by means of a prototype proof-of-concept system developed in the context of the UPCASE project. The system prototype is based on a set of general-purpose, inexpensive sensors connected to a regular smartphone via a Bluetooth-enabled sensor node. It is small enough to be embedded in clothes or other personal objects, and it operates in an unobtrusive manner after an initial training period. Context inference is based on decision tree induction, which provides a simple and lightweight procedure that can be implemented in devices with limited processing capabilities. The approach is effective in a number of applications, and we hope it will spur the interest in similar methods to identify and make use of contexts in mobile applications.

References

1. G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A Mobile Context-Aware Tour Guide. In *Proc. of the Intl. Conf. on Mobile Computing and Networking (MobiCom'96)*, pages 421–433, 1996.
2. T. Catarci, M. de Leoni, A. Marrella, M. Mecella, B. Salvatore, G. Vetere, S. Dustdar, L. Juszczyk, A. Manzoor, and H. Truong. Pervasive Software Environments for Supporting Disaster Responses. *IEEE Internet Computing*, pages 26 – 37, Jan.-Feb. 2008.
3. K. Cheverst, N. Davies, K. Mitchell, and A. Friday. Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project. In *Proc. of the Sixth Annual Intl. Conf. on Mobile Computing and Networking*, pages 20–31. ACM Press, 2000.
4. J. Coutaz, J. L. Crowley, S. Dobson, and D. Garlan. Context is Key. *Communications of the ACM*, 48(3):49–53, March 2005.
5. R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, Univ. California at Irvine (UCI), Irvine, Calif., 2000.
6. G. Haddow, J. Bullock, and D. Coppola. *Introduction to Emergency Management*. Butterworth-Heinemann, 2007.
7. T. Hansen, J. Eklund, J. Sprinkle, R. Bajcsy, and S. Sastry. Using smart sensors and a camera phone to detect and verify the fall of elderly persons. In *Proc. of the European Medicine, Biology and Engineering Conf. (EMBEC 2005)*, Nov. 2005.
8. J. Healey and B. Logan. Wearable wellness monitoring using ecg and accelerometer data. In *Proc. of the Ninth IEEE Intl. Symp. on Wearable Computers (ISWC'05)*, pages 220–221, Washington, DC, October 2005. IEEE Computer Society Press.
9. J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmäki, and H. Toivonen. Time series segmentation for context recognition in mobile devices. In *Proc. of the 2001 IEEE Intl. Conf. on Data Mining (CDM'01)*, pages 203–210, Washington, DC, Nov. 2001. IEEE Computer Society Press.
10. T. Hori, Y. Nishida, H. Aizawa, S. Murakami, and H. Mizoguchi. Sensor network for supporting elderly care home. In *Proc. of IEEE*, pages 575 – 578, Oct. 2004.

11. R. Hull, P. Neaves, and J. Bedford-Roberts. Towards situated computing. In *Proc. of the Intl. Conf. on Wearable Computers (ISWC'97)*, pages 146–153, 1997.
12. Y. Kawahara, H. Kurasawa, and H. Morikawa. Recognizing user context using mobile handsets with acceleration sensors. In *(IEEE) Intl. Conf. on Portable Information Devices (PORTABLE'07)*, pages 1–5, 2007.
13. A. Krause, A. Smailagic, and D. Siewiorek. Context-aware mobile computing: Learning context-dependent personal preferences from a wearable sensor array. *IEEE Trans. on Mobile Computing*, 5(2), Feb. 2006.
14. K. V. Laerhoven. Combining the kohonen self-organizing map and k-means for on-line classification of sensor data. In *Artificial Neural Networks (ICANN 2001)*, pages 464–470, 2001.
15. K. V. Laerhoven and O. Cakmakci. What shall we teach our pants. In *Proc. of the Proc. Fourth Intl Symp. Wearable Computers (ISWC'00)*, 2000.
16. F. Miskelly. Assitive technology in elderly care. *Oxford Journals Medicine Age and Ageing*, 30(6):455–458, 2001.
17. M. Presser, A. Gluhak, D. Babb, L. Herault, and R. Tafazolli. eSENSE - capturing ambient intelligence for mobile communications through wireless sensor networks. In *Proc. of the 13th Intl. Conf. on Telecommunications*, pages 27–32, May 2006.
18. J. Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, March 1986.
19. J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kauffman, San Francisco, Calif., 1993.
20. C. Randall and H. Muller. Context awareness by analyzing accelerometer data. In *Proc. 4th Intl Symp. on Wearable Computers (ISWC'00)*, pages 175–176, Oct. 2000.
21. R. Rao, J. Eisenberg, and T. Schmitt. *Improving Disaster Management: The Role of IT in Mitigation, Preparedness, Response, and Recovery*. National Academies Press, 2007.
22. H. Si, Y. Kawahara, H. M. H. Kurasawa, and T. Aoyama. A context-aware collaborative filtering algorithm for real world oriented content delivery service. In *Proc. of ubiPCMM*, 2005.
23. D. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, and F. Wong. Sensay: A context- aware mobile phone. In *Proc. 7th International Symposium on Wearable Computers (ISWC)*, 2003.
24. S. Skaff, H. Choset, and A. Rizzi. Context identification for efficient multiple-model state estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2435–2440, Oct. 2007.
25. V. Stanford. Using pervasive computing to deliver elder care. *Pervasive Computing, IEEE*, 1(1):10–13, Jan-Mar 2002.
26. H.-L. Truong, L. Juszczyk, A. Manzoor, and S. Dustdar. ESCAPE - an adaptive framework for managing and providing context information in emergency situations. In *Proc. of the 2nd European Conf. on Smart Sensince and Context (EuroSSC'07)*, volume 4793 of *Lecture Notes in Computer Science (LNCS)*, pages 207–222, Berlin / Heidelberg, 2007. Springer.
27. E. Welbourne, J. Lester, A. LaMarca, and G. Borriello. Mobile context inference using low-cost sensors. In *Location- and Context-Awareness*, volume 3479 of *Lecture Notes on Computer Science (LNCS)*, pages 254–263. Springer-Verlag, 2007.