Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 32 (2014) 997 – 1002

Procedia Computer Science

International Workshop on Enabling ICT for Smart Buildings (ICT-SB 2014)

Using a Residential Environment Domain Ontology for Discovering and Integrating Smart Objects in Complex Scenarios

Abdaladhem Albreshne[a*], Ayoub Ait Lahcen[b], Jacques Pasquier[a]

[a]*DIUF, University of Fribourg, 1700 Fribourg, Switzerland*
[b]*ENSA (Ecole Nationale des Sciences Appliquées), Kenitra, Morocco*

Abstract

Nowadays, in residential environments context, an increasing number of connected devices and objects (that fall broadly into two categories: actuators and sensors) provide useful functionalities and services to improve urban living conditions. These devices and their associated services, however, cannot be easily integrated to create new composite applications, such as controlling different home devices for saving energy or enhancing user comfort. In order to meet this requirement, we propose a domain ontology to discover, search, use and share information and services of residential environment devices. This ontology forms the central part of our proposed application development framework. In this paper, we briefly introduce this ontology and explain, through examples, how it is used during the different steps of an application development process.

Keywords: Smart Residential Environments; Ontology; Semantic Discovery and Composition; Sensors; Actuators; Domain Specific Language; BPEL4WS; OWL.

1. Introduction

Recent researches on smart environments show that there are interesting challenges in developing composite applications that integrate multiple services provided by different devices[1]. To deal with these challenges, several development paradigms have been suggested in the literature[2]. One of them is Services Oriented Computing[3], which has been proposed as a solution for heterogeneous and changing environments. In spite of that, a number of open questions in controlling Smart Environments are still being explored by both industrial and academic communities. We distinguish two main problems in this context. The first one is how to describe the environment and devices that

* Corresponding author. Tel.: +41-26-3008329; fax: +41-26-3009726
E-mail address: abdaladhem.albreshne@unifr.ch

doi:10.1016/j.procs.2014.05.524

users, respectively, live in and interact with. The second one is the problem of modeling, discovering, orchestrating and executing services in smart environments, allowing users to control them. To address these limitations, we have proposed in[4] a generic software framework for managing Smart Residential Environments (SRE). The latter has been designed to increase the accessibility of physical entities in SRE and to help integrate them into applications, with the help of an original Domain Specific Language (DSL) called GPL4SE (Generic Process Language for Smart Environments). The interested reader is strongly invited to have a quick look on this paper in order to get a general understanding of our SRE software development framework.

In the present paper, we mostly focus on the use of our domain ontology, that forms the central part of our proposed framework. In particular, we will show how it is used for both discovering and compositing services. In fact, the defined ontology describes how actuators and sensors are related to smart entities (locations, persons or objects) and which services (actions, queries or events) they provide. It allows for discovering the needed services and provides groundings to access and compose them. The rest of this paper is organized as follows. Section 2 introduces a motivating scenario that will serve as a guiding thread for the remaining sections. Sections 3, 4 and 5 respectively explain, through the implementation of the motivating scenario, the ontology instantiation, the service registry and the discovery process. Finally, Section 6 briefly presents some related works.

2. Motivating scenario

In order to illustrate the usefulness of our approach, let us explain it through the implementation of a simple scenario that takes place in a smart home (see Fig.1(a)). The home contains a set of smart entities (smart rooms, doors, lamps, and heaters). The Home Control System supervises the smart entities in order to achieve some user needs (e.g., saving energy, enhancing comfort, security, healthcare, etc.). Our simple scenario consists in switching off all lamps once the system is informed that home's occupants are outside.

It might seem that the proposed scenario is overly simplistic to show the full advantages of our approach. One, however, should keep in mind the "limited goal" of this paper, which consists in describing the concept of our ontology, as well as the whole process of using it to instantiate smart entities, which can then be discovered and managed within a scenario written in GLP4SE. The latter is indeed able to manage much more complex scenarios with events coming in parallel and list of smart entities filtered through generic and/or specific properties such as all the windows located in a given space or the air-conditioner associated with Room X. For a more detailed insight, the interested reader is referred to[4] where an extended version of the scenario is presented, including switching lamps on and off according to presences in the various rooms; taking actions to keep the temperature within a reasonable range; and controlling that one does not use the shower more than a given interval of time.

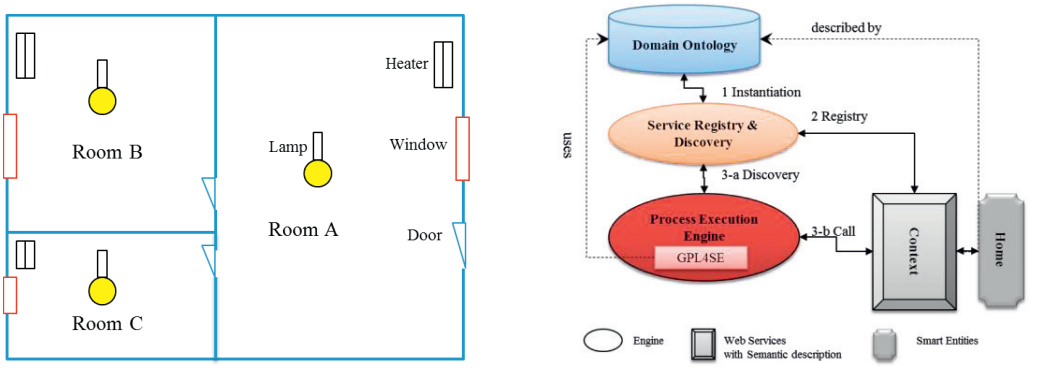

Fig. 1. (a) The Smart Home of our scenario (inspired from Fig.1 of [4]); (b) Service registry, discovery and call (action) processes

In order to perform the proposed plan, the system not only needs to have a precise idea of the involved actions (*switchoff*), but also to be able to link them with the actuators and/or sensors of the concerned entities (*lamps*). This is performed in 3 steps as shown in Fig.1(b):

1. The ontology is instantiated for the given smart home.

2. When the actuators and sensors are installed in the smart home, the address of their WSDL[5] (Web Service Description Language) files and ontology-based semantic description documents (OWL-S[6]: Semantic Markup for Web Services) are published in the service registry.
3. Upon starting the execution of the actions, the execution engine of the framework invokes the discovery engine to find all the necessary services and matches them with their corresponding entities. An ontology inference mechanism (e.g., SPARQL[7]:Protocol and RDF Query Language) is used to perform this task.

3. The ontology instantiation

For a given smart home, the configuration of the environment corresponds to an instantiation of the ontology classes presented in Fig. 3. Note that the ontology proposes a hierarchy of entities (Location, Room, Object), which are smart because they have attached sensors (providing queries and, through their publishers, events services) and actuators (providing actions services). To illustrate that, an extract of the OWL[8] (Web Ontology Language) file that contains the instantiation of Room A is given in Fig. 2. This extract of code is organized as follows:

- Lines 1-32 define that the instance *Room_1* which has an entityID *Room A,* is a type of the class *Room*.
- Lines 3-20 define that the Location *Room A* has the entity instance *Lamp_1*.
- Lines 5-18 define that the instance *Lamp_1* has the actuator instance *LampController_1*.
- Lines 7-10 define that the instance *LampController_1* has *switchOff* action.
- Lines 21-31 define that the instance *Room_1* has the sensor instance *AccessSensor_1* with an event publisher.

```
1. <diuf:Room rdf:ID="Room_1">
2.   <diuf:entityID rdf:datatype="#string">Room A</diuf:entityID>
3.   <diuf:hasEntity>
4.     <diuf:Lamp rdf:ID="Lamp_1">
5.       <diuf:hasActuator>
6.         <diuf:LampController rdf:ID="LampController_1">
7.           <diuf:hasAction>
8.             <diuf:switchOff rdf:ID="switchOff_1">
9.               <diuf:owlsLocation rdf:datatype="#string"></diuf:owlsLocation>
10.            </diuf:switchOff>
11.            . . . . . .
15.          </diuf:hasAction>
16.        <diuf:actuatorID rdf:datatype="#string">4754</diuf:actuatorID>
17.      </diuf:LampController>
18.    </diuf:hasActuator>
19.  </diuf:Lamp>
20.  </diuf:hasEntity>
21.  <diuf:hasSensor>
22.      <diuf:AccessSensor rdf:ID="AccessSensor_1">
23.        <diuf:hasPublisher>
24.          <diuf:AccessEventPublisher rdf:ID="AccessPublisher_1">
25.            <diuf:hasEvent>
26.              <diuf:AccessEvent rdf:ID="AccessEvent_1"></diuf:AccessEvent>
27.            </diuf:hasEvent>
28.         </diuf:AccessEventPublisher>
29.       </diuf:hasPublisher>
30.    </diuf:AccessSensor>
31.  </diuf:hasSensor>
32. </diuf:Room>
```

Fig. 2. Code extract of a smart Lamp instantiation and its corresponding services

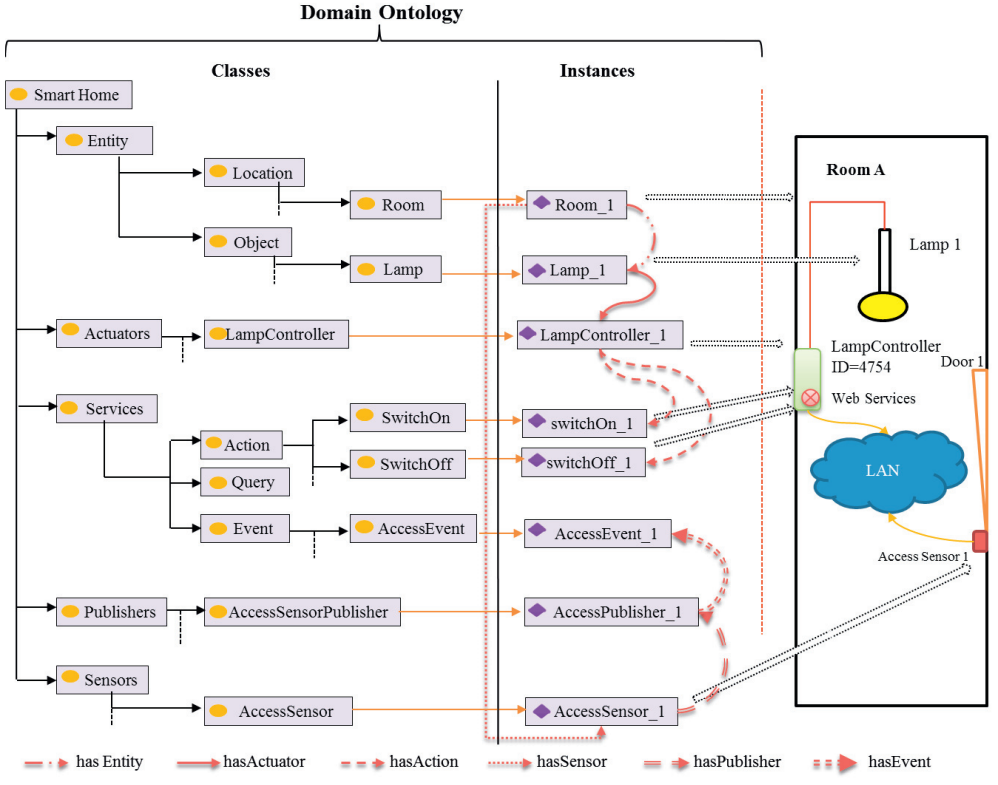

Fig. 3. The domain ontology (simplified) and its instantiation for a specific home

4. Service registry

Once the ontology has been partially instantiated (without grounding), the final registry process is started by an actuator or a sensor associated to a smart entity. It sends first a request to be added to the service providers list using SOAP[9] protocol. Then, its specifications are registered into the service registry. This information consists of a device Identifier, a list of available services and their descriptions. As an example, an extract of a SOAP message sent by the actuator (service provider) *LampController* is given in Fig. 4. It shows that the actuator is identified by actuator ID "4754" and provides two services *switchOn* and *switchOff*. These services can be called using their WSDL and OWL-S URL addresses.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body xmlns:ns1="http://ws/"><ns1:registreActuator><id>4754</id>
        <service><name>switchOn</name>
            <owlURL>http://diufpc10:8080/SmartHome/LivingRoomLamp.owl</owlURL>
            <wsdlURL>http://diufpc10:8080/SmartHome/LivingRoomLamp?WSDL</wsdlURL>
        </service>
        <service><name>switchOff</name>
            <owlsURL>http://diufpc10:8080/SmartHome/LivingRoomLamp.owl</owlsURL>
            <wsdlURL>http://diufpc10:8080/SmartHome/LivingRoomLamp?WSDL</wsdlURL>
        </service>
    </ns1:registreActuator></soap:Body>
</soap:Envelope>
```

Fig. 4. SOAP message sent by a LampController actuator

5. Using the smart objects (the discovery process)

Once the ontology has been instantiated and all services registered, diverse scenarios can be simply written with our GPL4SE language. With the latter, it is possible to discover and interact with different smart entities, create loops, declare variables, copy and assign values, as well as to register to and wait for events from sensors publishers. Scenarios execution can be sequential or parallel. Additionally, GPL4SE offers control flow constructs (while, if, etc.). To illustrate the discovery process and how GPL4SE interacts with smart entities, an extract of code and its corresponding SPARQL query is given in Fig.5. The program structure is divided into three parts: a discovery section, an event registry section and an execution section. The first section consists of the semantic discovery process which begins by sending a request using the *discover* keyword. The request has two parameters: the semantic description of the required entities and their locations (optional). Once the request is received by the Discovery Engine, it is automatically translated by the discovery engine into a SPARQL query to find all smart entities in Room A. The second section is to subscribe the process to the AccessEvent to be informed about any new change which may occur. When an Access Event is fired, the process then executes the appropriate onEvent action.

```
List lamps=discover(Lamp, "Room A");

subscribe AccessEvent;

OnEvent(AccessEvent, Map accessInput){
 //check if nobody at home
 If(accessInput.get("PresenceState")== false){
    // A for loop to iterate over each lamp
    for (Lamp lamp: lamps){
       // SwitchOff this specific lamp.
      action (lamp, "switchOff");
    }
 }
}
SELECT ?entityType ?locationType ?action ?actionType
       ?wsdlLocation ?owlsLocation ?locationID
  WHERE{
     ?entity rdf:type ?entityType.
     filter (?entityType = diuf:Lamp).
     ?location diuf:hasEntity ?entity.
     ?location diuf:locationID ?locationID.
     filter (?locationID = 'Room A').
     ?location  rdf:type  ?locationType.
     ?entity diuf:entityName ?entityName.
     ?entity diuf:hasActuator ?actuator.
     ?actuator diuf:actuatorID ?id.
     ?actuator diuf:hasAction ?action.
     ?action a ?actionType.
     ?actionType rdfs:subClassOf diuf:Action.
     ?action diuf:wsdlLocation ?wsdlLocation.
     ?action diuf:owlsLocation ?owlsLocation.
  }
```

Lamps Discovery section

Event Registry Section

Event Handler Section

Scenario template written with GPL4SE

SPARQL request at Discovery Engine

Fig. 5. A GPL4SE extract of code and its corresponding SPARQL query

6. Related work

In order to improve interoperability between heterogeneous systems, a range of research works have been proposed, over the recent past years, to apply semantic web technologies in smart computing environments. For instance, ontologies have been used to represent, model, reason, manipulate and access context information in smart environments[10,11]. However, few of these works focus on developing composite applications that integrate discovered services[2,12]. Some efforts like[13,14] use semantic description and ontology to help in improving both discovery and composition operations. As far as we are concerned, we didn't reinvent the wheel, but we offer a modular software solution to cleanly integrate proven technologies for modeling and controlling at a high level of abstraction smart residential environments. To achieve this goal, we offer: (1) an ontology model for smart environments (homes, hospitals, schools, etc.), which serves as the glue between the entities of the physical world and the services of the virtual one and (2) a generic process language to control such environment which allows for describing different scenarios using a precise vocabulary without having to program intricacies such as services grounding, and which is then transparently translated into a standard process language (e.g. BPEL4WS[15]:Business Process Execution Language for Web Services).

7. Conclusion

In this paper, we proposed an ontology-based framework in order to provide greater ease in the specification and execution of complex smart residential environment scenarios. We showed how the proposed ontology is used to provide a significant facilities for the discovery of smart entities and how it used to represent the whole environment in order to clearly answer questions such as: what are the actuators and the sensors present in a smart space; to which smart entities are they attached to; and which are the services (actions, queries and/or events) that they might provide ? In fact, our solution introduces the whole power of a standard query language such as SPARQL into our scenario process language.

References

[1] Sadri F.Ambient Intelligence. A Survey. In: ACM Computing Surveys (CSUR), New York; 2011. pp. 36:1-36:66.
[2] Thanos, G. Stavropoulos, Dimitris, V., Ioannis V. A survey of service composition in ambient intelligence environments. In: Artificial Intelligence Review Journal, Springer, Netherlands; 2011.
[3] Degeler V., Lopera L.I., Leva M., Shrubsole P., Bonomi S., Amft O, Lazovik A. Service-Oriented Architecture for Smart Environments. In: Proceedings of the IEEE International Conference on Service Oriented Computing and Applications (SOCA); 2013.
[4] Albreshne A., Ait Lahcen A, Pasquier J. A Framework and its Process Oriented Domain Specific Language for managing Smart Residential Environments. In: International Journal of Smart Home; 2013. Vol. 7, No. 6, pp. 377-392.
[5] W3C. Web Services Description Language. http://www.w3.org/TR/wsdl [Accessed 20.01.2014].
[6] W3C. OWL-S: Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S [Accessed 22.01.2014].
[7] W3C. SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/ [Accessed 20.01.2014].
[8] W3C, "OWL 2 Web Ontology Language", http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/ [Accessed 25.01.2014].
[9] W3C. SOAP Simple Object Access Protocol. http://www.w3.org/TR/2007/REC-soap12-part2-20070427/ [Accessed 22.01.2014].
[10] Hsien-Chou L., Chien-Chih T.. A RDF and OWL-Based Temporal Context Reasoning Model for Smart Home. In: Information Technology Journal 6 (8);2007.pp. 1130-1138.
[11] Haesung L., Joonhee K. Ontology Model-based Situation and Socially-Aware Health Care Service in a Smart Home Environment. In: International Journal of Smart Home; 2013.Vol.7, No.5, pp.239-250.
[12] Noha I., Frédéric M. A Survey on Service Composition Middleware in Pervasive Environments. In: IJCSI International Journal of Computer Science; 2009.pp. 1-12.
[13] Feng W., Kenneth .J. An Ontology-based Actuator Discovery and Invocation Framework in Home Care Systems. In: 7th International Conference on Smart Homes and Health Telematics, Springer, Berlin; 2009. pp. 66-73.
[14] Nenad S., Yongchun X. An Intelligent Event-driven Approach for Efficient Energy Consumption in Commercial Buildings. In: Proceedings of the 5th ACM International Conference on Distributed Event-based System, ACM, New York; 2011. pp. 303-312.
[15] OASIS. Web Services Business Process Execution Language Version 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html [Accessed 19.01.2014].