

Sensor Abstractions for Opportunistic Activity and Context Recognition Systems

Marc Kurz and Alois Ferscha

Institute for Pervasive Computing
Johannes Kepler University, A-4040 Linz, Austria
Phone: +43(0)732/2468/8527; Fax: +43(0)732/2468/8426
{kurz,ferscha}@pervasive.jku.at

Abstract. Pervasive environments are inherently characterized to draw from sensor infrastructures in order to become situation aware. Very recent technological evolutions of sensor hardware (e.g. for geolocation, acceleration, orientation, noise, light, humidity, chemical properties, etc.) have fertilized an explosive growth of sensor infrastructures, introducing whole new challenges for sensor software architectures like heterogeneity, redundancy and replaceability, fault tolerance, mobility, massive deployment, but most of all frequency of change and spontaneous availability. In this paper we address the very fundamental issue of exploiting spontaneous sensor configurations by introducing mechanisms for sensor self-description, goal-oriented sensing missions and dynamic sensor ensemble management. The concept of “sensor abstractions” is introduced, making the use of physical as well as immaterial sensors transparent from any technical sensor properties. An opportunistic sensor software architecture has been implemented, reversing former architecture principles (e.g. fusion of all available sensors) into a spontaneous, selective, utility driven involvement of sensors based on their sensing mission contribution potentials. The framework is implemented in OSGi, and demonstrated for activity recognition missions.

1 Introduction and Motivation

Activity and context recognition systems are an interesting research field in the area of pervasive and ubiquitous computing [13]. By applying (wireless) sensor networks which deliver data of the physical world and the persons living and acting in it [23] such systems interpret the data in terms of inferring activities and more generally the context [1,5,9] of persons and subjects in real world environments.

All such activity and context recognition systems have shared one major problem so far: the sensor deployment is application-specific and thus the mapping from sensor signals to context and activities has to be known and defined at *design time* [26,27]. This results in a static and predefined sensor infrastructure and provides no flexibility in the sensor deployment, the sensors actual positions (e.g. body-worn sensors have to be placed *exactly* at predefined positions) or the sensors availability (e.g. sensors can fail due to various reasons). Due to advancements

in electronics and wireless communication enabling technologies, the development of low-cost, low-power, small and multifunctional sensor nodes is possible [3]. As sensor systems nowadays are getting smaller and smaller and due to their capability of communicating with wireless interfaces, their mobility is increasing [7]. Due to this node-mobility on the one hand and the limited power resources resulting from the fact that the small and wireless devices are not permanently attached to a power socket [2,30] on the other hand, the infrastructure of a wireless sensor network can change permanently and cannot assure that a probably predefined sensor setup is static and stable over a certain amount of time.

Therefore we are working on the development of mobile opportunistic activity and context recognition enabling technologies. The term *opportunistic* in this context means that no predefined sensor infrastructure has to be defined at design time, the system rather makes best use (by applying a utility-driven approach [18]) of the available sensor devices according to a recognition goal, whereas when the system once started to operate, this sensor-setup is also not presumed to be fixed. The system reacts on changes in the sensor network topology at *runtime*. For example, new sensor nodes can be added to a running system, or existing nodes may be disconnected (e.g. they may run out of power). An opportunistic activity and context recognition system has to react properly at runtime to such dynamics in the sensor infrastructure. Furthermore, neither the network topology of sensor devices is fixed, nor the types of sensors that can be used within an opportunistic system are predefined, nor is the recognition goal hard-coded in the application at design time, it can be dynamically stated by users and/or applications in an abstract manner at *runtime*. Based on this high-level recognition goal, which has to be further processed and translated by the system to a coordinated machine-readable sensing mission, the available sensing devices configure themselves to so-called *ensembles* by applying self-* capabilities, like *self-description*, *self-organization* and *self-management* [12,17]. Different algorithms, methodologies and mechanisms have to be assessed on the way to a fully functioning system that enables opportunistic goal-driven activity and context recognition in the various steps of the recognition chain (like goal processing [18], formation of coordinated sensor ensembles and cooperative sensing [15,19], sensor signal acquisition, ensemble reconfiguration, data processing, feature extraction, etc.). The *OPPORTUNITY framework* that is currently being developed is (i) a prototypical implementation for evaluating and testing the approaches and (ii) a first step towards a ready-to-use middleware for building opportunistic activity and context recognition applications for different domains.

Figure 1 shows the concept and general activity and context recognition chain for an opportunistic architecture, whereat in this illustration two different goals ("*I need*"-*Request*) are passed on to the system. On top, the users and applications are located who formulate a recognition goal in an abstract manner that is handed to the system. This request is translated into a machine readable translation (the *sensing mission*). According to this mission, the available sensor nodes (at the very bottom of Fig. 1) organize and configure themselves to ensembles which are the best available set of sensors to execute this very sensing

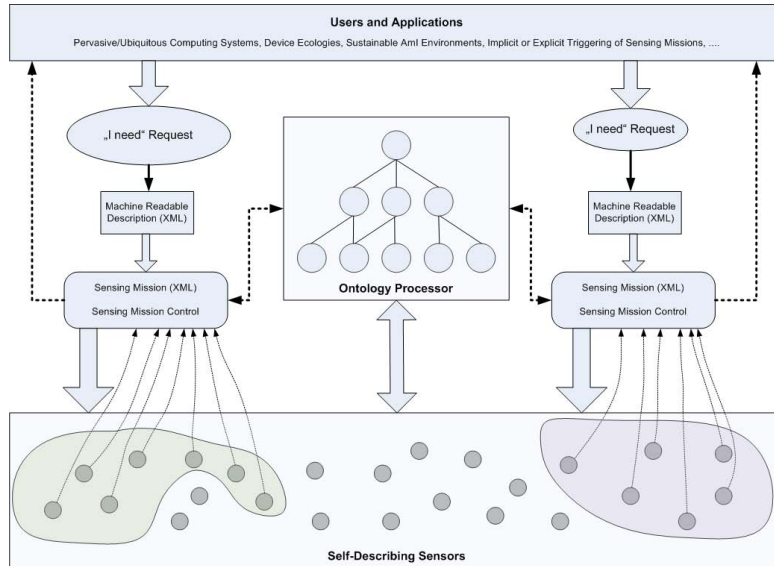


Fig. 1. Architectural concept for an opportunistic activity and context recognition system illustrating the general recognition chain and key-units in an opportunistic system

mission. If possible, according to the available sensor devices, a result for the recognition goal is returned to the requesting entity on top. During the execution of a sensing mission, the system reacts at runtime on topological changes (disconnects, connects or re-connects) in the sensor infrastructure. In the middle of the figure a knowledge-base and knowledge-processing unit (the *Ontology Processor*) is located which is indispensable in an opportunistic system for (i) goal processing and translation, (ii) for describing semantic relations from the sensor's capabilities to the capabilities required for a sensing mission and (iii) to configure coordinated sensor ensembles.

As an opportunistic activity and context recognition system does not specify its initial sensor configurations at design time and thus which types of sensor systems can be used in general, it neither does specify which hard- and software prerequisites that the sensor nodes should be capable of and it also does not specify which type of data the sensors should deliver. Therefore, we need easy to use and commonly accessible software abstractions of different types of sensors for enabling the prototypical implementation of the OPPORTUNITY framework. The purposes of the rest of this paper are to reason and explain (i) why we need sensor abstractions in an opportunistic activity and context recognition enabling framework, (ii) which types of sensors we want to abstract and how these abstractions and implementations can be done from a technical point of view, and (iii) how far we are currently with the implementations and what the next steps in the future are.

The remainder of the paper is structured as follows. The following Section 2 describes the reasons for developing sensor abstractions in detail and which different abstractions are valuable for an opportunistic system by providing examples for the identified types of sensors. Section 3 provides the identified sensor types for the OPPORTUNITY framework including implementation details and an example of a graphical visualization. In Sect. 4 related work is presented and in the last Sect. 5 you can find our conclusion and an outlook to future work.

2 Sensor Abstractions

Different sensor systems that can be roughly classified in the groups physical, logical or virtual devices [16] have different working characteristics, they measure different environmental quantities, deliver different types of data (e.g. acceleration, orientation, sound-level, temperature, humidity, etc.) and they might be accessible and controllable by a system or an application in different ways. Therefore, to have a simple and common standardized access to different sensor devices, wrappers are needed, which encapsulate hardware details and hide the low-level access details (e.g. direct memory access, data transmitting, etc.) that might be very appropriate for a specialized sensor device. Thus, a wrapper is a software abstraction of a sensor device that provides the complete functionality to a system by hiding the complexity. In the *Context Toolkit* (see [10] and [28]) the authors introduce the concept of *context widgets*, which is very similar to the aforementioned *wrappers*. *Dey et al.* define a context widget as [...] *software component that provides applications with access to context information from their operating environment*. The context widgets hide the complexity of the sensor systems, they abstract context information and they provide reusable building blocks for a system. All those characteristics can also be taken as mandatory for the sensor abstractions (the wrappers) in an opportunistic activity and context recognition system. Additionally, we further abstract the sensor systems in different types as given by their working-characteristic and their practicability and suitability, because dealing only with a set of physical sensor abstractions is not satisfactory in an opportunistic activity and context recognition system. The following Sect. 2.1 - Sect. 2.6 provide a description of the different sensor types that we have identified.

2.1 PhysicalSensor

One type that is inevitable are the common physical sensors (referred to as *PhysicalSensor*). According to [31] such sensors systems are small and low-power devices equipped with one or more sensing entities that measure and gather real-world information quantities. They are mostly composed of a core set of units, namely (i) a processing unit, (ii) a memory, (iii) a power supply and (iv) an interface for enabling (wireless) communication with a sink, other sensor nodes or another data processing unit of a system. Take a Sun SPOT (Small

Programmable Object Technology)¹ [29] and an InterSense InertiaCube3² as example. Both are physical devices consisting of different hardware units that are able to deliver different environmental data. The SunSPOT (working with a 180MHz 32-bit ARM920T core processor with 512K RAM and 4M Flash, using the 2,4GHz IEEE 802.15.4 radio standard for communication) is equipped with a 3-axis accelerometer, a temperature sensor and a light sensor and is comfortably accessible, programmable and adjustable by using the programming language Java. The InertiaCube3 includes a gyroscope, magnetometers and accelerometers with respect to gravity for 3DoF acceleration, angular velocity and orientation updates at a maximum of 180Hz. The InertiaCube3 sensor systems are accessible with an InterSense SDK dynamic link library (dll). To access and configure both of the sensors easily and equally in a system a wrapper abstracting the physical devices is required.

2.2 OnlineSensor

Another important source of information for an opportunistic activity and context recognition system that requires abstraction are online sources like web-services (referred to as *OnlineSensor*). This type does not rely on a physical device that is attached to a computer. It is rather some online accessible source of information. An example could be a webservice that provides regional weather information. If an opportunistic system requires weather information to successfully execute a recognition goal it could use such an online source if no physical device, that measures temperature and humidity, is currently available. Again, from the system's point of view an OnlineSensor is just another entity acting as sensor that delivers environmental data. The abstraction therefore has to deal with the connection to the remote source, the data acquisition and transmission and the data processing to the opportunistic activity and context recognition system.

2.3 PlaybackSensor

The third type of sensors is more or less a mixture of PhysicalSensors and OnlineSensors, the so called *PlaybackSensor*. Environmental data from a physical sensor device that has been recorded and stored at a previous point in time is replayed in the system and simulating this very (physical) sensor as being available. The replaying of the sensor data is being developed in the OPPORTUNITY framework to work with the same sampling rate and working characteristics as its physical pendant. In [24] and [25] the recording of a multimodal dataset in a kitchen scenario has been done using 72 sensor systems of 10 modalities. For developing, evaluating and testing opportunistic activity and context recognition enabling technologies, a rich dataset can be of high value. It is obvious that it is almost impossible to re-configure the sensor environment to be able to work

¹ <http://sunspotworld.com>

² http://www.intersense.com/InertiaCube_Sensors.aspx

with the same sensor network topology as in the initial (real-world) scenario. Therefore, the PlaybackSensors can be used to simulate a physical sensor device by replaying the recorded data by taking into account the recording characteristics. The location and nature of the (pre-recorded) data source is not fixed. A simple approach could be to use a simple text-, dat- or XML-file as data storage. The following Listing 1 shows a clipping from the content of the data file (stored as *.dat file) for the magnetic reed switches used in the experimental scenario (recorded at a sampling rate of 100Hz). The first two columns indicate the time when the piece of data was recorded (seconds and microseconds) and the following 16 columns provide the binary (On/Off) data for each of the used magnetic reed switches. A more sophisticated approach for storing the pre-recorded data could use a database or even an online/remote source with distributed data locations. Again, from the systems point of view, the type of storing and accessing the data to enable a sensor simulation is abstracted and thus hidden.

1243345571	488238	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
1243345571	488522	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
1243345571	504245	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
1243345571	504482	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0
1243345571	520270	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0
1243345571	536281	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0
1243345571	536349	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0
.

Listing 1. A clipping from the data file from the magnetic reed switches, recorded at the experimental dataset recording sessions [24, 25]

2.4 SyntheticSensor

A *SyntheticSensor* also simulates a physical device, but the difference to the group of PlaybackSensors is the fact that such a synthetic device does not rely on a pre-recorded data source. It rather tries to autonomously generate data and simulate a certain sensors behavior and working characteristics. Different types of data generation mechanisms are applicable for this type of sensors, mainly depending on their field of application. The synthetic device could only be used for generating random (but still plausible and authentic) data that can be utilized for evaluation, testing and demonstration purposes. In this case the data generation entity in a SyntheticSensor relies on a random generator that provides a datastream following a set of given working characteristics of the sensors physical antetype (e.g. sampling rate). Another case when a synthetic device could be useful is to deal with the temporary filling of gaps in the sensor data stream. Take a physical accelerometer that is part of a configured ensemble as example. It delivers data to an opportunistic activity and context recognition system that uses this data in real-time. When this very sensor suddenly slips or disconnects (e.g.

it runs out of power, or simply moves out of the range of the system), this could make a complete reconfiguration of the running sensor ensemble according to a recognition goal necessary. In case that the sensor disappears from the network topology permanently, the reconfiguration of the system might be inevitable. But in case that the sensor reconnects after a few seconds, the reconfiguration of the ensemble might be an effort that could have been avoided. Therefore, a *SyntheticSensor* could act as a placeholder for at least a few seconds for a physical device to avoid the time- and power-consuming reconfiguration effort. The data generation entity in this case could generate data that is constructed from the historical data that has been delivered by the physical sensor, simply to bridge a possible gap in the datastream and to keep the (running) system stable.

2.5 HarvestSensor

A *HarvestSensor* is more or less a sub-type composition of the *PlaybackSensor* and the *PhysicalSensor*, but nevertheless worth mentioning it as independent type. A *HarvestSensor* in our understanding is a physical device that autonomously collects environmental data, stores this data locally on some internal memory and provides this recorded data when the system puts the sensor in replay-mode. An example for such a device is the GT3X sensor manufactured by *ActiGraph*³. It provides activity measures, like steps taken and energy expenditure from persons, and is equipped with a 4MB flash memory, that is capable of storing data for more than one year without having the system attached to a power supply.

2.6 ProxySensor

The last identified sensor type that requires an abstraction in an opportunistic activity and context recognition system is the *ProxySensor*. This type acts as an arbiter much like it is known in a computer network, respectively as surrogate as known from the design patterns in software engineering [14]. A proxy acts as surrogate for another sensor by providing an interface for accessing this system.

2.7 Summary

The aforementioned sensor abstractions are currently being developed in the OPPORTUNITY framework. Main focus lies on the implementation of the sensor systems that have been used in the kitchen-dataset recording [20,24]. The following Tab. 1 provides an overview and summary of the six identified sensor types.

The next Sect. 3 provides a description of the framework, an overview on implementation details of the sensor abstractions and shows exemplary screen dumps from visualizations of the different sensor types.

³ <http://www.theactigraph.com/>

Table 1. Sensor Type Overview

Sensor Type	Purpose	Example
<i>PhysicalSensor</i>	hide complexity and hardware details of physical sensor device	use different sensor systems (e.g. SunSPOT and InterSense InertiaCube3) with a common and easy interface
<i>OnlineSensor</i>	hide connection details and data acquisition from online information sources	an online weather-webservice can provide valuable local weather information
<i>PlaybackSensor</i>	replay pre-recorded sensor data by simulating a physical device	reconfigure and simulate a test-setup virtually
<i>SyntheticSensor</i>	simulate a physical sensor by generating plausible data	bridge a gap of a physical accelerometer until it reconnects to the system for at least a few seconds
<i>HarvestSensor</i>	provide autonomously collected environmental data to a system	sensor system collects data and stores it locally on its memory
<i>ProxySensor</i>	act as a surrogate for another sensor system	a new sensor can be used that is not yet implemented as one of the above types

3 Providing Sensor Abstractions

In the previous Sect. 2 we have introduced six different types of sensor abstractions that have to be differentiated in an opportunistic activity and context recognition system. This section provides implementation and technical details as the abstractions are part of the OPPORTUNITY framework that is currently being developed. Figure 2 summarizes the identified types of sensor abstractions, whereas this figure illustrates the main idea behind this concept. At the very bottom the different types of sensors are located, their software wrappers that encapsulate and hide hardware details are located in the containers directly above. All types of sensors are directly derived from the general type *Sensor*, located at the very top of the figure. The system internally only deals with data delivering entities over a common and easy accessible interface that is equal for all types of sensors (as all of them can be allocated to the general type *Sensor*). Basic and established concepts in object-oriented software design, namely *inheritance* and *abstraction*, enable a sophisticated concept of multi-typed sensor abstraction in an opportunistic activity and context recognition system. The containers hide the complete complexity, the hardware details in case of physical sensor devices, the accessing and connection methods in case of online information sources, or the data generation or replaying mechanisms in case of playback and synthetic devices. From the system's point of view a data delivering source is accessed as *Sensor* with one common interface.

The OPPORTUNITY framework is - as already mentioned - a prototypical implementation of an opportunistic activity and context recognition system used

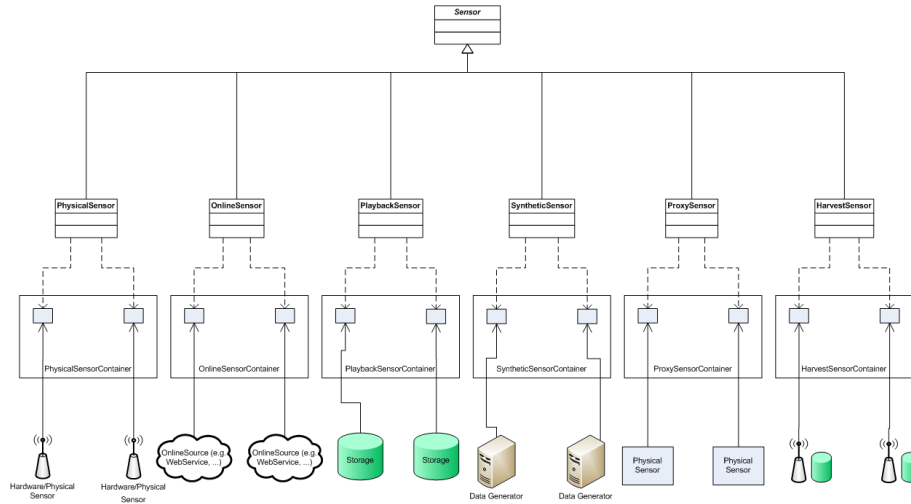


Fig. 2. The six different sensor types and the concept of realization in an opportunistic activity and context recognition system

for evaluation, testing and development. The framework is implemented using the Open Services Gateway Initiative (OSGi)⁴ [4,6] and acts as a runtime environment together with a code base and libraries, able to execute autonomously on a target platform. Reasons for implementing the framework in OSGi are (i) the universality of code deployment, (ii) the life cycle management capabilities of OSGi, (iii) the modularity and component oriented paradigm, (iv) the portability and thus compatibility with various different hardware platforms, and (v) the ability to install, restore, start and stop applications at runtime. Generally, OSGi is a collection of bundles that can be interconnected using the OSGi Wire Admin service specification⁵, following the producer consumer paradigm to query and propagate data internally from sensors (producers) to sensing mission(s) (consumers) or even from one sensor to another sensor if a direct communication is necessary (e.g. to share localization information, or to negotiate configuration issues).

Furthermore, for demonstrating the actions, the data flows and the ensemble (re-)configurations in an opportunistic activity and context recognition process over time, a real-time graphical visualizer is included in the OPPORTUNITY framework. It displays the currently available sensor device(s) and sensing mission(s), and how they are connected. Figure 3 shows two screen dumps of sensor configurations trying to execute sensing mission(s) by delivering environmental data to the consumer(s). Every colored ellipse illustrates one sensor device that is currently available in the environment and the red rectangle is the visualization unit for a sensing mission. The bigger, gray-colored rectangle at the very

⁴ <http://www.osgi.org>

⁵ <http://www.osgi.org/About/Technology>

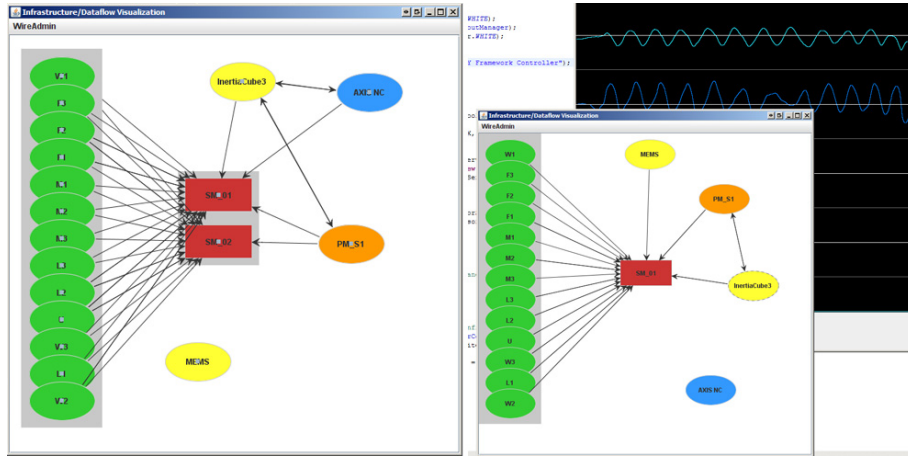


Fig. 3. Screen dumps of two examples showing the visualization of (i) Playback-, (ii) Online-, (iii) Physical- and (iv) Synthetic-Sensors in the OPPORTUNITY Framework

left side of both examples, that holds 13 sensor devices, and the gray rectangle in the left screen dump that holds the two red rectangles means that all of the included are running in the same bundle of the framework (in this case the magnetic reed switches from type PlaybackSensors, and two sensing missions). The arrows in the illustration show the datastreams from the sensors to the sensing mission and between sensors themselves. In the left example two different sensing missions are active that are currently executed. We can see 13 sensors of type *PlaybackSensor* (the green bubbles), whereas all of them are magnetic reed switches that replay prerecorded data. Furthermore, the example shows one sensor of type *OnlineSensor* ("AXIS NC"), which is a network camera and illustrated as blue ellipse, two *PhysicalSensor* ("MEMS" and "InertiaCube3") devices, namely a microphone measuring the noise level and one InertiaCube3 sensor located in a shoe sole that can be used for gait recognition [11], both visualized as yellow bubble and one *SyntheticSensor* ("PM.S1") which is a 3-axis accelerometer synthetic device that delivers random generated acceleration data (displayed as orange-colored bubble). Not every sensor in this example delivers its measured environmental quantities to every sensing mission. According to the sensors capabilities, their locations (which are included in the sensors self-descriptions) and the recognition goals requirements the ensembles are configured to have the best available set of sensors grouped. Furthermore, as we can see in both examples, sensor nodes can share information among themselves, e.g. for sharing localization data, or for configuration purposes. In the left-hand side example, the microphone (one of the yellow bubbles) is not connected to a mission or another sensor node. It would be available and ready to deliver data but it is currently not needed. The screen dump on the right side shows a similar example. There, only one sensing mission is currently active, whereas the sensor

devices are the same as in the example on the left. However, as the sensing mission in the second example has other requirements, the ensemble configuration is different. Additionally, we can also see a clipping in the screen dump of the graphical visualization of the data delivered from the InterSense InertiaCube3 sensor (the 3-axis orientation diagram). In case of the types *PlayBackSensor* and *SyntheticSensor*, the OPPORTUNITY framework furthermore provides the functionality to connect, disconnect and reconnect sensor devices at runtime. This is necessary, to create flexible sensor environments that can be use to test and evaluate the ensemble (re-)configuration and more general the execution of sensing missions with different scenarios.

4 Related Work

A lot of related middleware solutions, systems and frameworks exist that enable autonomic, self-organizing computing in pervasive environments, and especially wireless sensor networks with respect to become situation aware. This section provides a summarization of the most valuable solutions and discusses their shortcomings for the opportunistic activity and context recognition problem domain.

The *TinyLime Middleware* [8] supports the development of sensor network applications. This approach departs from the traditional setting where sensor systems deliver measured environmental quantities to one centralized entity. Instead, the middleware supports multiple stations to access and collect data from different sensors. Nevertheless, the authors clearly define that the sensors used with the middleware do not need to be able to communicate with each other. Furthermore, neither the types of supported sensors, nor the integration and abstraction, nor the highly dynamic sensor infrastructure is mentioned, which are key issues in opportunistic systems.

Madden et al. present in [21] the *Tiny AGgregation Service for Ad-Hoc Sensor Networks (TAG)*. The main contribution of *TAG* is the expression of declarative queries that are distributed and executed in wireless sensor networks. This approach is developed for networks of TinyOS sensor nodes which is a limitation that does not meet the requirements of dynamic multi sensor usage by applying sensor abstractions in an opportunistic system.

In [22] the authors present a middleware suitable for dynamic network environments. The approach called *TOTA (Tuples On The Air)*, supports adaptive context-aware activities in pervasive scenarios. Basic principle is to rely on tuples, which are propagated across a network on base of application specific rules. Nevertheless, this approach does not sufficiently meet the requirements of an opportunistic system, like the spontaneous availability of resources, therefore exploiting spontaneous configurations and the goal-oriented sensing capabilities.

Some of the aforementioned approaches are capable of spreading queries to the sensor systems, others propose a non-centralized approach of sensor signal acquisition, which are all important issues in an opportunistic system. Furthermore, as we investigate in building a highly dynamic and flexible system, we cannot presume a static set of sensors, therefore we need the abstracted

sensor types solution and as we also do not rely on a fixed recognition goal, the framework must be capable of goal processing and goal-oriented sensing.

5 Conclusion and Future Work

This paper identifies six (*PhysicalSensor*, *PlaybackSensor*, *OnlineSensor*, *SyntheticSensor*, *HarvestSensor* and *ProxySensor*) different types of sensors applicable in an opportunistic activity and context recognition system and describes how they are abstracted. These different sensor abstractions are necessary as an opportunistic system does not specify at design time (i) which sensor system can/shall be used, (ii) how the used sensor devices have to be placed and located and (iii) how they have to be configured according to a recognition goal as this is also not static and fixed. Furthermore, we introduce the OPPORTUNITY framework which is a flexible runtime environment and a prototypical OSGi implementation of an opportunistic activity and context recognition enabling framework. The system uses the sensor abstractions for hiding the complexity and to have a common interface for using different sensor types. For demonstration and testing purposes the framework also includes a graphical visualization of the sensor environment, displaying the currently available sensor devices, the sensing missions that are derived from a recognition goal and the data flows among the sensor devices and the sensing mission(s). With our sensor abstractions that enable the usage of different sensor types and the development of an opportunistic prototypical framework the implementation and realization of a set of a new generation activity and context recognition applications will be possible.

Future work concerns the data generation method in the synthetic devices to enable the gap bridging mechanism to act as a placeholder for a temporary disconnected physical device that is mentioned above. Another important thing that we are currently working on is the processing and translation from a recognition goal formulated in an abstract manner by a user or an application to the machine-readable sensing missions. Therefore, we are working on a goal-description language and appropriate mechanisms to be as flexible and dynamic as possible.

Acknowledgments. The project OPPORTUNITY acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 225938.

References

1. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a Better Understanding of Context and Context-Awareness. In: Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing, HUC 1999, pp. 304–307. Springer, London (1999)
2. Agre, J., Clare, L.: An integrated architecture for cooperative sensing networks. *Computer* 33(5), 106–108 (2000)

3. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. *IEEE Communications Magazine* 40(8), 102–114 (2002)
4. Alliance, O.: OSGi Service Platform Release 4. architecture p. 9 (2000)
5. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.* 2(4), 263–277 (2007)
6. Bartlett, N.: OSGi in practice. *Bd* (January 11, 2009)
7. Chong, C.Y., Kumar, S.: Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE* 91(8), 1247–1256 (2003)
8. Curino, C., Giani, M., Giorgetta, M., Giusti, A., Murphy, A.L., Picco, G.P.: Mobile data collection in sensor networks: The tinyline middleware. *Pervasive and Mobile Computing* 1(4), 446–469 (2005), <http://www.sciencedirect.com/science/article/B7MF1-4H9GRV7-1/2/bf4f596735fdf93adcf4a0ecfd6255a> (special Issue on PerCom 2005)
9. Dey, A.K.: Understanding and using context. *Personal Ubiquitous Comput.* 5(1), 4–7 (2001)
10. Dey, A.K., Abowd, G.D.: *The Context Toolkit: Aiding the Development of Context-Aware Applications*, pp. 434–441. ACM Press, New York (1999)
11. Doppler, J., Holl, G., Ferscha, A., Franz, M., Klein, C., dos Santos Rocha, M., Zeidler, A.: Variability in foot-worn sensor placement for activity recognition. In: *Proceedings of the 13th International Symposium on Wearable Computers (ISWC 2009)*, Linz, Austria, September 4-7. IEEE Computer Society Press, Los Alamitos (2009)
12. Dressler, F.: *Self-Organization in Sensor and Actor Networks*. John Wiley & Sons, Chichester (2007)
13. Ferscha, A.: What is Pervasive Computing, pp. 93–108. Universitätsverlag Rudolf Trauner (June 2003)
14. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design patterns: elements of reusable object-oriented software*. Addison-wesley, Reading (1995)
15. Garcia, M., Lloret, J.: A cooperative group-based sensor network for environmental monitoring. In: Luo, Y. (ed.) *Cooperative Design, Visualization, and Engineering*. LNCS, vol. 5738, pp. 276–279. Springer, Heidelberg (2009)
16. Indulska, J., Sutton, P.: Location management in pervasive systems. In: *Proceedings of the Australasian Information Security Workshop Conference on ACSW frontiers 2003, ACSW Frontiers 2003*, pp. 143–151. Australian Computer Society, Inc., Darlinghurst (2003)
17. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. *Computer* 36(1), 41–50 (2003)
18. Kephart, J.O., Das, R.: Achieving self-management via utility functions. *IEEE Internet Computing* 11(1), 40–48 (2007)
19. Li, W., Kamil, Y., Manikas, A.: A wireless array based cooperative sensing model in sensor networks. In: *Global Telecommunications Conference on IEEE GLOBECOM 2008, December 4-30*, pp. 1–6. IEEE, Los Alamitos (2008)
20. Lukowicz, P., Pirkl, G., Bannach, D., Wagner, F., Calatroni, A., Förster, K., Holleczeck, T., Rossi, M., Roggen, D., Troester, G., Doppler, J., Holzmann, C., Riener, A., Ferscha, A., Chavarriaga, R.: Recording a complex, multi modal activity data set for context recognition. In: *Proceedings of the 1st Workshop on Context-Systems Design, Evaluation and Optimisation (CosDEO 2010)*. VDE Publishing House, Hannover (February 2010)
21. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tag: a tiny aggregation service for ad-hoc sensor networks. In: *IN OSDI (2002)*

22. Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications with the tota middleware. *ACM Transactions on Software Engineering and Methodology*, 18(4), Article 15 263–273 (2009)
23. Puccinelli, D., Haenggi, M.: Wireless sensor networks: applications and challenges of ubiquitous sensing. *IEEE Circuits and Systems Magazine* 5 (2005)
24. Roggen, D., Calatroni, A., Rossi, M., Holleczeck, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirkel, G., Ferscha, A., Doppler, J., Holzmann, C., Kurz, M., Holl, G., Chavarriaga, R., Creatura, M., del Milln, R.: Collecting complex activity data sets in highly rich networked sensor environments. In: *Proceedings of the Seventh International Conference on Networked Sensing Systems (INSS)*, Kassel, Germany (June 2010)
25. Roggen, D., Calatroni, A., Rossi, M., Holleczeck, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirkel, G., Wagner, F., Ferscha, A., Doppler, J., Holzmann, C., Kurz, M., Holl, G., Chavarriaga, R., Creatura, M., del Milln, R.: Walk-through the opportunity dataset for activity recognition in sensor rich environments (May 2010), <http://vimeo.com/8704668>
26. Roggen, D., Förster, K., Calatroni, A., Bulling, A., Holleczeck, T., Troester, G., Lukowicz, P., Pirkel, G., Bannach, D., Ferscha, A., Riener, A., Holzmann, C., Chavarriaga, R., del Milln, R.: OPPORTUNITY: activity and context awareness in opportunistic open-ended sensor environments. Poster at the 1st European Future Emerging Technologies Conference (FET 2009), Prague, Czech Republic (April 2009)
27. Roggen, D., Förster, K., Calatroni, A., Holleczeck, T., Fang, Y., Troester, G., Lukowicz, P., Pirkel, G., Bannach, D., Kunze, K., Ferscha, A., Holzmann, C., Riener, A., Chavarriaga, R., del Milln, R.: OPPORTUNITY: Towards opportunistic activity and context recognition systems. In: *Proceedings of the 3rd IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC 2009)*, June 2009, IEEE CS Press, Kos (2009)
28. Salber, D., Dey, A.K., Abowd, G.D.: The context toolkit: aiding the development of context-enabled applications. In: *Proceedings of the SIGCHI Conference on Human factors in Computing Systems, CHI 1999*, pp. 434–441. ACM, New York (1999)
29. Smith, R.B.: Spotworld and the sun spot. In: *Proceedings of the 6th International Conference on Information Processing in Sensor Networks, IPSN 2007*, pp. 565–566. ACM, New York (2007)
30. Ververidis, C.N., Polyzos, G.C.: Service discovery for mobile ad hoc networks: A survey of issues and techniques. *IEEE Communications Surveys and Tutorials* 10(1-4), 30–45 (2008)
31. Yick, J., Mukherjee, B., Ghosal, D.: Wireless sensor network survey. *Computer Networks* 52(12), 2292–2330 (2008)