Leveraging Smart Environments for runtime resources management

Paolo Barsocchi, Antonello Calabró, Francesca Lonetti, Eda Marchetti and Filippo Palumbo

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche (CNR)
via G. Moruzzi, 1 - 56124 Pisa, Italy
{firstname.lastname}@isti.cnr.it

Abstract. Smart environments (SE) have gained widespread attention due to their flexible integration into everyday life. Pervasive sensing technologies, such as RFID and wireless sensors, are used to recognize and track the activities that people perform during the day, and to allow communication and cooperation of physical objects. Applications leveraging these smart environments rely on regular exchange of critical information and need accurate models for monitoring and controlling the SE behavior. Different rules are usually specified and centralized for correlating sensor data, as well as managing the resources and regulating the access to them, thus avoiding security flaws. In this paper, we propose a dynamic and flexible infrastructure able to perform runtime resources' management by decoupling the different levels of SE control rules. This allows to simplify their continuous updating and improvement, thus reducing the maintenance effort. The proposed solution integrates low cost wireless technologies and can be easily extended to include other possible existing equipments. A first validation of the proposed infrastructure on a case study is also presented.

Key words: Smart environment, Monitoring, Sensors, Access Control Policy

1 Introduction

The vision of Mark Weiser on ubiquitous computing [1] is now closer to reality than it was in 1991, anticipating a technological revolution in which the computation has a dominant role in our everyday world. The new reality of disappearing technologies into the environment has been enabled by the miniaturization of mobile devices, new possibilities offered by wireless communications, new localization techniques, falling costs, sizes, and power requirements. The interaction between people and ubiquitous computing is identified as a Smart Environment (SE), in which there is a seamless integration of people, computation, and physical reality. SE is any area of interest where there are means to detect the occupants' context, so that contextual information can be used to support and enhance their abilities in executing application-specific actions by

providing information and services tailored to their user immediate needs [2]. The SE paradigm depends on communication and cooperation among numerous devices, sensor networks embedded in the environment itself, servers in a fixed infrastructure and the increasing number of mobile devices carried by people.

Thousands of smart devices are now operating in cities, gathering information and providing smart applications for e.g. environmental monitoring, energy management, traffic management, smart buildings and smart parking [3]. These devices integrate computation, networking, and physical processes and are able to monitor and control physical objects providing an extremely efficient and economic mean for improving the quality of life of citizens.

From the networking point of view, one of the solutions adopted by the SE is constituted by the Wireless Sensor Networks (WSNs) [], which are autonomous devices managing sensing, computing and wireless communication capabilities. The integration of computation, networking, and physical processes require regular exchange of critical information in timely and reliable fashion and a specific modeling and control of the SE behavior. Considering, in particular, the control point of view, a SE usually involves three levels of rules: i) the rules for managing and correlating sensors data and technologies; ii) the rules able to guide the SE process, to define the users and the systems behavior, and to protect against possible problems and inconveniences faults; iii) the access control rules that manage the resources (sensors, technologies, data, and so on) and protect against possible malicious use or security flaws.

However, independently of the formalism adopted, writing control policies is a hard, verbose and error-prone activity. Modifications and updating of the access control policy may cause inconsistencies and security flaws; therefore, an accurate, time and effort consuming validation and verification should be implemented [4, 5].

Existing solutions generally try to centralize the management and control of the different rule levels providing tools and architecture resources and workflow management. However, due to the complexity of SEs and the several on-the-fly necessary updating and improvements on the different rules, the proposed available infrastructures are not flexible enough to support such a continuous modification. Especially in large scale organizations, in order to partially solve this problem, the common practice is to define and implement just the basic rules policies that may remain unchanged for a considerable amount of time, usually extracted by the internal regulations and network specification requirements. As side effect of that modeling and management of SE behavior could become outdated over time, leading either inconsistencies with the evolved behavioral and technological organization environment, or security vulnerabilities.

From the previous considerations, the main goal of this paper derives: to provide a dynamic and flexible infrastructure that decouples the different levels of rules, so to maximize the effectiveness and reduce as much as possible the maintenance and the updating effort. In the proposed approach, the more dynamic rules for the continuous decision and control, typical of the standard usage control environment, have been completely separated from the more sta-

ble rules of the physical network behavior and the access control strategies, so to let a more dynamic and flexible management. To this purpose, the levels of rules considered are: the *Business Rules*, which define the sensors behavior and activities; the *Usage Control Rules*, which define the users and sensors interactions; and the *Access Control Rules*, which manage the accesses to the different resources expressed through a specific control policy formalism. To maximize the decoupling and flexibility, in the proposed infrastructure the management of each of the three rule levels is in charge of a different independent reasoner, as better detailed in the rest of this paper.

Moreover, in the developing and deploying of the proposed infrastructure, important requirements have been considered: i) the low cost of the proposed solutions; ii) the non-invasiveness of the installations; iii) the possibility of integration with other possible existing equipment. Among the different proposals, the trade-off solution adopted in this paper relies on ZigBee [6]. This is a standard-based wireless technology designed to address the unique needs of low-power WSNs, and to model the different resource capabilities.

The paper is organized as in the following. Section 2 shows a motivating example. Section 3 presents some basic concepts about usage and access control systems. Section 4 presents the proposed infrastructure for runtime management and control of smart environments. Section 5 provides a first validation of the proposed approach on a case study. Related work are presented in Section 6 whereas Section 7 concludes the paper also hinting at future work.

2 Motivating Example

To better explain the solution adopted, we here describe a very simple example of application considering the management of a greenhouse where different kinds of agricultural experimentations are developed. In the greenhouse, the business rules that manage and coordinate the sensors (temperature and humidity) and the actuators (vaporizer, dehumidifier, heating, air-conditioner) are: i) if humidity goes lower (upper) than a boundary range, a vaporizer (dehumidifier) is switched on; ii) if the temperature goes lower (upper) than boundary range, a heating (air-conditioner) is switched on.

These rules are fixed and may vary only in the boundary ranges required for the different experimentations, or in case new sensors and/or actuators are installed in the room. Therefore, due to the low variability, the rules can be directly embedded into the access control engine controlling the sensors' behavior and sporadically assessed for reducing the overall effort for their maintenance.

The access control rules are specified in agreement with the administrative regulations of the Laboratory hosting the greenhouse and establish that it will be open to all the agronomists only during the working time (8am-8pm). For security reasons, only guardians may access the greenhouse out of the working time. Due to the low flexibility of the access control rules, these can be specified once for all into a policy and directly managed by an access control engine.

However, the above mentioned regulation (business and access rules) do not completely satisfy the management exigences of the greenhouse. Indeed, due to the frequent turnover of the hosted agricultural experimentations, different usage rules have to be implemented, as for instance: i) access to the greenhouse must be inhibited to anyone in case of chemical treatment; ii) access to the greenhouse must be allowed only to the responsible of the experiment so to not compromise the final results; iii) access to the greenhouse must be allowed during the night to an agronomist for measuring specific oxygen parameters; iv) air conditioner must be inhibited in presence of specific plants.

Of course, these are just simple examples of usage rules, voluntarily ignoring complex, safe, and security aspects of the greenhouse management. Not all the mentioned rules have to be verified at the same time and for all the experimentations. The peculiarity of the proposed infrastructure is to leave the freedom to the responsible of the experimentation to define each time the usage rules more suitable for his/her purpose. This can be done either by specifying the proper usage rules or to select the most suitable one from a pre-defined collection of frequently adopted ones. A dedicated engine will manage the frequent updates/-modifications of the usage behavior, overriding when necessary the business and access rules, without an impact on the overall management of the greenhouse.

The usage management could have been implemented by writing a standard usage control policy enforced by a standard usage control engine. However, this should have had the same issues as writing the access control policies. The specification of all the varieties of the possible revocations and exceptions in several multiple situations should be an extremely complex activity causing errors and inconsistencies, if not properly assessed.

The solution of decoupling the usage control from the business and access one will provide a dynamic, flexible and adaptable infrastructure management suitable for any experimentation and activity.

3 Background

In the following Section, we provide some basic concepts about the formalism used for defining the access and usage control rules.

3.1 Access and Usage Control

Access control is one of the most adopted security mechanisms for the protection of resources and data against unauthorized, malicious or improper usage or modification. In the last decades, a large variety of policies and policy models have been proposed to define authorized operations on specific resources, such as (RBAC) Model [7] and (XACML) [8].

An XACML policy defines the access control requirements of a protected system. An access control request aims at accessing a protected resource in a given system whose access is regulated by a security policy. The request is

evaluated on the Policy Decision Point (PDP) against the policy and the access is granted or denied.

A simplified vision of an XACML policy has a hierarchical structure: at the top level there is the policy set, which can contain in turn one (or more) policy set(s) or policy elements. A policy set (a policy) consists of a target, a set of rules and a rule combining algorithm. The target specifies the subjects, resources, actions and environments on which a policy can be applied. If a request satisfies the target of the policy set (policy), then the set of rules of the policy set (policy) is checked, else the policy set (policy) is skipped. A rule is composed by: a target, which specifies the constraints of the request that are applicable to the rule; a condition, which is a boolean function evaluated when the request is applicable to the rule. If the condition is evaluated to true, the result of the rule evaluation is the rule effect (*Permit or Deny*), otherwise a *NotApplicable* result is given. If an error occurs during the application of a request to the policy, *Indeterminate* is returned. The rule combining algorithm specifies the approach to be adopted to compute the decision result of a policy when more than one rule may be applicable to a given request. An example of XACML policy is given in Listing 1. Usage control model (UCON) [9] is one of the emerging and comprehensive attribute based access control models that has the ability of monitoring the continuous updates in a system addressing the limitations related to attribute mutability and continuous usage permission validation. UCON model is an extension of the traditional access control models, which besides authorizations introduces new factors in the decision process, namely obligations, conditions, and mutable attributes. Mutable attributes are paired with subjects and objects, and their values are updated as a consequence of the decision process. Hence, the attributes that have been evaluated by the security policy to initially grant an access to a resource could change their values, while the access is in progress in such a way that the access right does not hold anymore. In this case, the access should be interrupted to preserve the system security. For this reason, UCON policies specify whether or not a decision factor must be evaluated before and/or during the usage of the resource (continuous policy enforcement).

4 Infrastructure

In this Section, we provide some details about the infrastructure used to decouple Business, Usage Control and Access Control rules. As shown in Figure 1, the Infrastructure is conceptually divided in different nodes (see Figure 1):

- the *Access Control Engine* is the node in charge of implementing the access control management
- the *Glimpse: Monitoring Infrastructure* is the node monitoring and enforcing the Business and Usage rules
- the *Sensors and Actuators* are physical (hardware) components of the infrastructure
- the *Management Interface* is the GUI through which the different rules can be defined and feedbacks and log analysis can be provided.

As in Figure 1, the *Administrators* are in charge of providing the definition of the three levels of rules for the overall infrastructure. This can be done by means of a *GUI* on which *Rules editor* and *Policies editor* components are running. Specifically, through *Rules Editor* the Administrators can define the Business and Usage rules using a specific language (further details are provided in Section 4.2). Additionally, by means of *Policy Editor* they can define the XACML access control policies that will rule the resources access. Finally, through the *GUI* the Administrators can visualize logging data, monitoring results, sensors and actuators status. In the following subsections, more details about the above mentioned nodes are provided.

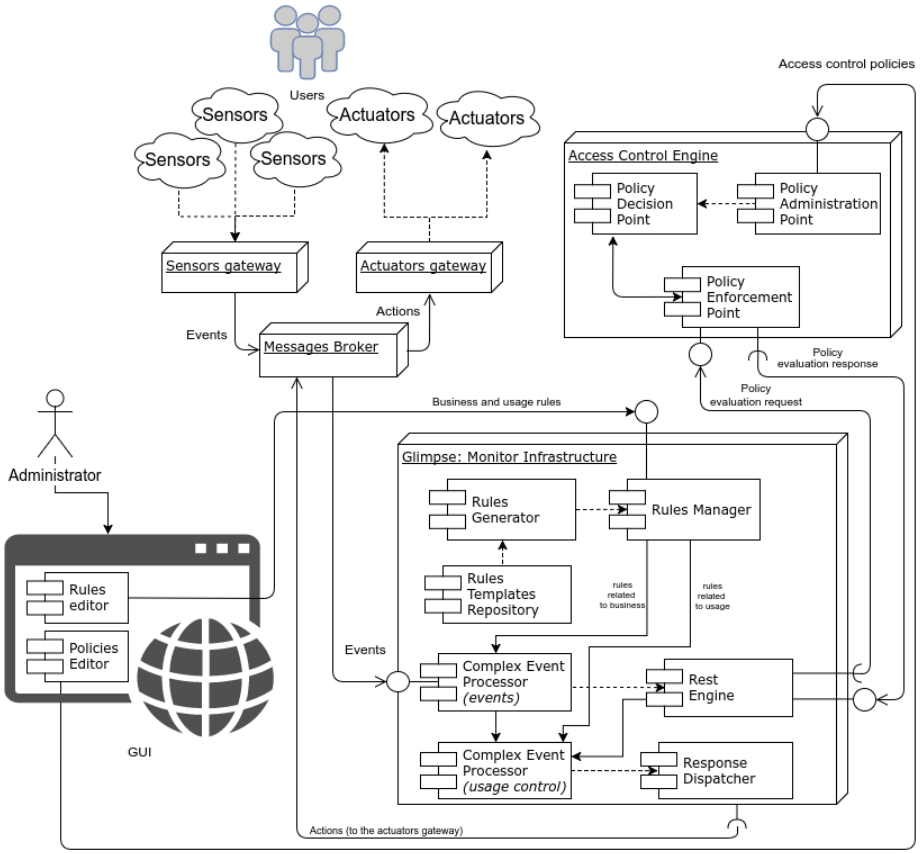

Fig. 1. Proposed architecture

4.1 Access Control Engine

This node manages the resource access by enforcing the XACML Policy defined by the Administrators. In particular, the *Access Control Engine* node contains three components (Figure 1 top right):

- The *Policy Enforcement Point (PEP)*, usually embedded into an application system; it receives the access request in its native format from the *Glimpse: Monitoring Infrastructure*, constructs an XACML request and sends it to the *Policy Decision Point (PDP)*; it receives the PDP responses and forwards them to the *Glimpse: Monitoring Infrastructure* through its *REST Engine Interface*;
- The *Policy Decision Point (PDP)* evaluates the policy with respect to the request and returns the response, including the authorization decision to the *PEP*;
- The *Policy Administration Point (PAP)* is the component entity in charge of managing the policies and deploying them on the PDP; it receives the XACML access control policy by the *Management Interface*.

4.2 Monitoring components

The monitor infrastructure is integrated into the proposed infrastructure; it is a flexible, adaptable and dynamic solution independent of any specific sensor or access control network notation or execution. The proposed solution allows to take counter measures for recovering from violations of defined performance constraints. These constraints are not mandatory specified at the system startup, but can be automatically raised from the rule engines involved or can be improved at runtime by injecting new rules on the complex event processors. The monitoring framework presented in this paper has been inspired by the monitoring architecture presented in [10, 11]. The *Glimpse: Monitoring Infrastructure* node (Figure 1) manages the complex event processing and the interactions with *Sensors*, *Actuators* and *Access Control Engine*, and includes new features devoted to the usage and access control request generation.

The main monitoring components are:

- The *Rules Manager* component is in charge of orchestrating the rules generation starting from the templates stored within the component *Rule templates Repository* through the *Rules Generator* component.
- The *Rules Generator* is the component in charge of synthesizing the rules starting from the directives received by the *Rules Manager* by means of techniques based on generative programming approaches [12, 13].
- The *Rules Templates Manager* is an additional internal repository storing the meta-rules enabling the run-time adaptation by means of generative procedures.
- The *CEP - Events* is a rule engine realized by means of the Drools rule language [14]. It correlates the events flowing from *Sensors* with the rules loaded by the *Rules Manager* component.

- The *CEP - Usage* is in charge of correlating complex events generated by the *CEP - Events* with the rules related to the usage of the resources, loaded by the *Rules Manager*.
- The *Rest Engine*, is the component in charge of communicating through REST [15] interfaces with the *Access Control Engine* in order to send/receive the *Access Control Engine* request/response.
- The *Response Dispatcher* through the *Message Broker (AMQ)* sends events to the actuators managed by the *Actuators gateway*.

The peculiarities of the proposed architecture is to include a chain of two CEP entities, the *CEP - Events* and the *CEP - Usage*, for decoupling the activities concerning the management of the sensors from those more related to the administration of the resource usage and alarming situations. This makes easier the definition of new primitive events generated by (new/updated) sensors and the inferring of events in the form of composite events in a way completely independent of the access and usage control rules. Moreover, it lets a quick and high level updating of the general resource access and usage regulations and the planning of specific corrective actions in case of alarms or resource violations, leveraging from the specific sensor network on which they are implemented.

From a practical point of view, all the communications among monitoring components are performed through [16] messages sent to the *Message Broker* aka *AMQ*, running on top of an Enterprise Service Bus like ServiceMix. In order to improve the communication security, each component exposes a SSL certificate (self-signed certificates).

4.3 Sensors and Actuators components

Sensors and *Actuators* (in top left side of Figure 1) are deployed over the network and communicate with the infrastructure through the *Sensors gateway* and the *Actuators gateway*, respectively. These hardware components send messages to the *Glimpse: Monitoring Infrastructure* through a *Message Broker (AMQ)* realized by means of ActiveMQ [17] using a predefined event type.

From a technical point of view, the sensor nodes considered in the proposed infrastructure are based on the module Core2530, produced by WaveShare Electronics[1]. The sensor nodes are connected with a ZigBee node [2], which is able to: evaluate the real power consumption of the target environment; identify all the indoor movements of users through PIR-based motion sensors; detect environmental noise; measure temperature and relative humidity. Every node of the distributed sensor network is configured as a ZigBee router to exploit multihop functionality and it speriodically sends the measured data to the ZigBee coordinator. The ZigBee network ensures a reliable communications in indoor environments without suffering due to the multipath effect [18]. Moreover, each

[1] http://www.wvshare.com/
[2] http://www.zigbee.org/

user is equipped with a Bluetooth Low Energy (BLE) beacon [3], which periodically sends a message useful for locating and identifying the user. Figure 2 shows the hardware used for sensing the environment. In particular, on the left there are the RadBeacon Dot [4] and the BLED112 [5] used as a sender and receiver beacon, while on the right side there is a ZigBee node.

Fig. 2. The hardware used to sensing the environment

The middleware, named *Sensor Weaver*, uses ZB4O[6] to interact with sensors and actuators deployed in the ZigBee networks [19] and integrates the BLE in order to abstract the different kinds of technologies.

The gateway node provides access to the sensors discovered through an IP network and a communication platform. The main goal of Sensor Weaver is to provide a secure communication platform for the exchange of sensing information in a distributed sensor network environment [20, 21]. Moreover, Sensor Weaver also provides tools and applications that enable long-term sensor monitoring and automatically control the actuators deployed in the WSN [22, 23].

In order to separate communication concerns of Sensor Weaver, we designed several communication buses. Each bus has a specific managing role (see Figure 3): (i) a *Service Bus* for the service life-cycle events; (ii) a *Context Bus* for the sensor measurement updates; (iii) a *Control Bus* for the invocations of actuators. We implement Sensor Weaver on top of the OSGi platform and we use the MQTT messaging service [24, 25].

5 Infrastructure Application

The infrastructure described in the previous Section provides the facilities to detect and to correlate events generated by different layers, supporting access and usage control features.

In this Section, we present the infrastructure application to a simplified version of the management of a cold storage inside the *ISTI-CNR* (Istituto Scienza e

[3] https://developer.mbed.org/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon/
[4] http://store.radiusnetworks.com/collections/all/products/radbeacon-dot
[5] https://www.bluegiga.com/
[6] http://zb4osgi.aaloa.org/

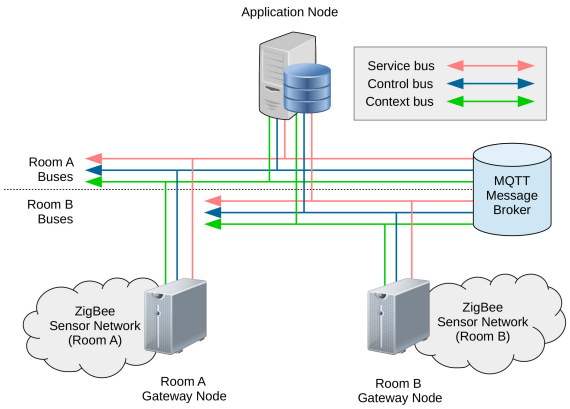

Fig. 3. The architecture used with Sensor Weaver.

Tecnologie dell'Informazione - Consiglio Nazionale delle Ricerche of Pisa Italy) [7] research area. Specifically, we refer to a medical cold storage called *Laboratory Cold Room (LCR)*, which has been instrumented to keep the demanding temperature required for safe and secure storage of a wide range of laboratory materials.

The administration of this kind of refrigerators includes: i) business rules for managing the security boundary value of each sensor or combination of them (for instance, the tolerance temperature, humidity ranges, and so on); ii) access control policies ruling who and when can access LCR (name or the role of people that are allowed to work in the LCR); iii) usage control rules for managing sensors failures, resource violations, and alarming situation in general (for instance, the technical personnel to be called in case of problems). It is out of the scope of this paper to go into details of the complex set of rules necessary for managing the LCR. Here, we voluntarily keep the scenario simplified to better explain the role and the advantages of the proposed infrastructure. The complete description of the implementation can be found in [26].

In the following Sections, more details about the case study set up and the runtime monitoring of LCR are provided.

5.1 Case Study Set Up

As shown in Figure 4, from a practical point of view, the LCR has been equipped with a beacon receiver for gathering data from BLE beacons and with several sensors (see Section 4.3 for more details), which are: Temperature and humidity; Presence (PIR-based); Energy consumption; Noise detector; RFID Bagde Reader for Room access Control. An instance of the *Monitoring Infrastructure* has been deployed on: *ubuntu@n037.smart-applications.area.pi.cnr.it*, a virtual machine running on top of ISTI-CNR cloud infrastructure, while the *Access Control Engine* was running on *avalon.isti.cnr.it*.

[7] http://www.isti.cnr.it

The probes that generate events related to the sensors and the actuators were running on top of the middleware: *energia.isti.cnr.it* and the events were flowing through the message broker *AMQ* running on *atlantis.isti.cnr.it*.

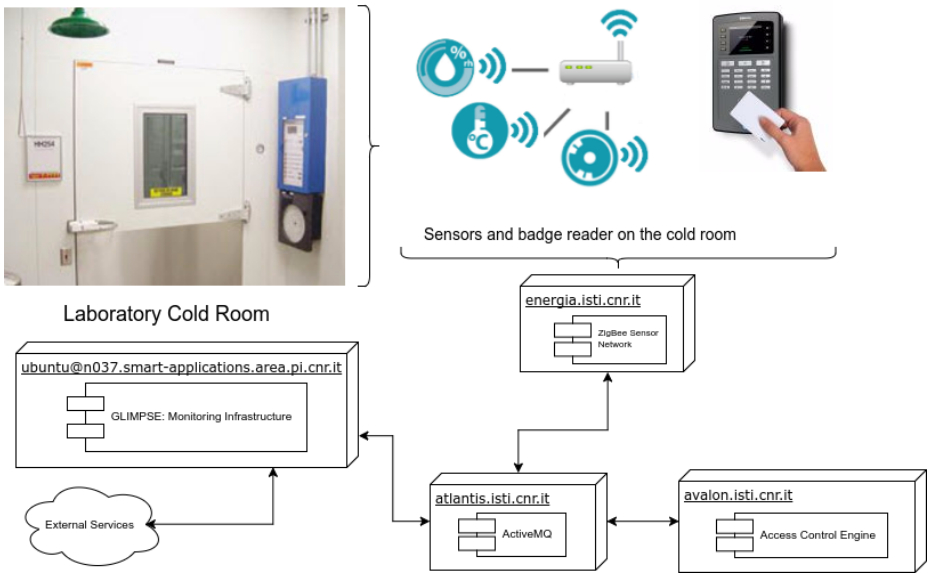

Fig. 4. Deployment Configuration

5.2 Business, Access and Usage rules Management

In this Section, we focus on the interaction between *CEP - Events*, *CEP - Usage* and the Access control Engine for the enforcement of the business, access and usage rules. Specifically, as shown in Figure 1, through the *Rules Editor* the Administrator loaded the business rule useful for monitoring the sensors status and the access control rule for the identification of who is currently asking resource access.

When an employee, through the RFID Bagde Reader, tries to access the LRC, the *CEP - Events* receives the access request and extracts the room ID and the badge ID. By querying the ISTI-CNR internal employees database, the *CEP - Events* retrieves (Role, Owner room id) attributes related to the user who is asking for the access. Using the data collected, the *CEP - Events* sends a *Policy evaluation request* through the *Rest Engine* to the *Access Control Engine* node and a *PdpAccessRequest* event to the *CEP - Usage* to notify that an access request has been sent. The *PEP* translates the request into an XACML access request and sends it to the *PDP*, which evaluates the request according to the access control policies injected by *PAP*, and sends back the response to the*PEP*, which in turn sends back to the *CEP - Usage* through the *Rest Engine*.

An example of a business rule used by the *CEP - Events* for controlling all the installed sensors is shown in listing: 2. In particular, when the *CEP - Events* receives from the monitored sensors, for two consecutive times, null or out-of-range values (lines 13-14 and 17-18 of Listing: 2), the *CEP - Events* generates a complex event called *SensorFailureEvent* for notifying the detected failure to the *CEP - Usage*, so that it can activate the corrective actions.

Going into details of the LCR access control policy, the Listing 1 presents a simplified version of the overall XACML access control policy considered.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
3   xmlns:xacmlcontext="urn:oasis:names:tc:xacml:2.0:context:schema:os"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os␣http://docs.oasis-open.org/xacml/
          access_control-xacml-2.0-policy-schema-os.xsd"
6   PolicyId="SmartPolicy"
7   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable">
8    <Target>
9     <Subjects>
10     <Subject>
11      <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
12       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
13        >employee</AttributeValue>
14       <SubjectAttributeDesignator
15        AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
16        DataType="http://www.w3.org/2001/XMLSchema#string"/>
17      </SubjectMatch>
18     </Subject>
19    </Subjects>
20    <Resources>
21     <Resource>
22      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
23       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">CNR</AttributeValue>
24       <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType=
               "http://www.w3.org/2001/XMLSchema#string"/>
25      </ResourceMatch>
26     </Resource>
27    </Resources>
28    <Actions>
29     <Action>
30      <ActionMatch
31       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
32       <AttributeValue
33        DataType="http://www.w3.org/2001/XMLSchema#string">acces</AttributeValue>
34       <ActionAttributeDesignator
35        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
36        DataType="http://www.w3.org/2001/XMLSchema#string"/>
37      </ActionMatch>
38     </Action>
39    </Actions>
40   </Target>
41   <Rule RuleId="Rule1" Effect="Permit">
42   <Description> A subject who is employee can access to its office room or to
43    the common room from 8am to 8pm.</Description>
44   <Target>
45    <Resources>
46     <Resource>
47      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
48       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">CNR</AttributeValue>
49       <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType=
               "http://www.w3.org/2001/XMLSchema#string"/>
50      </ResourceMatch>
51      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
52       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">office room</AttributeValue>
53       <ResourceAttributeDesignator AttributeId="urn:cnr:names:office:id" DataType="http://www.w3.org/2001/
               XMLSchema#string"/>
54      </ResourceMatch>
55     </Resource>
56     <Resource>
57      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
58       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">CNR</AttributeValue>
59       <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType=
               "http://www.w3.org/2001/XMLSchema#string"/>
60      </ResourceMatch>
61      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
62       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">common room</AttributeValue>
63       <ResourceAttributeDesignator AttributeId="urn:cnr:names:common:id" DataType="http://www.w3.org/2001/
               XMLSchema#string"/>
64      </ResourceMatch>
65     </Resource>
66    </Resources>
67    <Environments>
68     <Environment>
```

```
    <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:time-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">8:00:00</AttributeValue>
     <EnvironmentAttributeDesignator
      AttributeId= "urn:oasis:names:tc:xacml:1.0:environment:current-time"
      DataType="http://www.w3.org/2001/XMLSchema#time"/>
    </EnvironmentMatch>
   </Environment>
   <Environment>
    <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal">
     <AttributeValue
      DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration">PT12H</
          AttributeValue>
     <EnvironmentAttributeDesignator
      AttributeId= "urn:oasis:names:tc:xacml:1.0:environment:current-time"
      DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration␣"/>
    </EnvironmentMatch>
   </Environment>
  </Environments>
 </Target>
</Rule>
<Rule RuleId="Rule2" Effect="Permit">
 <Description> A subject who is employee and he is biologist or physician can access to the LCR
  from 8am to 8pm.</Description>
  <Target>
   <Resources>
    <Resource>
     <ResourceMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
       >LCR</AttributeValue>
      <ResourceAttributeDesignator
       AttributeId="urn:cnr:names:lab:id"
       DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </ResourceMatch>
    </Resource>
   </Resources>
   <Environments>
    <Environment>
     <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:time-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">8:00:00</AttributeValue>
      <EnvironmentAttributeDesignator
       AttributeId= "urn:oasis:names:tc:xacml:1.0:environment:current-time"
       DataType="http://www.w3.org/2001/XMLSchema#time"/>
     </EnvironmentMatch>
    </Environment>
    <Environment>
     <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal">
      <AttributeValue
       DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration">PT12H</
           AttributeValue>
      <EnvironmentAttributeDesignator
       AttributeId= "urn:oasis:names:tc:xacml:1.0:environment:current-time"
       DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration␣"/>
     </EnvironmentMatch>
    </Environment>
   </Environments>
  </Target>
  <Condition>
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
    <SubjectAttributeDesignator SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-
         subject" AttributeId="current-context" DataType="http://www.w3.org/2001/XMLSchema#string" Issuer=
         "admin"/>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">biologist</AttributeValue>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">physician</AttributeValue>
   </Apply>
  </Condition>
 </Rule>
<Rule RuleId="Rule3" Effect="Permit">
 <Description> A subject who is employee and he is a technician can access to LCR.</Description>
 <Target>
  <Subjects>
   <Subject>
    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
      >technician</AttributeValue>
     <SubjectAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
   </Subject>
  </Subjects>
  <Resources>
   <Resource>
    <ResourceMatch
     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
      >LCR</AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:cnr:names:office:id"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
```

```
155       </ResourceMatch>
156      </Resource>
157     </Resources>
158     </Target>
159   </Rule>
160   <Rule RuleId="default" Effect="Deny"/>
161   </Policy>
```

Listing 1. Smart environment access policy

In particular, three kinds of employees (subjects) are defined: *Biologist, Physician* and *Technician*. There are also three types of rooms (resources): *common room, office room*, and *LCR*. The required action is always *access*, while the environment is represented by the different time slots, in which an employee can access the different rooms. Specifically, each employee can access his own office and the common rooms during the business-time (from 8am to 8pm) (Rule1, line 41 of Listing 1). Only an employee, who is either *Biologist* of *Physician*, can access the *LCR* during the business-time (Rule2, line 88 of Listing 1). The *Technician* can access the *LCR* at any time (Rule3, line 132 of Listing 1).

The request, sent by the *CEP - Events* to the *Access Control Engine*, is evaluated by the PDP component and the corresponding reply is sent back to the *CEP - Usage*, which uses the received (permit or deny) response to allow the resource access or to deny it in case of possible violations or resource misuses. In both cases, the *CEP - Usage* is in charge of notifying the Actuators of the (corrective) actions to be executed. An example of usage rule implemented by the *CEP - Usage* is shown in listing: 2. In particular, the rule checks if there are pending access requests to the *LCR* and ongoing alarms. In this last case, it retrieves from the employee data base the contact data of the technician in charge of managing the alarm and it inhibits any possible access to *LCR*, apart from the selected technician.

In the following subsection, the above mentioned scenario is described in more details.

5.3 Maintenance activity scenario

This section describes the management of the scenario in which an alarm is raised by the sensors and a corrective maintenance request is sent to the technician. The scenario preconditions are the following: i) Each employee is registered on the internal ISTI-CNR personal data base; ii) Each employee accesses the different rooms bymeans of a personal badge equipped with a beacon bluetoot, as shown in Figure: 2; iii) The *LCR* room is closed by default and constantly monitored by sensors able to send events to the Monitoring Infrastructure; iv) No one is currently inside the LCR and sensor values are within their allowed ranges.

Initially, a *Physician* requires to access the LCR by using the LCR RFID badge reader connected the to nearest network. According to the interaction described in Section 5.2, an event is sent to the *CEP - Events* through the *Message Broker* and the proper access request is sent to the *Access Control Engine*. This last evaluates the request and sends back the response to the *PEP*, which in turn sends back to the *CEP - Usage* through the *Rest Engine*.

As shown in Listing 3, if: any revocation of permission is ongoing (line 8), there are not critical conditions (i.e. the values of temperature, humidity, energy consumption, noise are in the allowed ranges - line 18), and PDP response includes a permit (i.e. *Physician* requires to access the LCR during the business-time -line 15), the *CEP - Usage* sends an event to the *Actuator gateway* for enabling the door opening through the *Response Dispatcher*.

Supposing instead that a critical condition has been detected by *CEP - Events*, for instance either the power consumption sensors is out of range or there are significant variations in the noise or the temperature is in a not allowed range, a *SensorFailureEvent* event is sent to the *CEP - Usage* (line 46 of Listing 2). This last overrides the PDP response allowing the access to the technician only and sends an event to the *Actuator gateway* for enabling the door opening to the technician only (line 50-51 of Listing 3).

Moreover, the *CEP - Usage* sends an alarm event to the Supervision through a specific *Actuator gateway* for requesting exceptional maintenance of the LCR (line 37 of Listing 3).

```
1  [..setup and import omitted..]
2
3  declare SensorFailureEvent
4   @idroom: int
5   @idsensor: int
6  end
7
8  rule "Check␣data␣from␣temperature␣sensor"
9    no-loop true
10   salience 1
11   dialect "java"
12 when
13   $aEvent:GlimpseBaseEventSB(this.isConsumed == false, this.isException == false,
14     (this.getTemperature == null || < -20 || > 0 ) );
15
16   $bEvent:GlimpseBaseEventSB(
17     this.isConsumed == false, this.isException == false,
18     (this.getTemperature == null || < -20 || > 0 ),
19     this after $aEvent, this.getSensorID == $aEvent.getSensorID);
20 then
21   SensorFailureEvent failureDetected = new SensorFailureEvent(idRoom,idSensor);
22   CepBinder.sendEventTo("CEP␣-␣Usage", failureDetected);
23   $aEvent.setConsumed(true);  $bEvent.setConsumed(true);
24   retract($aEvent);  retract($bEvent);
25 end
```

Listing 2. Business Rule

```
1  [..setup and import omitted..]
2
3  declare SensorFailureEvent
4   @idroom: int
5   @idsensor: int
6  end
7
8  rule "If␣there␣are␣NOT␣pending␣alarm␣forward␣PDP␣access␣response"
9  no-loop true
10 salience 1
11 dialect "java"
12
13 when
14   $aEvent:PdpAccessRequest();
15   $bEvent:PdpAccessResponse(this.isConsumed == false, this.isException == false,
16     this.getIdRequest == $aEvent.getIdRequest, ($bEvent.getResponse == "Permit" || "Deny"),
17     this after $aEvent);
18   not(SensorFailureEvent(this.isConsumed == false, this.isException == false,
19     this.idRoom == $aEvent.idRoom, this.idSensor == $aEvent.idSensor));
20
21 then
22   Actuators.ManageAccess($aEvent.getIdSensor(),$aEvent.getIdroom(), $bEvent.getResponse()));
23 end
24
25 rule "If␣there␣are␣failures␣take␣countermeasures"
26 no-loop true
27 salience 1
28 dialect "java"
```

```
29
30  when
31    $aEvent:SensorFailureEvent();
32  then
33    Alarm.NotifyToSupervision($aEvent.idsensor, $aEvent.idroom);
34  end
35
36  rule "If there are pending alarm check accesses"
37  no-loop true
38  salience 1
39  dialect "java"
40
41  when
42    $aEvent:PdpAccessRequest();
43    $bEvent:PdpAccessResponse(this.isConsumed == false, this.isException == false,
44      this.getIdRequest == $aEvent.getIdRequest, ($bEvent.getResponse == "Permit" || "Deny"),
45      this after $aEvent);
46    $cEvent:SensorFailureEvent(this.isConsumed == false, this.isException == false,
47      this.idRoom == $aEvent.idRoom);
48
49  then
50    Actuators.ManageAccess($aEvent.getIdSensor(),$aEvent.getIdroom(),
51              PersonnelDatabase.checkIfIsTechnician($aEvent.getIdUser));
52  end
```

Listing 3. Usage Rule

6 Related Work

This work spans over several research directions, including: smart environment, access and usage control and monitoring approaches.

Enabling Platforms in Smart Environments The SE paradigm depends on communication and cooperation between numerous devices, sensor networks embedded in the environment itself, servers in a fixed infrastructure and the increasing number of mobile devices carried by people. In order to enable the SE paradigm, diverse platforms and software infrastructures have been proposed in the literature [27]. Among these, FI-WARE [8] is emerging as a core standard platform for Smart and Connected Communities (SCC) [28, 29]. The FI-WARE project is producing new tools to facilitate the development of application and fostering a major inclusion of software standards for smart cities [30]. These tools are provided as Generic Enablers (GE): software components that can be configured and deployed on a cloud platform in order to easily implement an application. Another important enabling platform for SE is represented by the universAAL architecture [9], with a particular focus on IoT and Ambient Assisted Living (AAL) [31]. Besides its concrete open source implementation, universAAL proposes an architectural reference model based on a set of virtual communication buses as semantic brokers for context events, semantic service requests (and responses), and user interaction functions. For these reasons, we used the separation of concerns and the service discovery capabilities offered by the universAAL reference model to build the middleware architecture proposed in this paper.

Access and Usage Control Concerning the testing of access control policies, combinatorial approaches have been proven to be effective in the automated generation of the test cases (access requests). Among the various proposals, the Targen

[8] http://www.fiware.org
[9] http://www.universaal.info/

tool [32] generates test inputs by using combinatorial coverage of the truth values of independent clauses of XACML policy values, while the X-CREATE tool [33] relies on combinatorial approaches of the subject, resource, action and environment values taken from the XACML policy. The main advantage of this approach with respect to Targen is the higher structural variability of the derived test inputs able to guarantee the coverage of the input domain of the XACML policy. Other works address model-based testing and provide a methodology for the generation of test cases based on combinatorial approaches of the elements of the model (role names, permission names, context names).

Concerning the Usage Control systems, the work in [34] proposes a usage control model based on UCON and describes a framework to implement it in an operating system kernel, on top of the existing DAC mechanism. Other available solutions, such as [35], propose proactive mechanisms for preventing possible policy violations and present a combination of runtime monitoring and self-adaptation to simplify the autonomic management of authorization infrastructures. In recent years, as surveyed in [36], testing of authorization systems has been focused on evidencing the appropriateness of the UCON enforcement mechanism, focusing on the performance analysis or establishing proper enforcement mechanisms by means of formal models. Finally, the authors of [37] address the testing of the Policy Decision Point (PDP) implementation within the PolPA authorization system, which enables history-based and usage-based control of accesses proposing two testing strategies specifically conceived for validating the history-based access control and the usage control functionalities of the PolPA PDP.

Monitoring Several general-purpose monitoring proposals are currently available, which can be mainly divided into two groups: those that are embedded in the execution engine, such as [38, 39], and those that can be integrated into the execution framework as an additional component, such for instance [40, 41, 42]. Both the solutions have specific advantages. For sure, an embedded solution reduces the performance delay of the execution framework, mainly in terms of interactions and communication time. Coverage indicators can be directly evaluated by the execution framework, which can also execute corrective actions in case of important deviations. The main disadvantage of these approaches is the lack of flexibility in the data collection, the coverage measure definition and the language adopted.

7 Conclusions

In this paper, we proposed an infrastructure for runtime management and control of smart environments. The main advantages of the proposed solution are its flexibility and the possibility of decoupling the different levels of rules that are defined and implemented for managing the resources of the smart environments and regulate the access to them. Specifically, three levels of rules are defined: the *Business Rules* for correlating sensors data and technologies; the *Usage Control*

Rules, which define the users and sensors interactions; and the *Access Control Rules*, which manage the accesses to the different resources expressed through a specific control policy formalism. This allows an easy maintenance and updating of control rules when context changes or constraints violations take place.

A first validation on a real cased study, considering a medical cold storage and implementing an XACML policy, has been described. The presented scenario evidenced the effectiveness of the proposed approach to correlate events generated by different sensors and to leverage different levels of rules for raising alarms, when critical situations are detected.

As a future work, we would like to validate the proposed solution in other smart environments with different peculiarities and security constraints as well as different access control policy specification languages. Moreover, we plan to extend the infrastructure to include more refined levels of rules, further decoupling the management and control functionalities of the proposed infrastructure.

Acknowledgments

This work has been partially funded by the projects GAUSS, a National Research project (MIUR, PRIN 2015, Contract 2015KWREMX), and SIGS (FAR/FAS 2014-2016 research program of the Tuscany Region).

References

1. Weiser, M.: The computer for the 21st century. Scientific american **265**(3) (1991) 94–104
2. Ryan, N., Cinotti, T.S., Raffa, G.: Smart environments and their applications to cultural heritage. Smart Environments and their Applications to Cultural Heritage (2005) 7
3. Tragos, E.Z., Bernabe, J.B., Staudemeyer, R.C., Luis, J., Ramos, H., Fragkiadakis, A., Skarmeta, A., Nati, M., Gluhak, A.: Trusted iot in the complex landscape of governance, security, privacy, availability and safety. Digitising the Industry–Internet of Things Connecting the Physical, Digital and Virtual Worlds. River Publishers Series in Communications (2016) 210–239
4. Bertolino, A., Daoudagh, S., Lonetti, F., Marchetti, E.: An automated testing framework of model-driven tools for XACML policy specification. In: 9th QUATIC 2014, Guimaraes, Portugal, September 23-26, 2014. (2014) 75–84
5. Daoudagh, S., Kateb, D.E., Lonetti, F., Marchetti, E., Mouelhi, T.: A toolchain for model-based design and testing of access control systems. In: MODELSWARD 2015 Angers, Loire Valley, France, 9-11 February, 2015. (2015) 411–418
6. Palumbo, F., Barsocchi, P., Furfari, F., Ferro, E.: Aal middleware infrastructure for green bed activity monitoring. Journal of Sensors **2013** (2013)
7. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer **29**(2) (1996) 38–47
8. Standard, O.: extensible access control markup language (xacml) version 2.0 (2005)
9. Park, J., Sandhu, R.: The ucon abc usage control model. ACM Transactions on Information and System Security (TISSEC) **7**(1) (2004) 128–174

10. Calabrò, A., Lonetti, F., Marchetti, E.: Monitoring of business process execution based on performance indicators. In: 41st,EUROMICRO-SEAA 2015, Madeira, Portugal, August 26-28, 2015. (2015) 255–258
11. Calabrò, A., Lonetti, F., Marchetti, E.: KPI evaluation of the business process execution through event monitoring activity. In: ES 2015, Basel,Switzerland, October 14-15, 2015. (2015) 169–176
12. Czarnecki, K., Eisenecker, U.W.: Generative programming - methods, tools and applications. Addison-Wesley (2000)
13. Bertolino, A., Calabrò, A., Angelis, G.D.: A generative approach for the adaptive monitoring of SLA in service choreographies. In: Web Engineering - 13th International Conference, ICWE 2013, Aalborg, Denmark, July 8-12, 2013. Proceedings. (2013) 408–415
14. Drools, J.: Drools Fusion: Complex Event Processor `http://www.jboss.org/drools/drools-fusion.html`.
15. Wilde, E., Pautasso, C., eds.: REST: From Research to Practice. Springer (2011)
16. Oracle: Java message service documentation. Technical report (2016) `http://www.oracle.com/technetwork/java/docs-136352.html`.
17. Apache: Apache ActiveMQ `http://activemq.apache.org/`.
18. Barsocchi, P., Oligeri, G., Potortì, F.: Measurement-based frame error model for simulating outdoor wi-fi networks. IEEE Transactions on Wireless Communications **8**(3) (2009) 1154–1158
19. Furfari, F., Girolami, M., Lenzi, S., Chessa, S.: A service-oriented zigbee gateway for smart environments. Journal of Ambient Intelligence and Smart Environments **6**(6) (2014) 691–705
20. Palumbo, F., Ullberg, J., Štimec, A., Furfari, F., Karlsson, L., Coradeschi, S.: Sensor network infrastructure for a home care monitoring system. Sensors **14**(3) (2014) 3833–3860
21. Barsocchi, P., Ferro, E., Fortunati, L., Mavilia, F., Palumbo, F.: EMS@CNR: An energy monitoring sensor network infrastructure for in-building location-based services. In: High Performance Computing & Simulation (HPCS), 2014 International Conference on, IEEE (2014) 857–862
22. Barbon, G., Margolis, M., Palumbo, F., Raimondi, F., Weldin, N.: Taking Arduino to the Internet of Things: the ASIP programming model. Computer Communications **89** (2016) 128–140
23. Palumbo, F., La Rosa, D., Chessa, S.: Gp-m: Mobile middleware infrastructure for ambient assisted living. In: Computers and Communication (ISCC), 2014 IEEE Symposium on, IEEE (2014) 1–6
24. Kim, Y., Schmid, T., Charbiwala, Z.M., Srivastava, M.B.: Viridiscope: design and implementation of a fine grained power monitoring system for homes. In: Proceedings of the 11th international conference on Ubiquitous computing, ACM (2009) 245–254
25. Girolami, M., Palumbo, F., Furfari, F., Chessa, S.: The integration of zigbee with the giraffplus robotic framework. In: International Joint Conference on Ambient Intelligence, Springer (2013) 86–101
26. CNR: Smartcampus technical report. Technical report (2016) `http://www.smart-applications.area.pi.cnr.it/doc/smartareaTechnicalDescription.pdf`.
27. Namiot, D., Sneps-Sneppe, M.: On software standards for smart cities: Api or dpi. In: ITU Kaleidoscope Academic Conference: Living in a converged world-Impossible without standards?, Proceedings of the 2014, IEEE (2014) 169–174

28. Glikson, A.: Fi-ware: Core platform for future internet applications. In: Proceedings of the 4th annual international conference on systems and storage. (2011)
29. Sun, Y., Song, H., Jara, A.J., Bie, R.: Internet of things and big data analytics for smart and connected communities. IEEE Access **4** (2016) 766–773
30. Ramparany, F., Marquez, F.G., Soriano, J., Elsaleh, T.: Handling smart environment devices, data and services at the semantic level with the fi-ware core platform. In: Big Data 2014, IEEE (2014) 14–20
31. Salvi, D., Montalva Colomer, J.B., Arredondo, M.T., Prazak-Aram, B., Mayer, C.: A framework for evaluating ambient assisted living technologies and the experience of the universaal project. Journal of Ambient Intelligence and Smart Environments **7**(3) (2015) 329–352
32. Martin, E., Xie, T.: Automated Test Generation for Access Control Policies. In: Supplemental Proc. of 17th International Symposium on Software Reliability Engineering (ISSRE). (November 2006)
33. Bertolino, A., Daoudagh, S., Lonetti, F., Marchetti, E., Schilders, L.: Automated testing of extensible access control markup language-based access control systems. IET Software **7**(4) (2013) 203–212
34. Teigao, R., Maziero, C., Santin, A.: Applying a usage control model in an operating system kernel. Journal of Network and Computer Applications **34**(4) (2011) 1342–1352
35. Bailey, C.: Application of self-adaptive techniques to federated authorization models. In: Software Engineering (ICSE), 2012 34th International Conference on, IEEE (2012) 1495–1498
36. Nyre, Å.: Usage control enforcement-a survey. Availability, Reliability and Security for Business, Enterprise and Health Information Systems (2011) 38–49
37. Bertolino, A., Daoudagh, S., Lonetti, F., Marchetti, E., Martinelli, F., Mori, P.: Testing of polpa-based usage control systems. Software Quality Journal **22**(2) (2014) 241–271
38. Daoudagh, S., Lonetti, F., Marchetti, E.: Assessment of access control systems using mutation testing. In: TELERISE 2015, Florence, Italy, May 18, 2015. (2015) 8–13
39. Bertolino, A., Daoudagh, S., Kateb, D.E., Henard, C., Traon, Y.L., Lonetti, F., Marchetti, E., Mouelhi, T., Papadakis, M.: Similarity testing for access control. Information & Software Technology **58** (2015) 355–372
40. Carvallo, P., Cavalli, A.R., Mallouli, W., Rios, E.: Multi-cloud applications security monitoring. In: 12th GPC 2017, Cetara, Italy, May 11-14, 2017, Proceedings. (2017) 748–758
41. Bertolino, A., Calabrò, A., Lonetti, F., Marchetti, E.: Towards business process execution adequacy criteria. In: 8th SWQD 2016, Vienna, Austria, January 18-21, 2016, Proceedings. (2016) 37–48
42. Calabrò, A., Lonetti, F., Marchetti, E., Spagnolo, G.O.: Enhancing business process performance analysis through coverage-based monitoring. In: 10th QUATIC 2016, Lisbon, Portugal, September 6-9, 2016. (2016) 35–43