J Med Syst (2016) 40: 200
DOI 10.1007/s10916-016-0549-7

PATIENT FACING SYSTEMS

On the Design of Smart Homes: A Framework for Activity Recognition in Home Environment

Franco Cicirelli[1] · Giancarlo Fortino[1,2] · Andrea Giordano[1] · Antonio Guerrieri[1] · Giandomenico Spezzano[1] · Andrea Vinci[1]

Received: 22 February 2016 / Accepted: 12 July 2016 / Published online: 28 July 2016

Abstract A smart home is a home environment enriched with sensing, actuation, communication and computation capabilities which permits to adapt it to inhabitants preferences and requirements. Establishing a proper strategy of actuation on the home environment can require complex computational tasks on the sensed data. This is the case of activity recognition, which consists in retrieving high-level knowledge about what occurs in the home environment and about the behaviour of the inhabitants. The inherent complexity of this application domain asks for tools able to properly support the design and implementation phases. This paper proposes a framework for the design and implementation of smart home applications focused on activity recognition in home environments. The framework mainly relies on the *Cloud-assisted Agent-based Smart home Environment (CASE)* architecture offering basic abstraction entities which easily allow to design and implement Smart Home applications. CASE is a three layered architecture which exploits the distributed multi-agent paradigm and the cloud technology for offering analytics services. Details about how to implement activity recognition onto the CASE architecture are supplied focusing on the low-level technological issues as well as the algorithms and the methodologies useful for the activity recognition. The effectiveness of the framework is shown through a case study consisting of a daily activity recognition of a person in a home environment.

Keywords Smart homes · Internet of things · Activity recognition · Cloud computing · Multi agent system · Analytics · Wireless sensor and actuator networks · Wearable wireless body sensor networks

This article is part of the Topical Collection on *Patient Facing Systems*

✉ Andrea Giordano
giordano@icar.cnr.it

Franco Cicirelli
cicirelli@icar.cnr.it

Giancarlo Fortino
g.fortino@unical.it

Antonio Guerrieri
guerrieri@icar.cnr.it

Giandomenico Spezzano
spezzano@icar.cnr.it

Andrea Vinci
vinci@icar.cnr.it

[1] CNR – National Research Council of Italy, Institute for High Performance Computing and Networking (ICAR), Via P. Bucci 7-11C, 87036, Rende, Italy

[2] Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica, Università della Calabria, 87036 Rende, Italy

Introduction

In the context of the Internet of Things (IoT) [1] many devices have been enhanced with respect to their ordinary role to be proactive and collaborative with other devices [2]. In this background, the Smart Homes [3] are advancing as a disruptive trend in the literature. Regarding Smart Homes, many progresses have been done in the last few years on several fields: *Home Automation and Domotics* [4], *Energy Optimization* [5], *Distributed Sensing and Actuation* [6, 7], *Activity Recognition* [8], *Ambient Assisted Living* [9, 10], *Indoor Positioning Systems* [11], and *Home Security* [12].

The development of Smart Homes, and in general Smart Environments, introduces several challenges [13] regarding the integration of heterogeneous systems and technologies, scalability, reliability and functional extensibility. In particular, functional extensibility refers to the problem of adding new services to the set of the existing ones already working in a given home environment. Despite the big interest around Smart Homes, the currently adopted solutions lack in design and development guidelines, approaches for the integration of heterogeneous services and infrastructures, as well as ICT solutions which allow an holistic Smart Homes development. Moreover, even though home environments are naturally distributed, most of the actual Smart Home implementations rely on centralized approaches. Finally, many realized services are developed as independent monolithic blocks, and the interaction among them is not considered.

This paper proposes a framework for the development of Smart Homes featuring activity recognition. As a distinguishing feature, proposed framework naturally permits to address the above listed challenges which are related to the design and implementation of general Smart Environments. The framework relies on a platform, called *Cloud-assisted Agent-based Smart home Environment (CASE)*, that easily allows the distributed sensing and actuation in Smart Home environments. CASE is a layered architecture which permits the physical entities (i.e. sensors and actuators) to be transparently integrated with a distributed multi-agent system and a cloud infrastructure. CASE is able to manage both environmental sensors, spread on the home environment, and mobile wearable sensors worn by the home inhabitants.

CASE architecture enables agents [14–16], which implement specific applications and analytics services, to be dynamically uploaded either on a cloud system, for exploiting its huge computational resources, or close to the physical entities fostering real-time computation. The capability of supporting dynamic deployment of agents naturally fosters functional extendibility of the system.

Real-time computation is carried out on a network of computing nodes deployed over the home area. Further computing nodes can be dynamically added thus favouring system scalability. The architecture is able to hide the heterogeneity of both communication protocols and physical devices (e.g. sensors, actuators, or even complex objects) which are connected to the computing nodes. The devices connected to a node are directly managed by the agent residing on the node itself thus promoting computation near the source of data. Important side benefits of such computation model include: (i) a faster reactivity to events which can be managed where they are generated, (ii) a better exploitation of communication bandwidth, as data are locally processed and only the required aggregated information is propagated across the system, (iii) an increase of reliability and scalability, since the proposed architecture fosters the use of distributed algorithms.

One of the major application scenario that the platform is able to support is related to Activity Recognition in Smart Home environments. *Activity Recognition* [8] using data mining techniques can be successfully adopted for giving support to elder people living alone that require a 24/7 assistance. Indeed, applying these techniques on data coming from the home environment permits to recognize and analyse human activities in real time and, eventually, to send alarms to relatives or doctors of the monitored person.

Together with the description of the framework, the paper focuses on how the CASE platform can be exploited for supporting a general activity recognition schema in Smart Homes. Information about technological solutions (i.e. related to communication protocols and wearable and environmental sensors) available for implementing activity recognition are also provided. A case study has been drawn out which aims to underline the framework features and benefits in this field.

The reminder of the paper is organized as follows: "Related work" introduces the related work, "Cloud-assisted agent-based smart home environment" presents the CASE architecture and shows how it can be used in the activity recognition field, and "Case study: Designing a smart home environment" details the chosen case study, gives some hints on the technological solutions for developing Smart Home applications, and provides some experimental results. Finally, some conclusions and future work are drawn.

Related work

Nowadays, Smart Homes represent an important research field both in academia and in industry [3]. One of the major challenges in the realization of these applications is supporting interoperability among various heterogeneous devices and their proper deployment. Thus, the need for new architectures and frameworks - supporting both smart control and actuation - has been identified by many researchers. Works about this topic are described in the first part of this section. As introduced in the previous section, an important application scenario for Smart Homes is in-home activity recognition [17]. Related works on this field is presented later in this section.

Frameworks for smart home environments The research community has presented several proposals regarding middlewares and frameworks for the rapid prototyping of smart environments in general, or smart homes in particular.

As an example, authors in [18] presented *Voyager*, that is a framework conceived for the support to the realization

of smart environments. Voyager relies on the exploitation of small bluetooth devices, which are user programmable, used to give intelligence to realized systems.

The work in [19] introduces *JCAF* (Java Context-Awareness Framework), which is an extensible framework that supports the creation of context-aware in-home applications. The main components of the framework are *clients*, *actuators*, *monitors*, and *services*. JCAF allows all of them to be dynamically added, updated or removed at runtime.

Authors of [20] show *Gaia*, a middleware supporting the control of heterogeneous resources in a physical space. Gaia allows the developers to see collections of individual devices as a whole by introducing programmable *active spaces*. Active spaces are programmable through a scripting language called *LuaOrb* [21].

The *Syndesi* framework is introduced in [22]. Syndesi exploits Wireless Sensor Newtorks to create personalized smart environments. The environment augmented with the Syndesi framework allows to identify people and to realize specific actions which are based on people profiles.

In-home activity recognition Several works have been realized in the direction of activity recognition in home environments.

Authors of [23] introduce an activity recognition system for single person smart homes using *active learning* [24]. They apply data mining techniques to cluster in-home sensor samplings so that each cluster represents a human activity. The users have to label each cluster with the corresponding activity to allow the system learning how to recognize future activities. The system has been designed to detect not only single but also overlapping activities.

In the context of the CASAS project, authors of [8, 25] present an automatic approach to activity discovery and monitoring for assisted living in a real world setting. It does not use supervised methods for activity recognition, but it automatically discovers activity patterns. Moreover, the system is designed to discover variations in such patterns and is also able to handle real life data by dealing with different sensor problems.

The work in [26] shows a prototype used to forecast the behaviour and wellness of the elderly by monitoring the daily usages of appliances in a smart home. Elderly people perform activities at regular intervals of time. When daily activities are performed regularly, it means that the elderly man/woman is in a state of wellness. In this work, Predictive Ambient Intelligence techniques are used involving the extraction of patterns related to sensor activations.

In [27] authors introduce an ontology-based hybrid approach to activity modeling that combines domain knowledge based model specification and data-driven model learning. Central to the approach is an iterative process that begins with "seed" activity models created by ontological engineering. The "seed" models are deployed, and subsequently evolved through incremental activity discovery and model update. The rationale of this work is to provide (on single-user single-activity scenarios) generic activity models suitable for all users and then create individual activity models through incremental learning.

With respect to the reviewed literature, the proposed CASE framework is novel as it allows:

- the exploitation of both environmental and weareable sensors in developing smart home applications;
- the exploitation of both reactive computation on distributed computational nodes spread in the home environment as well as cloud-based computation, both for sensing and actuation purposes;
- the exploitation of a distributed and dynamic computational paradigm which foster extendibility, scalability and reliability;
- the development of applications which can realize activity discovery and recognition, by purposely exploiting all the features of the architecture;
- the possibility to dynamically evolve the classification model by adding or updating classifiers as soon as new activities are discovered;
- an easy integration of different kinds of smart home applications with the activity recognition one.

Cloud-assisted agent-based smart home environment

In this section we introduce the basic concepts of the *Cloud-assisted Agent-based Smart home Environment (CASE)* architecture which allows to easily design and develop optimized complex smart home applications. CASE architecture is particularly suitable in those scenarios where complex human activities (e.g. cooking, sleeping etc.) need to be figured out from raw sensors data. In our approach, we distinguish two kinds of sensors: environmental sensors and mobile sensors. Environmental sensors are typically placed in fixed positions in the home environment while mobile sensors are, for example, worn directly by humans, and their positions dynamically change when people move around the home area. Combining both environmental and human-related data with analytics methods allows discovering and analysing complex activities which usually can not be derived solely from environmental data neither from the human-related ones.

In the next section, the CASE architecture is described in order to better clarify the rationale and the basic concepts underlying the proposed approach.

CASE architecture

The CASE architecture can be seen as composed by three coarse-grained layers (see Fig. 1):

IoT Layer. It consists of the collection of all sensors, actuators or more complex devices which are spread across the home environment or worn by humans.

Virtualization Layer. It is the layer focused on abstracting the physical part so to simplify the upper layers work by supplying a uniform interface to heterogeneous IoT devices. This layer also addresses all the low level issues such as connectivity, reliability and resilience.

Analytics Layer. The above described layers allow to develop simple monitoring on the sensed data as well as simple actuation strategies. Anyway, passing from the raw sensory data to a complex model of reality, which is able to recognize high level human activities, requires the adoption of complex algorithms which may fall into several research fields such as data-mining, swarm intelligence and so on. Since their complexity, such algorithms frequently need the high availability of computational resources that the cloud provides. All these kinds of algorithms/techniques are called hereafter *Analytics Services*.

By using the CASE architecture it is possible to build applications related to the smart home context. In particular, the CASE effectiveness in the case of monitoring of human activities for recognizing critical or dangerous situations will be shown in "Case study: Designing a smart home environment".

The main issues addressed by the CASE architecture concern: the heterogeneity of physical elements (i.e. sensors, actuators and other devices), the real-time distributed nature of the smart home, and the inherent complexity of the analytics. CASE hides heterogeneity by means of the *IoT Devices Abstraction Layer* which *wraps* the physical entities that are managed through a well-defined *Object Interface* as detailed in "IoT devices abstraction layer". The real-time distributed nature of these kinds of application and the inherent complexity of the analytics involved are addressed adopting a distributed multi-agent system (MAS) and the cloud technology. The distributed MAS aims to keep the computation close to the information sources (i.e. the physical devices), thus decreasing the need for remote communications so as to foster fast actuations based on real-time changes in the environment. The Cloud technology may be suitably exploited to carry out heavy computational tasks which could be required for running complex analytics.

CASE architecture is composed by a set of computing nodes of two kinds: *fixed* and *mobile*. The fixed computing nodes are single-board computers like Raspberry PIs or BeagleBoards. They manage the fixed environmental sensors and actuators to which are connected. The single-board computers can be effectively placed across the home area because of their low costs, low energy consumptions and small dimensions. The mobile computing nodes are achieved using smartphones. This is a valuable solution because nowadays smartphones are very cheap and commonly used. Even if people (especially elderly) not necessarily want to wear sensors [28], we think that they can accept to carry with them both sensors and a smartphone, once they understand the advantages (e.g. in terms of prolonging their independence over time) that can be achieved by using applications built on top of our platform.

Each computational node contains both the IoT Devices Abstraction layer and the Agent Server. IoT Devices

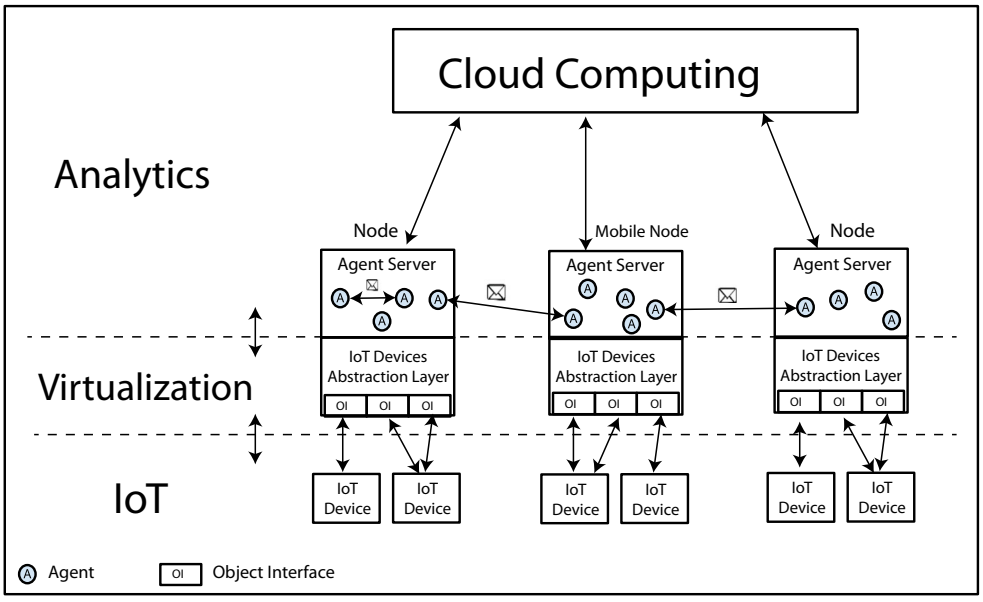

Fig. 1 The CASE architecture

Abstraction layer and Agent Server are located in the same computing nodes in order to guarantee that agents exploit the local physical part. Instead of transferring data to a central processing unit, processes are transferred (by injecting fine-grained agents onto the nodes) toward the data sources. As a consequence, a minimum amount of data needs to be transferred over a long distance (i.e. toward remote hosts) so local access and computation is fostered in order to achieve good performance and scalability.

IoT devices abstraction layer

The IoT Devices Abstraction layer aims to hide heterogeneity by supplying a well-established *Object Interface* (OI) permitting the physical parts to be suitably integrated with the rest of the system. All the physical devices must be arranged in a set of *Virtual Object*s (VOs) implementing OI [29].

A VO could be defined as a collection of physical entities like sensors and actuators, together with their computational abilities. It can be composed of just a simple sensor or it can be a more complex object that includes many sensors, many actuators, computational resources like CPU or memory and so on.

In general, VO outputs can be represented by *punctual values* (e.g. the temperature at a given point of a room) or *aggregated values* (e.g. the average of moisture during the last 8 hours). Also, the values returned by VOs could be just the measurement of sensors or could be the result of complex computations (e.g. the temperature at a given point of space computed by means of interpolation of the values given by sensors spread across the environment).

Furthermore, a VO could supply actuation functionality by changing the environment on the basis of external triggers or internal calculus.

These different kinds of behaviour that VO can expose must be taken into account. VO is therefore conceived as a complex object that can read and write upon many simple physical resources. More in detail, we consider that each VO exposes different *functionalities*. Each functionality can be either sensing or actuation and can be refined by further parameters that dynamically configure it.

The previous assumption leads to the definition of *resource* as the following *triplet*:

$$[VOId, FunctionId, Params]$$

Where `VOId` uniquely identifies the VO, `FunctionId` identifies the specific functionality and `Params` is an ordered set of parameter values that configure the functionality.

Besides read and write operations (i.e. sensing and actuation), it is provided for VOs to be able to manage events that occur in the physical part. Agents can define events by supplying *logical rules* in which one or more VOs could be involved [29]. The IoT Devices Abstraction Layer is in charge of notifying the agents of the events' occurrences.

CASE multi-agent system

The Multi Agent component of the CASE architecture is made up of the following entities: *Agents*, *Messages*, the *Agent Server* and the *Deployer*. Figure 2 shows these entities and how they interact among themselves and with the *IoT Devices Abstraction Layer*.

An *Agent* is an autonomous entity which executes its own behaviour interacting with other agents via Agent Server. In addition, each agent can interact with the physical part exploiting functionalities exposed by the IoT Devices Abstraction Layer (i.e. using the Object Interface) which, in turn, is in charge of notifying the events that occur in the physical part to the agent.

A *Message* is the atomic element of communication between agents. It carries an application specific content together with information about the sender agent and the receiver one.

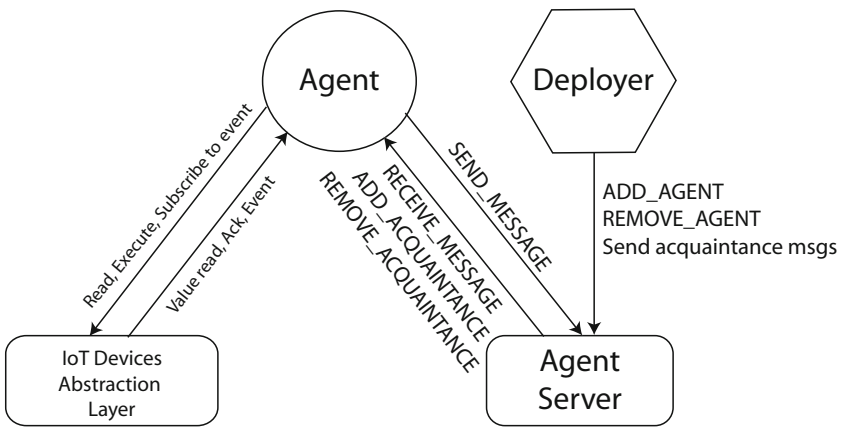

Fig. 2 CASE multi-agent entities

The *Agent Server* is the container for the execution of agents. It offers functionalities concerning the life cycle of the agents as well as functionalities for supporting agents' communication. Agent servers are arranged in a peer-to-peer fashion where each agent server hosts a certain number of agents and permits them to execute and interact among themselves in a transparent way. In other words, when an agent requests the execution of a functionality, its host agent server is in charge of redirecting transparently the request to the suitable agent server. In the following are listed the main functionalities each agent server exposes:

`SEND_MSG`. Through this functionality, the communication between agents is performed. The Agent Server is responsible for correctly delivering messages from the sender agent to the receiver one. If the sender and the receiver do not belong to the same agent server, the message is forwarded to the suitable "peer" agent server which is, in turn, engaged to deliver the message (see Fig. 3a).

`ADD_AGENT`. It instantiates an agent onto an agent server. CASE Multi Agent system is designed to permit agents to be dynamically loaded onto the agent server they have to belong to. As in SEND_MSG operation, agent servers are in charge for exchanging information among themselves in order to guarantee the ADD_AGENT request to be delivered to the correct agent server. This mechanism is shown in Fig. 3b. The latter figure also shows how the code is dynamically loaded exploiting *class repository server*. More in detail, when an ADD_AGENT request reaches the suitable agent server, if the agent code is not already available, the agent server automatically downloads it from a class repository.

`REMOVE_AGENT`. It removes an instance of an agent hosted by an agent server. This operation also exploits the "forwarding" mechanism described above.

Our architecture provides for specific kinds of message, that are the *acquaintance message*s. Those messages are used for establishing an acquaintance relationship among agents. The acquaintance message carries information about the location of a given agent (i.e. the location of the hosting agent server). The agent who receives the acquaintance message will use this information when it needs to send messages toward that destination. This kind of mechanism ensures agent behaviour to be completely independent w.r.t. the locations of agents it has to collaborate with.

For instance, let's consider that an agent is a computing node interconnected with others by means of a ring network. Each agent, therefore, can only interact with its previous agent nodes and its next one. Whenever further nodes must be connected to the ring network, only the acquaintance relationships have to be updated. In other words, a third entity can establish dynamically those acquaintance relationships without resorting to modify, re-build or restart any agent.

In CASE architecture the entity which is in charge of sending acquaintance messages in order to establish the acquaintance network is called *Deployer*. Deployer could be an external process as well as an agent, it can run during the configuration phase as well as during application execution. The Deployer concept will be described in details in "Dynamic deployment and roles".

As well as for the Agent Server and the IoT Devices Abstraction Layer, also agents expose a well-established interface. More specifically, the functionalities an agent exposes are called by the Agent Server for delivering messages and by the IoT Devices Abstraction Layer for notifying events that occur in the physical part. In the following are listed the main functionalities of an agent:

`RECEIVE_MESSAGE`. It is called when there is a Message to be delivered to the agent.

`HANDLE_EVENT`. It is called by the IoT Devices Abstraction Layer to notify that an event is occurred in the physical part.

`ADD_ACQUAINTANCE`. It is called when there is an acquaintance message to be delivered to the agent. The

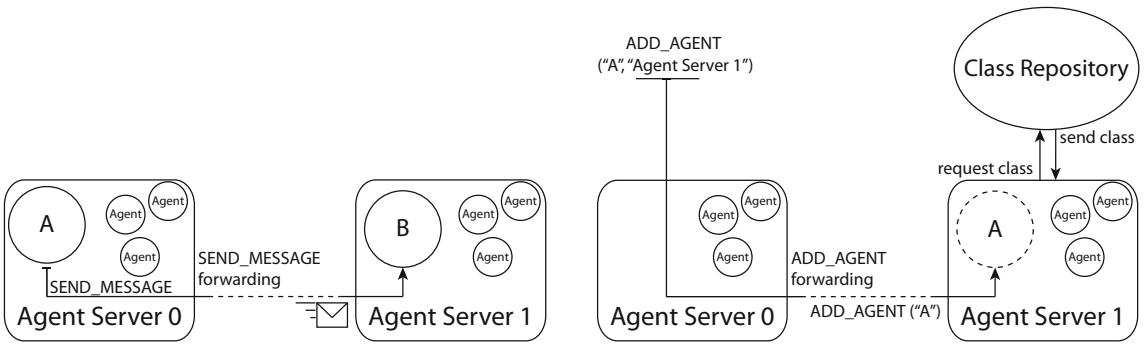

Fig. 3 Forwarding mechanism

implementation of this functionality concerns the store of the acquaintance relationship between the agent itself and the agent identified inside the message.

`REMOVE_ACQUAINTANCE`. It is called for removing a previously stored acquaintance relationship.

The specific behaviour of an Agent is realized through the implementation of `RECEIVE_MESSAGE` and `HANDLE_EVENT` functionalities.

Dynamic deployment and roles

The deployment of the agents as well as the configuration of the acquaintance relationships and the start-up of the application are actions performed by the so-called *Deployer*. An external process or even an agent can act as a Deployer. The deployment phase is typically executed just before the application can start properly; however, it is possible to act as Deployer even during application execution in order to update the configuration dynamically for hosting new features or adapting to foreseen and unforeseen changes in the environment. Deployer can be implemented centrally or in a distributed way. Basically, who acts as a Deployer operates using the `ADD_AGENT` functionality for deploying a new instance of an agent into an agent server, `REMOVE_AGENT` for removing a running agent from an agent server. Furthermore, Deployer is responsible for sending acquaintance messages that eventually end with calls to `ADD_ACQUAINTANCE` or `REMOVE_ACQUAINTANCE` on the specific agents. Finally, Deployer is also in charge of sending suitable "start" messages using `SEND_MSG` in order to start the application properly.

The acquaintance relationship is formally defined by a triplet: [A, B, R] where A and B are the agents involved in the relationship and R is a *Role* label. The triplet above means that agent A knows agent B and that B has the role R as acquaintance of A. During the execution, an agent exploits the Roles of its acquaintances to discriminate about how to interact with them. The dynamic deployment and the roles features are useful to modify the behaviour of an existing application by adding new functionalities at runtime. For example, in the case study supplied in "Case study: Designing a smart home environment", when a new activity is discovered a new *classifier agent* is created and deployed at runtime as will be better specified in the following.

CASE cloud

In the CASE cloud part a set of CASE nodes are deployed and run on a cloud infrastructure (see Fig. 4). Such nodes lack of the IoT Devices Abstraction Layer since, obviously, there are no physical entities connected. For this reason each node consists of the agent server only. The communication between the nodes connected with the physical part and the nodes in the cloud occurs by means of message exchange (see "CASE cloud").

Agents located on the cloud nodes act as intermediary between the CASE MAS and cloud analytics services. The feature of adding new agents at runtime can be used to link new services in the cloud during the execution of the system.

The CASE cloud part is "platform as a service", namely it provides a software stack and a set of libraries for the application execution. Anyway, it also can be seen as "software as a service" because the CASE user, i.e. the application developer, can inject his application by remotely adding the needed agents.

CASE activity recognition

Even though the CASE architecture can be effectively used in many smart home application domains, this work is mainly focused on the activity recognition task because it can be suitably exploited to analyse human behaviour so to realize and prevent dangerous situations which may occur in home environments. In this section the activity recognition module of the CASE architecture is presented.

In Fig. 5 the four main tasks for the in home activity recognition are shown: *data acquisition*, *feature extraction*, *activity discovery*, *activity recognition*. The data acquisition task consists in collecting the sensor data from the home environment. The feature extraction task is focused on filtering the acquired data in order to extract the relevant information which are then rearranged and passed to the activity discovery and activity recognition tasks. The activity discovery task uses the features to discover new previously unknown activities in order to enlarge the knowledge thus dynamically producing new *classifiers*. Such classifiers are eventually used by the activity recognition task which is in charge of processing the features and recognizing the high level activities in real time.

Each task is carried out in a different layer of the CASE architecture (Fig. 5). In particular, the data acquisition task is assigned to the IoT Devices Abstraction Layer, the feature extraction is carried out by *extractor agents*. The activity discovery task is assigned to the CASE cloud part. Indeed, it typically involves complex data mining algorithms so it can suitably exploit the large computational resources of the cloud technologies. The activity discovery task produces as output new *classifier agents* which are dynamically deployed on the agent servers (see "Dynamic deployment and roles"). These agents run in the computational nodes physically placed in the home environment so they can classify the activities fast enough to fulfil the real time requirements.

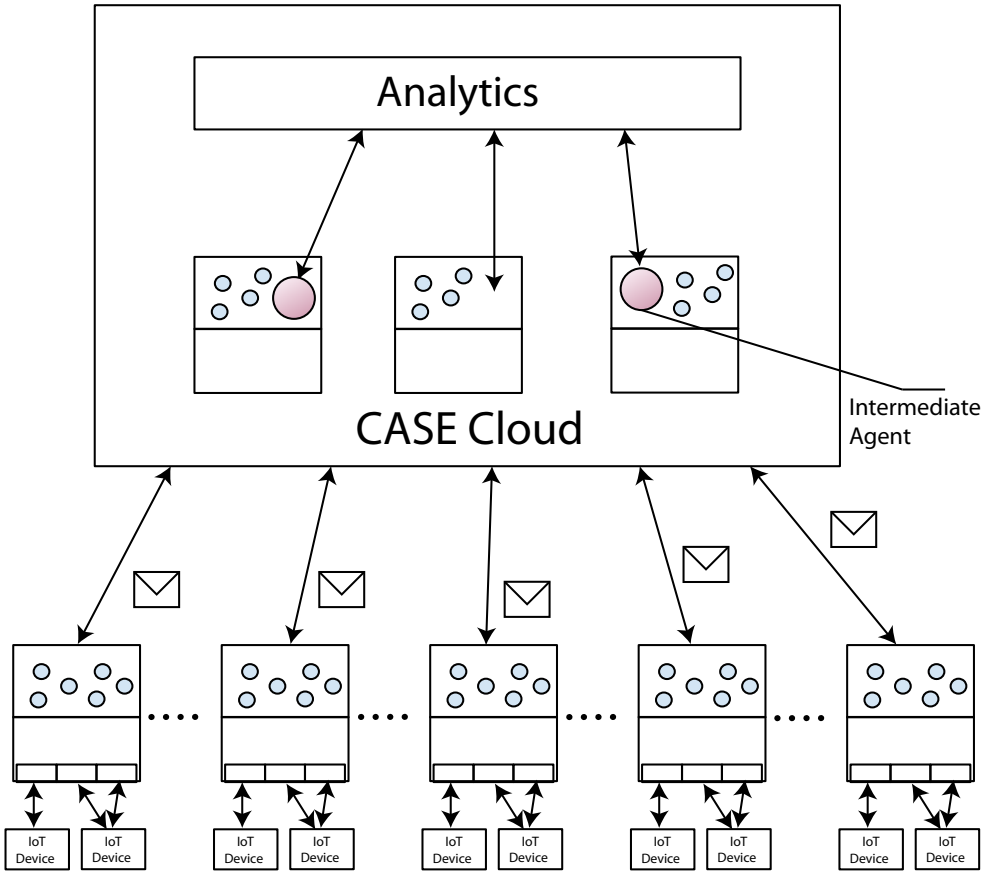

Fig. 4 The CASE cloud part

Case study: Designing a smart home environment

In order to highlight the effectiveness of the proposed platform, a case study has been designed inspired by the CASAS project [25]. Such case study concerns a single person who lives in a single apartment, which is instrumented with various proximity sensors and switch contacts. Each sensor provides boolean information (on/off) on the presence of someone in its sensed area. The apartment comprises five rooms: a kitchen, a dining room, a living room, a bedroom and a bathroom. In addition, the person who lives in the apartment always wears wearable sensors consisting of three axis accelerometers. Each room hosts a fixed CASE node where the environmental sensors are connected to. A mobile CASE node, e.g. a smartphone always carried by the person, can receive all the data of the wearable sensors.

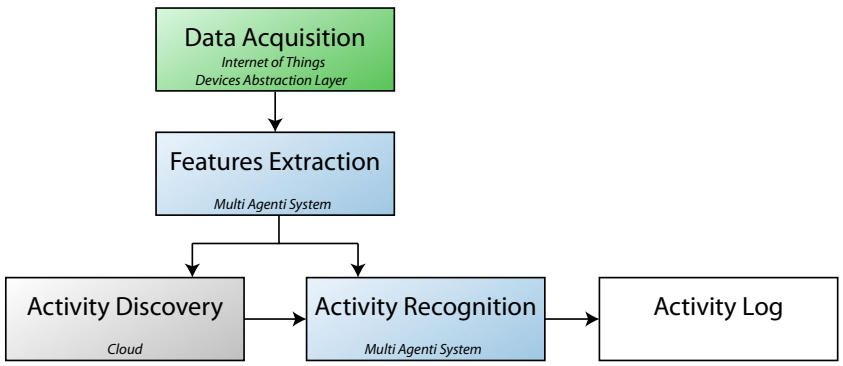

Fig. 5 Activity recognition tasks flow. Each task is executed in a different layer of the CASE architecture, as labeled

The aim of the case study is to show the design of the above described smart home environment which is integrated with a smart body. The goal of the designed system is to make discovery and recognition of in-house activities as well as the detection of situation of interest devoted to increase safety of the house inhabitants.

This section firstly furnishes an overview about the most used technologies enabling environment/body monitoring from both a sensor and a communication point of view. Such overview, even if not exhaustive, wants to provide readers with general guidelines on the available opportunities. Then, a detailed description of the recognizable activities is provided. Finally, the section details the design of the system by means of UML class diagrams.

Technologies for environmental monitoring

Several sensors are commonly used in Smart Home environments to discover/ classify different types of activities. In Table 1 the most common sensors used in Smart Home applications are shown. Sensors are classified with respect to which information type they can provide: people identification; comfort measurements; power consumption estimation; usage of devices, furnitures or appliances; and the detection of dangerous events.

Several platforms for sensor and actuator networks have been introduced so far. All of them can be applied in an IoT environment, since they can offer a distributed sensing, actuation, and low-level elaboration. The CASE architecture can exploit all the features offered by such networks for the environmental monitoring and control tasks since IoT devices abstraction layer exposes the same API regardless the low level communication protocols used [30, 31]. In literature several protocols have been introduced at different levels. In particular, Table 2 highlights the most important features of the commonly used communication standards for interconnecting smart devices in (home) smart environments. For every listed protocol, the pros and cons have been presented together with the transmissive mean used (wireless or wired).

Technologies for body monitoring

In order to create wearable body sensor networks to monitor human activities, several kinds of sensors and communication technologies have been used so far.

Such wearable body sensors are often implemented using wireless nodes to create Wireless Body Sensor Networks (WBSN) in order to simplify their deployment and because the wireless nodes are comfortable, lightweight, and unobtrusive. WBSN have been already used in literature for physical activity recognition, (remote) elderly assistance, rehabilitation, cardiac/respiratory diseases prevention, sleep quality monitoring and so on [54].

Table 3 shows the principal sensors used in the WBSN. In particular, such sensors are categorized according to the type of detection they allow. The most important detection types highlighted comprehend movement/direction, body posture, vital parameters, and comfort/wellness.

Table 2 gives a general overview of the most common communication technologies used, highlighting their pros and cons with reference to the WBSN environment [55, 56].

A schema for activity recognition

This case study is focused on the recognition of a set of activities through a chosen group of sensors. In particular, Table 4 shows all the activities that can be recognized. For each activity, the sensors used and the room involved are also shown.

To recognize daily activities from environmental sensors we follow the general schema of activity recognition presented in "CASE activity recognition" and the whole approach and algorithms introduced in [23]. Such approach

Table 1 Usage of sensors in Smart Home Environments

Detection Type	Sensors
Presence / Activity [32]	Passive Infrared, Active Infrared, Ultrasonic, Pressure, Smart Tiles, Camera, Microphone, Microwave Occupancy Detector
Identification [32]	Camera, RFID, Microphone, Active Infrared
Comfort / Environmental Measures [33]	Luminosity, Temperature, Humidity, Microphone, Gases Detector, Air Flow
Power / Electricity [34]	Smart Plug, Electricity Monitor Clip
Usage (things or devices) [34]	Magnetic Switch, Smart Plug, Pressure, Accelerometer
Emergency	Smoke, Liquid Presence, Glass Break, Temperature, Gases Detector

Table 2 Comparison of communication protocols for Smart Environments (SE) and Wireless Body Sensor Networks (WBSN). "+"/"-"/"x" stand for a pro/cons/irrelevant feature in the corresponding context

Protocol	Wired/Wireless	Features	SE	WBSN
X-10 [35]	Both	Power line wiring or radio frequency	+	Not used
		Widely used	+	
		Low bandwidth	–	
		Unmanaged signal collision	–	
		Lack of encryption	–	
UPB [36]	Wired	Uses power line	+	Not used
		Lack of encryption	–	
		Not many compatible devices	–	
INSTEON [37]	Both	Power line wiring or radio frequency	+	Not used
		X-10 compatibility	+	
		Complete systems are expensive	–	
		Closed eco-system	–	
IEEE 802.15.4 [38, 39]	Wireless	Low power	+	+
		Level 2 Protocol	–	–
		Requires high programming efforts	–	–
6LowPan [40]	Wireless	IPv6 compliant	+	+
		Mesh topology	+	x
		Not yet widely used	–	–
		Uses IEEE 802.15.4	x	x
Z-wave [41, 42]	Wireless	Many compatible devices	+	Not used
		Mesh topology	+	
		Low power	+	
		Uses a proprietary radio system	–	
ZigBee [39, 43–45]	Wireless	Low-cost	+	+
		Low-power	+	+
		Mesh topology	+	x
		Connects a large number of devices	+	x
		Low interoperability among different manufacturers	–	–
		Uses IEEE 802.15.4	x	x
802.11 (WIFI) [46, 47]	Wireless	Available in every home	+	Not used
		High data rate	+	
		High energy consumption	–	
Bluetooth Low Energy (BLE) [48]	Wireless	Low-power	+	+
		Available in smartphones and PCs	+	+
		High transfer rate	+	+
		Widely used	+	+
		Single-hop	–	x
		Limited range	–	x
		Suffers from interferences	–	–
Bluetooth [49]	Wireless	Available on smartphones and PCs	+	+
		High transfer rate	+	+
		Widely used	+	+
		Single-hop	–	x
		High power consumption	–	–
		Limited range	–	x
		Suffers from interferences	–	–

Table 2 (continued)

Protocol	Wired/Wireless	Features	SE	WBSN
UWB IEEE 802.15.6 [50]	Wireless	High transfer rate	Not used	+
		Low interferences		+
		High power consumption		–
		Not widely used		–
ANT [51]	Wireless	Light-weight protocol stack	Not used	+
		Low-power		+
		Widely used		+
		Proprietary		–
RuBee IEEE 1902.1 [52]	Wireless	Low-power	Not used	+
		Low frequency band		+
		Low transfer rate		–
		Small packet size		–
Sensium [53]	Wireless	Low-power	Not used	+
		Proprietary		–
		Not widely used		–
Zarlink	Wireless	Low-power	Not used	+
		Available for implantable devices		+
		Proprietary		–
		Not widely used		–
		Expensive		–

consists of (i) gathering sensor data, (ii) generating occupancy episodes, (iii) detecting frequent occupancy episodes, and (iv) clustering the instances of each frequent itemset so as to produce clusters representing particular activities. Those instances are used to recognize activities in realtime.

The data generated from ambient sensors produce events represented by a quadruple {*date, timestamp, sensorId, status*} that represents a status change (*on/off*) of the sensor identified by *sensorId*, at the time {*date, timestamp*}.

The raw data are then processed by collecting sensor firings segmented per room. The result of this operation is a list of occupancy episodes for each room, in the form of {*startTime, duration, usedSensors*}, where *startTime* is the beginning time of the occupancy episode, *duration* is how long the episode lasts, *usedSensors* is the set of the sensors activated during a particular occupancy episode. These episodes are the features that feed the activity discovery and recognition tasks.

The activity discovery task uses the frequent itemset mining Apriori algorithm [59, 60] for identifying the relevant occupancy episodes. In this case, the itemsets analyzed are the *usedSensors* of each occupancy episodes. A set of *usedSensors* is considered to be frequent if it occurs more than a fixed threshold. When a set of *usedSensors* is frequent, the related occupancy episode is considered relevant.

The relevant occupancy episodes are then clustered using the DBSCAN algorithm [61], determined by using the euclidean distance as similarity function evaluated on both the temporal attributes (i.e. *StartTime* and *Duration*). The output of this task is a set of clusters, that can be updated over time. Each cluster is represented by a tuple {*UsedSensors, MeanStartTime, MeanDuration*} which needs to be manually labelled.

Table 3 Usage of sensors in Wearable Sensor Networks

Detection type	Sensors
Movement / Direction [54]	Accelerometer, Gyroscope, Magnetometer
Body Posture [54]	Accelerometer, Gyroscope
Vital Parameters [57]	Glucometer, Blood pressure, Electrocardiography (ECG), Electroencephalography (EEG), Electromyography (EMG), Electrooculography (EOG), Heartbeat, Pulse Oxymeter, Temperature
Comfort / wellness [58]	Temperature, Galvanic Skin Response (GSR), CO2 Gas

Table 4 The activities to be recognized in the case study

Activity	Involved room	Involved sensors
SleepingInBed	Bedroom	IRMotion, SwitchContact
Bathing	Bathroom	IRMotion, SwitchContact
PersonalHygiene	Bathroom	IRMotion, SwitchContact
TakingMedicine	Bathroom	IRMotion, SwitchContact
MealPreparation	Kitchen	IRMotion, SwitchContact
Eating	LivingRoom	IRMotion, SwitchContact
LeaveEnterHome	LivingRoom	IRMotion, SwitchContact
Relaxing	LivingRoom	IRMotion, SwitchContact
StorageRoomUsed	StorageRoom	IRMotion
BedToToilet	Bedroom, Bathroom	IRMotion, SwitchContact
BodyPostureRecognition	All the rooms	3-axes Accelerometers

The set of labelled clusters constitutes the model for the activity recognition task, where each new occupancy episode is assigned to the most similar cluster (similarity between an episode and a cluster tuple is a distance function of *UsedSensors*, *StartTime*, and *Duration*). The recognized activities will feed an activity log that will be explained in detail below (see Fig. 6).

Regarding the wearable sensors of the case study, they are designed to periodically send to their Mobile CASE Node a set of features directly calculated on the sensor nodes which are equipped with three axes accelerometers to track body movements and postures, as already made in SPINE [57].

For the activity discovery/training phase, a set of experiments have been conducted in order to gather a set of labelled tuples with all the features that the sensor nodes are able to calculate. Then the most significant features used for activity recognition are selected using the SFFS (*sequential forward floating selection*) algorithm [62] applied on the acquired dataset. The chosen features act as input to the activity recognition system, designed in a CASE Agent, which will run a classifier recognizing the body postures and movements defined (i.e. *standing still, sitting, laying, walking, falling down*). For the recognition task, the K-Nearest Neighbor (KNN) classifier [63] is exploited, which uses the dataset with the most significant features as its "final" training set. In the KNN a new unlabeled input tuple is classified by assigning the most frequent label among the K training samples nearest to the input tuple itself.

The output of the KNN will be calculated every second so as the CASE Agent can process real-time information. Instead, the activity log introduced above will be fed everytime an update in the body posture is ready.

Figure 6 shows the main logical flow involved in this case study. At the bottom, the two kinds of activity recognition processes (explained above) are continuously executed: the first uses the environmental sensors' data (e.g. the IRMotion sensors) to recognize daily activities, while the second uses the wearable sensors to recognize the body postures of the apartment inhabitant. These two tasks will feed a single log, composed by triplets ⟨ DATE, TIMESTAMP, ACTIVITY ⟩, as shown in Fig. 6.

The activity log represents the knowledge base of an alert system which is able to produce notifications on the basis of a set of high level inference rules.

Such system can be implemented through an Alert Agent in charge of the safety of the house inhabitant (for further details see "A UML model of the case study"). As an example, in Table 5, some simple rules managed by an Alert Agent are listed.

Whenever the Alert Agent will detect that one of these rules is satisfied, it will generate an alert to notify an anomaly event. This feature of the system is particularly important in a Smart Home environment where elder

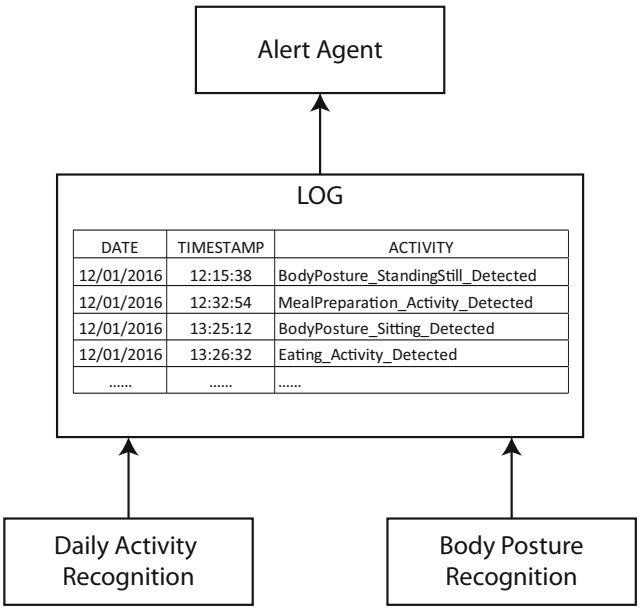

Fig. 6 Case study main flow

Table 5 Inference rules hosted in the Alert Agent

Condition one	Condition two	Alert
MealPreparation	BodyPosture_Lying	faint
PersonalHygiene	BodyPosture_Lying	faint
Night (Time)	Too long (BodyPosture_StandingStill)	illness
Bathing	Too long (BodyPosture_Sitting)	illness
AnyHomeActivity	BodyPosture_FallingDown	accidental fall or faint

assistance is performed. A wider set of alerts can be defined based on specific habits of people living in a smart home.

In this scenario, the Alert Agent combines results coming from both Daily Activity Recognition and Body Posture Recognition tasks in order to recognize high-level activities such as faint, illness, and so on. Anyway, the CASE platform allows also the combined use of data coming directly from both environmental and body sensors to feed

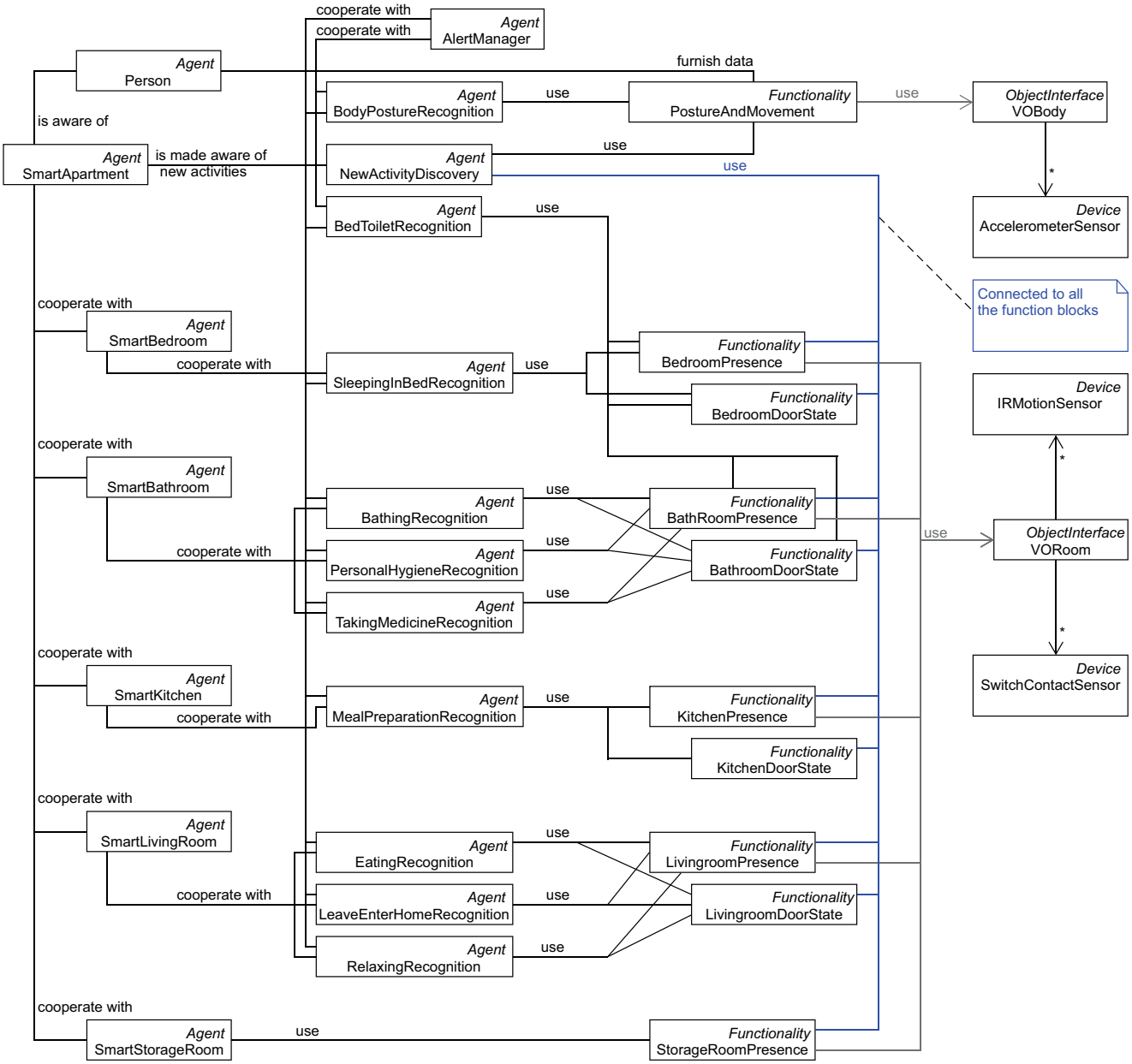

Fig. 7 A UML class diagram of the designed system

Table 6 Accuracy of activity detection

	Accuracy			
Activity	Proposed approach	COM [8]	AALO [23]	SPINE [54]
SleepingInBed	98 %	99.3 %	99.7 %	x
Bathing	72 %	72.9 %	77.5 %	x
PersonalHygiene	88 %	91.6 %	x	x
MealPreparation	94 %	97.8 %	64.5 %	x
LeaveEnterHome	86 %	88.3 %	89.9 %	x
Relaxing	94 %	95.9 %	x	x
StorageRoomUsed	98 %	x	x	x
BodyPosture_Sitting	96 %	x	x	96 %
BodyPosture_Standing	90 %	x	x	92 %
BodyPosture_Lying	98 %	x	x	98 %
BodyPosture_Walking	92 %	x	x	94 %
BodyPosture_Falling	96 %	x	x	100 %

a unique activity recognition task flow (see "CASE activity recognition").

A UML model of the case study

A conceptual model of the designed system is reported in the class diagram of Fig. 7. The whole model relies on four different kind of entities which are respectively *Agent*, *Functionality*, *ObjectInterface* and *Device*. They are *first-class entities* [64], i.e. entities directly made available by the exploited agent based platform. *Agent* is the basic abstract class from which all the applicative agents must be an heir. *Functionality* and *ObjectInterface* are instead the basic abstract classes respectively used for implementing (i) the functionalities offered by a virtual object and (ii) the virtual object itself. *Device* is instead a marker interface used to identify physical devices within the model.

The *SmartApartment* agent models the whole smart environment which is made by its component smart rooms each of them modelled by some other specific agents. In Fig. 7, such agents are those with the smart apartment agent directly cooperates. The *Person* agent, instead, models the person living in the apartment and which moves within it. The *AlertManager* is the agent having the responsibility of (i) monitoring the on-going activity carried out in the apartment, (ii) combining such activity information with the current state of the person in the apartment and (iii) to issue an alert if it is necessary. An alert situation is determined by using the rules reported in "A schema for activity recognition". The state of the person is provided by the *BodyPostureRecognition* agent which is able to detect the posture and movement of the person by using the *PostureAndMovement* functionality.

The activities carried out in the apartment are recognized through the agents whose class name ends with the word 'Recognition'. In the class diagram, each specific activity to recognize is related to the proper functionalities needed to accomplish it. These relationships between the agents and functionalities are highlighted in the diagram by using associations having the name 'use'.

All the functionalities ending with the word 'Presence' are devoted to detect the presence of a person within a specific environment. For instance, the functionality *Kitchen-Presence* is used to determine if the kitchen has an occupant. All the functionalities ending with the word 'DoorState' are instead used to detect if the doors existing in a given environment (entrance doors and doors of furniture) are closed or opened.

The *NewActivityDiscovery* agent is the only agent which resides in the cloud. It has the task of discovering new activities carried out in the apartment.

For each functionality, the UML diagram highlights the virtual objects used along with the exploited physical devices. Other details of the class diagram should be self explanatory.

Preliminary result analysis

After the description of the model for the proposed case study, this section is devoted to show some preliminary results obtained with a set of in-laboratory experiments.

Such experiments have been conducted by simulating in lab the set of activities listed in Table 6. Each room has been separately set up in laboratory, where sensors have been deployed as would be done in a real world setting. Environmental sensors are deployed using a set of waspmote[1]

[1] Waspmote website: http://www.libelium.com/products/waspmote/.

sensor nodes and a Raspberry Pi 2 model B[2] is used as a CASE computing node. Sensors are connected to the computing node through zigbee radios. Regarding the wearable sensors, the monitored person has been equipped with two Shimmer nodes[3] with accelerometers respectively placed on the thigh and the belt. These nodes are connected via Bluetooth to a smartphone hosting the mobile CASE node. A set of 100 simulations has been conducted for each activity listed in Table 6. The first half has been used for the training of the system while the second one for the activity recognition validation.

The obtained accuracy, i.e. the percentage of the activities correctly recognized, of both the Smart Home and the Body Monitoring activity recognitions is reported in Table 6. The table also portrays the accuracy obtained by the COM [8], the AALO [23] and the SPINE [54] approaches in recognizing similar activities. The obtained accuracy of our approach is good and very close to that obtained by the other approaches.

Conclusion

The paper has proposed a framework featuring activity recognition in Smart Home environment. The framework exploits the presented Cloud-assisted Agent-based Smart home Environment (CASE) platform. CASE is a novel platform for the distributed sensing and actuation in Smart Home environments which allows a simple integration between physical entities (such as sensors and actuators), a distributed multi-agent system, and a cloud infrastructure. In the CASE architecture, agents are disseminated both on local nodes and on the cloud to manage sensors/actuators or to execute complex algorithms. The framework also provides some methodological hints for the design of a Smart Home application enabling online recognition of daily living activities and body postures.

The designed case study shows a practical use of the framework, provides details on existing technological solutions exploitable in home environments, and furnishes a set of preliminary experimental results which are compliant with the state of the art.

Ongoing work focuses on the realization of a case study set up in a real home environments, where more activities, comprending several rooms together, can be analyzed.

Future work will be devoted both to an improvement of the CASE platform and to the realization of a wider use case that comprehends not only the activity recognition task but also home automation and energy optimization.

[2]Raspberry Pi website: https://www.raspberrypi.org/.

[3]Shimmer website: http://www.shimmersensing.com/.

Acknowledgments This work has been partially supported by "Smart platform for monitoring and management of in-home security and safety of people and structures" project that is part of the DOMUS District, funded by the Italian Government (PON03PE_00050_1).

References

1. Miorandi, D., Sicari, S., De Pellegrini, F., and Chlamtac, I., Internet of Things. *Ad Hoc Netw.* 10(7):1497–1516, 2012.
2. Fortino, G., Guerrieri, A., and Russo, W., Agent-oriented smart objects development. In: Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on, pp. 907–912, 2012.
3. Bierhoff, I., van Berlo, A., Abascal, J., Allen, B., Civit, A., Fellbaum, K., Kemppainen, E., Bitterman, N., Freitas, D., and Kristiansson, K. Smart home environment. COST Brussels, 2007.
4. Alkar, A.Z., and Buhur, U., An Internet Based Wireless Home Automation System for Multifunctional Devices. *IEEE Trans. on Consum. Electron.* 51(4):1169–1174, 2005.
5. Serra, J., Pubill, D., Antonopoulos, A., and Verikoukis, C. Smart HVAC Control in IoT: Energy Consumption Minimization with User Comfort Constraints The Scientific World Journal, 2014.
6. Fortino, G., Guerrieri, A., O'Hare, G., and Ruzzelli, A., A flexible building management framework based on wireless sensor and actuator networks. *J. Netw. Comput. Appl.* 35:1934–1952, 2012.
7. Guerrieri, A., Fortino, G., Ruzzelli, A., and O'hare, G. A WSN-based Building Management Framework to Support Energy-Saving Applications in Buildings. Hershey, PA USA: IGI Global, 2011.
8. Rashidi, P., and Cook, D.J., Com: A method for mining and monitoring human activity patterns in home-based health monitoring systems. *ACM Trans. Intell. Syst. Technol.* 4(4):64:1–64:20, 2013.
9. Pavón-Pulido, N., López-Riquelme, J.A., Ferruz-Melero, J., Vega-rodríguez, M. A., and Barrios-León, A. J., A service robot for monitoring elderly people in the context of ambient assisted living. *J. Ambient Intell. Smart Environ.* 6(6):595–621, 2014.
10. Dohr, A., Modre-Opsrian, R., Drobics, M., Hayn, D., and Schreier, G., The internet of things for ambient assisted living. In: Information Technology: New Generations (ITNG), 2010 Seventh International Conference on, pp. 804–809, 2010.
11. Richter, P., Toledano-Ayala, M., Soto-Zarazúa, G. M., and Rivas-Araiza, E.A., A Survey of Hybridisation Methods of GNSS and Wireless LAN Based Positioning System. *J. Ambient Intell. Smart Environ.* 6(6):723–738, 2014.
12. Sang-hyun, L., Lee, J.g., and Kyung-il, M., Smart Home Security System Using Multiple ANFIS. *Int. J. Smart Home* 7(3):121–132, 2013.
13. Fortino, G., Guerrieri, A., Lacopo, M., Lucia, M., and Russo, W., An agent-based middleware for cooperating smart objects. In: International Conference on Practical Applications of Agents and Multi-Agent Systems, pp. 387–398: Springer, 2013.
14. Jennings, N.R., On agent-based software engineering. *Artif. Intell.* 117(2):277–296, 2000.
15. Cicirelli, F., and Nigro, L.: Control centric framework for model continuity in time-dependent multi-agent systems. Concurrency and Computation: Practice and Experience. n/a–n/a cpe.3802, 2016.
16. Giordano, A., Spezzano, G., and Vinci, A., A smart platform for large-scale networked cyber-physical systems.In: *Management of cyber physical objects in the future internet of things methods, architectures and applications*: Springer, 2016.
17. Rashidi, P., and Mihailidis, A., A survey on Ambient-Assisted living tools for older adults. *IEEE J. Biomedical Health Informat.* 17(3):579–590, 2013.

18. Savidis, A., and Stephanidis, C., Distributed interface bits: dynamic dialogue composition from ambient computing resources. *Personal Ubiquitous Comput.* 9(3):142–168, 2005.
19. Bardram, J. E., Kjær, R. E., and Pedersen, M., Context-Aware User Authentication - Supporting Proximity-Based Login in Pervasive Computing. In: Dey, A., Schmidt, A., and McCarthy, J. (Eds.) UbiComp 2003: Ubiquitous Computing. Volume 2864 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 107–123, 2003.
20. Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H., and Nahrstedt, K., A Middleware Infrastructure for Active Spaces. *IEEE Pervasive Comput.* 1(4):74–83, 2002.
21. Cerqueira, R., Cassino, C., and Ierusalimschy, R., Dynamic Component Gluing Across Different Componentware Systems. In: Proceedings of the International Symposium on Distributed Objects and Applications. DOA '99, Washington, DC, USA, IEEE Computer Society, pp. 362–, 1999.
22. Evangelatos, O., Samarasinghe, K., and Rolim, J., Syndesi: A Framework for Creating Personalized Smart Environments Using Wireless Sensor Networks. In: Proceedings of the 2013 IEEE international conference on distributed computing in sensor systems. DCOSS '13, washington, DC, USA, pp. 325–330: IEEE Computer Society, 2013.
23. Hoque, E., and Stankovic, J.A., AALO: Activity recognition in smart homes using Active Learning in the presence of Overlapped activities. In: 6th International Conference on Pervasive Computing Technologies for Healthcare, PervasiveHealth 2012, pp. 139–146, 2012.
24. Aggarwal, C.C., Kong, X., Gu, Q., Han, J., and Yu, P.S., Active learning: A survey. In: Data Classification: Algorithms and Applications, pp. 571–606, 2014.
25. Cook, D. J., Krishnan, N. C., and Rashidi, P., Activity Discovery and Activity Recognition: A New Partnership. *IEEE Trans. on Systems, Man, and Cybernetics* 43(3):820–828, 2013.
26. Suryadevara, N., Mukhopadhyay, S., Wang, R., and Rayudu, R., Forecasting the behavior of an elderly using wireless sensors data in a smart home. *Eng. Appl. Artif. Intell.* 26(10):2641–2652, 2013.
27. Chen, L., Nugent, C., and Okeyo, G., An Ontology-Based hybrid approach to activity modeling for smart homes. *IEEE Trans. on Human-Machine Syst.* 44(1):92–105, 2014.
28. Steele, R., Lo, A., Secombe, C., and Wong, Y.K., Elderly persons perception and acceptance of using wireless sensor networks to assist healthcare. *Int. J. Med. Inform.* 78(12):788–801, 2009.
29. Giordano, A., Spezzano, G., and Vinci, A., Rainbow: An Intelligent Platform for Large-Scale Networked Cyber-Physical Systems. In: Proc. of the 5th International Workshop on Networks of Cooperating Objects for Smart Cities (UBICITEC), pp. 70–85, 2014.
30. Gomez, C., and Paradells, J., Wireless home automation networks: A survey of architectures and technologies. *IEEE Commun. Mag.* 48(6):92–101, 2010.
31. Starsinic, M., System architecture challenges in the home m2m network. In: Applications and Technology Conference (LISAT), 2010 Long Island Systems, pp. 1–7, 2010.
32. Ivanov, B., Ruser, H., and Kellner, M., Presence detection and person identification in smart homes. In: Int. Conf. Sensors and Systems, St. Petersburg, pp. 12–14, 2002.
33. Rawi, M.I.M., and Al-Anbuky, A., Wireless sensor networks and human comfort index. *Pers. Ubiquit. Comput.* 17(5):999–1011, 2013.
34. Basu, D., Moretti, G., Gupta, G.S., and Marsland, S., Wireless sensor network based smart home: Sensor selection, deployment and monitoring. In: Sensors applications symposium (SAS) 2013 IEEE, IEEE, pp. 49–54, 2013.
35. Chunduru, V., and Subramanian, N., Effects of power lines on performance of home control system. In: Power electronics, drives and energy systems, 2006. PEDES'06. International Conference on, IEEE, pp. 1–6, 2006.
36. Pulseworx: Upb technology description, 2016.
37. Darbee, P. Insteon: The details. Smarthome Technology, 2005.
38. Petrova, M., Riihijarvi, J., Mahonen, P., and Labella, S., Performance study of ieee 802.15.4 using measurements and simulations. In: Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE, Vol. 1, pp. 487–492, 2006.
39. Baronti, P., Pillai, P., Chook, V.W., Chessa, S., Gotta, A., and Hu, Y. F., Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. *Comput. Commun.* 30(7):1655–1695, 2007. Wired/Wireless Internet Communications.
40. Shelby, Z., and Bormann, C. 6loWPAN: The wireless embedded Internet. Volume 43 John Wiley & Sons, 2011.
41. Alliance, Z. W.: Z-wave: The new standard in wireless remote control, 2009.
42. Galeev, M. T., Catching the z-wave. *Embed. Syst. Des.* 19(10):28, 2006.
43. Zou, Z., Li, K. J., Li, R., and Wu, S., Smart home system based on ipv6 and zigbee technology. *Procedia Eng.* 15:1529–1533, 2011.
44. Gill, K., Yang, S. H., Yao, F., and Lu, X., A zigbee-based home automation system. *IEEE Trans. Consum. Electron.* 55(2):422–430, 2009.
45. Alliance, Z. Zigbee home automation public application profile. IEEE J. Select Areas Commun, 2007.
46. Hazmi, A., Rinne, J., and Valkama, M., Feasibility study of ieee 802.11 ah radio technology for iot and m2m use cases. In: Globecom workshops (GC wkshps) 2012 IEEE, IEEE, pp. 1687–1692, 2012.
47. Sun, W., Choi, M., and Choi, S., Ieee 802.11 ah: a long range 802.11 wlan at sub 1 ghz. *J. ICT Stand.* 1(1):83–108, 2013.
48. Mackensen, E., Lai, M., and Wendt, T.M., Bluetooth low energy (ble) based wireless sensors. In: Sensors, 2012 IEEE, IEEE, pp. 1–4, 2012.
49. Specification of the Bluetooth System v4.0. www.Bluetooth.org, 2016.
50. Kwak, K. S., Ullah, S., and Ullah, N., An overview of ieee 802.15.6 standard. In: Applied Sciences in Biomedical and Communication Technologies (ISABEL), 2010 3rd International Symposium on, IEEE, pp. 1–6, 2010.
51. [Online], D.I.I. Thisisant: the wireless sensor network solution, 2016.
52. 1902.1-2009 - IEEE Standard for Long Wavelength Wireless Network Protocol. http://standards.ieee.org/findstds/standard/1902.1-2009.html, 2016.
53. Wong, A., McDonagh, D., Omeni, O., Nunn, C., Hernandez-Silveira, M., and Burdett, A., Sensium: An ultra-low-power wireless body sensor network platform: Design and application challenges. In: Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE, pp. 6576–6579, 2009.
54. Bellifemine, F., Fortino, G., Giannantonio, R., Gravina, R., Guerrieri, A., and Sgroi, M., SPINE: a domain-specific framework for rapid prototyping of WBSN applications. *Softw. Pract. Experience* 41(03):237–265, 2011.
55. Chen, M., Gonzalez, S., Vasilakos, A., Cao, H., and Leung, V.C.M., Body area networks: a survey. *Mobile Netw. Appl.* 16(2):171–193, 2010.
56. Dementyev, A., Hodges, S., Taylor, S., and Smith, J., Power consumption analysis of bluetooth low energy, zigbee and ant sensor

nodes in a cyclic sleep scenario. In: Wireless symposium (IWS), 2013 IEEE International, IEEE, pp. 1–4, 2013.
57. Fortino, G., Giannantonio, R., Gravina, R., Kuryloski, P., and Jafari, R., Enabling Effective Programming and Flexible Management of Efficient Body Sensor Network Applications. *IEEE Trans. Human-Machine Syst.* 43(1):115–133, 2013.
58. Hao, Y., and Foster, R., Wireless body sensor networks for health-monitoring applications. *Physiol. Meas.* 29(11):R27, 2008.
59. Agrawal, R., and Srikant, R., Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th international conference on very large data bases. VLDB '94, san francisco, CA, USA, Morgan Kaufmann Publishers Inc, pp. 487–499, 1994.
60. Hipp, J., Güntzer, U., and Nakhaeizadeh, G., Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explor. Newsl.* 2(1):58–64, 2000.
61. Ester, M., Kriegel, H.P., Sander, J., and Xu, X., A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd, Vol. 96, pp. 226–231, 1996.
62. Pudil, P., Novovičová, J., and Kittler, J., Floating search methods in feature selection. *Pattern Recogn. Lett.* 15(11):1119–1125, 1994.
63. Cover, T., and Hart, P., Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.* 13(1):21–27, 1967.
64. Weyns, D., Omicini, A., and Odell, J., Environment as a first class abstraction in multiagent systems. *Auton. Agent. Multi-Agent Syst.* 14(1):5–30, 2007.