Agile Framework for Rapid Deployment in Ambient Assisted Living Environments

Article · November 2016

CITATIONS
0

READS
31

4 authors, including:

Joaquim Bellmunt
Institut Mines-Télécom
12 PUBLICATIONS **18** CITATIONS
SEE PROFILE

Hamdi Aloulou
Institut Mines-Télécom
31 PUBLICATIONS **159** CITATIONS
SEE PROFILE

Agile Framework for Rapid Deployment in Ambient Assisted Living Environments

Joaquim Bellmunt
IPAL CNRS
Singapore
joaquim.bellmunt@ipal.cnrs.fr

Mounir Mokhtari
IPAL CNRS
Singapore
mounir.mokhtari@ipal.cnrs.fr

Bessam Abdulzarak
Université de Sherbrooke
Sherbrooke, Canada
Bessam.Abdulrazak@USherbrooke.ca

Hamdi Aloulou
Institut Mines Télécom
Montpellier, France
hamdi.aloulou@mines-telecom.fr

ABSTRACT

Agile and rapid deployment solution, which enables to discover surrounding context and construct the semantic knowledge, is a major enabler for large scale deployment of Ambient Assisted Living (AAL)[1] solutions in aging people environments. Therefore, we built UbiSMART framework, an AAL ontology/web solution based on a shared cloud-based semantic reasoner, to enable easy and quick deployment of hardware and software component. UbiSMART is complemented by simple and user-friendly services to populate and modify the deployment model. This solution does not need any prior knowledge of the system or technical expertise. Using UbiSMART we were able to decrease the preparation time from one hour (1) to 20 min and the deployment time from four (4) hours to 20 minutes. In this paper, we present the description of our approach, the technical design, and its implementation. We also detail the validation in real scenarios and the obtained results, as well as the lessons learned and the future perspectives.

Categories and Subject Descriptors

D.2.11 [**Software engineering**]: Software Architectures; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods; L.1.0 [**Knowledge and Media**]: Knowledge Construction/Representation

Keywords

Ambient Assisted Living; Context Discovery; Web Framework; Ageing in Place, Knowledge Construction

[1]http://www.aal-europe.eu

iiWAS '16, November 28 - 30, 2016, Singapore, Singapore

ISBN 978-1-4503-4807-2/16/11.

DOI: http://dx.doi.org/10.1145/3011141.3011196

1. INTRODUCTION

Discovering the context and creating knowledge in Ambient Assisted Living (AAL) has become a key challenge when facing distant, agile, quick and large scale deployments. Nowadays, the third generation of AAL platforms [6] focus on preventing risks, monitoring, and assisting aging in place. However, real deployments have to be addressed individually to fit its end-user needs, space peculiarities or even connectivity issues. Based on our experience, a deployment of our former framework needed an hour of technical preparation and about four hours of deployment time. Moreover, most of the times an expert is required to provide these details manually and often hard coded. This procedure represents an even bigger obstacle when facing distant deployments at city scale. It adds complexity to any maintenance and makes the integration of these technologies and the involvement of end-users difficult and time consuming. Therefore, we introduce an agile framework to allow non-expert users in computer science (e.g. local technician) to rapidly deploy the sensing kit for AAL applications.

The number of the research activities that target quick easy deployment is limited. In 2006 Abdulzarak *et al.* presented 'A Smart Home in a Box' approach [1], based on a generic reference architecture for programmable pervasive spaces. This architecture makes the construction of smart spaces quick and easy and makes them more programmable and utilizable. In 2012, Aloulou *et al.* presented a semantic Plug&Play proof-of-concept in a real-world AAL framework leveraging web technologies [3]. The authors proposed the use of *abstract* and *instance* models achieving good results in terms of deployment velocity and scalability. Another study related to a do-it-yourself (DIY) smart home is presented by Woo and Lim in 2015 [9]. The researchers conducted a three weeks in situ observational study identifying six different stages when the users get familiar with a new system. They concluded that involving the user in a DIY procedure helps the integration of these technologies in their routine. Recently, in 2016, Hu *et al.* presented their evolution of the "Smart Home in a Box" [7]. The authors provided a complete plug & play solution, which can be deployed rapidly. Yet, the solution is previously prepared and configured, and the user participation is limited to following the instructions. The approach we present in this paper is an evolution of the 'A Smart Home in a Box' approach [1].

Following, Section 2 introduces our approach, Section 3 and 4 explain in detail the architecture with the technical design and implementation. Section 5 presents the validation with 40 real deployments in aging people houses, and Section 6 concludes the paper. It also presents the lessons learned and future perspectives.

2. UBISMART DESIGN

We target several goals in our research. The first goal is to develop and deploy continuous and unobtrusive AAL platforms in real scenarios. The aim is to recognize aging ADL through semantic and rule-based reasoning [8] with various level of abstraction [4]. The second goal is to move towards distant and large scale deployments of AAL platforms [5, 2]. Therefore, we built UbiSMART, an agile framework that has the advantages of using a web platform to manage a large scale deployment; the versatility and rapidity that this solution offers in a plug & play approach; and facilitates the active participation of end-users. UbiSMART is a generic, scalable and intuitive cloud Web-Based AAL framework. UbiSMART uses two ontologies: 1) a generic ontology (*abstract model*) that contains the shared properties and 2) a specific ontology (*instance model*) that contains the details of each end-user environment. This AAL platform converts any environment into a smart space in five (5) minutes to assess dependent people at their homes. The final purpose of UbiSMART is to detect the ADL and provide services at the right time and through appropriate device. It minimizes the impact upon end-users by reducing the number of sensors, however, the quality of the assessment is not affected. We have evolved the existing middleware by providing a friendly and simple web framework. This allows end-users to enter specific context information, which confers versatility and adaptability. Moreover, we have automated the installation and the maintenance of the hardware, reduced the human effort, and allowed the user to be an active part of the system. The framework disposes of dedicated end-users services which build the *instance* model without requiring prior or expert knowledge. UbiSMART is able to use the same semantic engine for many spaces by combining the *abstract* and the *instance* model. Thus, it infers decisions independently and accordingly to peculiarities of each deployment.

3. UBISMART ARCHITECTURE

The agile approach we used in UbiSMART enables to deploy in an AAL environment within 20 minutes through the web browser by any non-expert user. The system is composed of three sections (figure 1).

First, the **UbiGATE**, is located in the user environment and corresponds to the 'Smart-home in a box' package. It connects automatically to the server and the inclusion in the knowledge base (KB) is done through a dedicated service. UbiGATE is composed of multiple industrial sensors (Passive Infrared sensors (PIR) and contact sensors), an out of shell gateway (a Raspberry Pi), and a communication device (LTE or WiFi). It is in charge of processing the raw data from the sensors, converting them into events and sending them to the server via the Internet. The user receives the UbiGATE package in his environment and installs it himself by interacting with our services.

Second, the **Server**, which is the main brain of UbiSMART, handles the information from all the UbiGATES

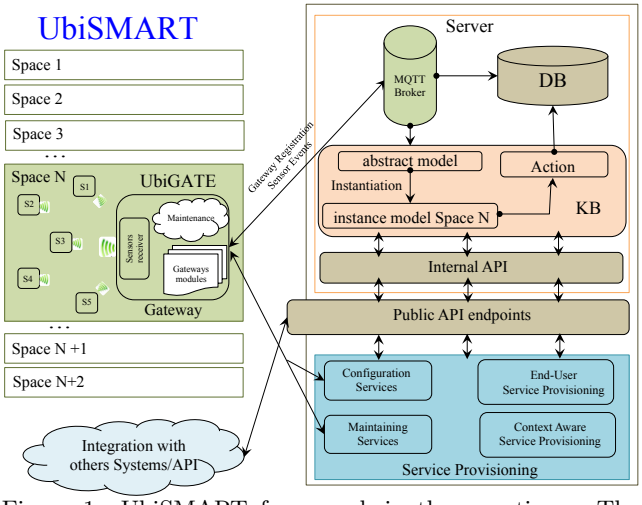


Figure 1: UbiSMART framework in three sections. The UbiGATE, the Server and the Service Provisioning.

using MQTT protocol. It employs a semantic reasoning for the activity recognition and the service provisioning. The reasoning is based on building a generic model common for any deployment called *abstract* model and the specific model for each house, *instance* model. Together, the two models describe the entire characteristics of the smart environment. The *abstract* model is initialized at the UbiSMART launch and encompasses the declaration of the variables, classes, object relations, and data relations. On the other hand, the *instance* model is completed by the interaction of the user and it populates the KB through services.

Third, the **Service Provisioning** (SP), which proposes the end user a friendly web interface letting them discover the context, create knowledge and access their daily information. This section is built on a Service Oriented Approach (SOA) and it allows the user to interact with the Server without requiring any expert background, e.g., SP offers numerous services to end-users such as, a service for the automatic discovery of UbiGATE; or another service to enter specific information, such as: personal details, environment description, and sensors and device location. The information obtained through those services populates *instance* model.

4. KNOWLEDGE DISCOVERY

The Service Provisioning is in charge of the knowledge management. It is through services proposed by SP that the system and end-users manage (create and process) the context. Each service is based on the same prototype, which is publicly available. It allows the user to implement any routine in JavaScript and to provide the HTML template for the data collection through a browser. The interaction of services with the Server is performed through the dedicated API to avoid the access to sensible and personal data, such as: passwords, addresses or functional code.

For the service of creation of user-profile and his environment, SP provides a simple form view that creates a user profile and second one that creates an environment linking it to the user. The creation of an environment triggers the generation of a new *instance* model, and the inclusion of this information in the KB is automatically done in the background (view code sample 1).

As for the UbiGATE discovery service, the software embedded in the gateway is customized and optimized. Thus, when the gateway is plugged for the first time to the electricity and internet, it tries to connect to a specific public API of the Server through HTTPS protocol. This service listens to these requests and shows the information in real time. End-user is then required to accept the connection and to pair the gateway with a specific environment. Once the gateway is included in the reasoning, UbiSMART creates the MQTT credentials for further communications automatically. Figure 2 illustrates the information exchange between the UbiGATE and the Server.

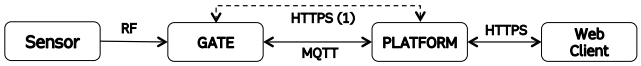


Figure 2: Communication protocol between UbiGATE and the Server during the space discovery.

The Server is coded in JavaScript using NodeJS. It uses MQTT Publish and Subscribe communication protocol as a simple and scalable way to deal with data coming from different spaces at the same time. As presented in [5], UbiSMART uses Mosquitto as MQTT broker because of its compatibility with JavaScript. The reasoning is a Knowledge driven approach. It can be easily personalized to each user and it is adapted for complex systems. UbiSMART framework uses Jos de Roo's Eye reasoning engine[2] and in its server-side it uses a JavaScript node wrapper made available by Ruben Verborgh. It also uses Ruben Verborgh's TripleStore for N3 in JavaScript[3].

```
// User creation
kb.store.addTriple(house.id,'hom:'"'+user.name'"','
    ↪ rdf:type','"'+house.id+'"^^xsd:integer');
kb.store.addTriple(house.id,'hom:'"'+user.name'"','
    ↪ rdf:type','qol:Resident');
kb.store.addTriple(house.id,'hom:'"'+user.name'"',
    ↪ qol:residentIn,'hom:'"'+house.id'");
// House creation and link with the user profile
kb.store.addTriple(house.id,'hom:house','qol:name
    ↪ ','"'+house.name+'"^^xsd:string');
kb.store.addTriple(house.id,'hom:house','qol:id
    ↪ ','"'+house.id+'"^^xsd:integer');
// Inclusion of the triples in the Knowledge Base
kb.store.addTriples(houseId, triples);
```

Code 1: Background routine creating the user profile, the environment and the inclusion in the *instance* model.

The context discovery service completes the *instance* model with the necessary information to run the inference engine. First, it gathers personal information and environment distribution: resident, caregiver, the number of pieces, connected objects. The proposed pieces and objects are loaded from the *abstract* model to keep a consistency in the declaration of variables. Second, the sensor discovery is automatic. Once a sensor is within the range of detection of UbiGATE, they can be eligible for this space. Then, an end-user can approve the sensors he wants to use one by one. Finally, the service lists the devices linked to this user and environment to be approved and placed. E.g., sensor *A2* is a *motionSensor* placed in the *Toilet* or tablet *D1* placed in the *Bedroom*. Figure 3 illustrates this dedicated service to populate the *instance model* for each environment. The code sample 2 represents the automatic generation of the triples which will be integrated into the *instance model*.

[2]http://eulersharp.sourceforge.net/
[3]https://github.com/RubenVerborgh/node-n3

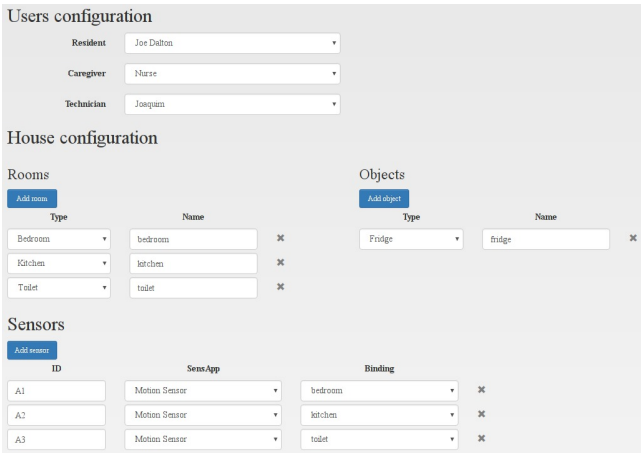


Figure 3: Smart home configuration service.

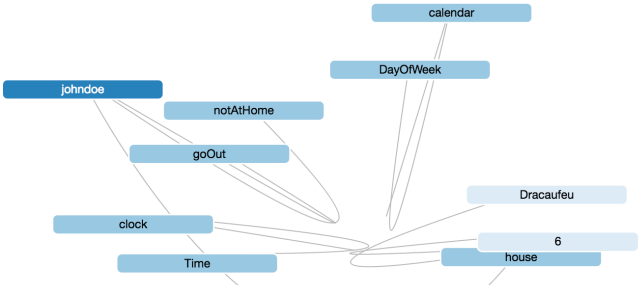
```
// generate triples for rooms
_.each(pN3.rooms, function(room) {
triples.push(['hom:'+room.name, 'rdf:type', 'qol
    ↪ :'+room.type]);
triples.push(['hom:'+room.name, 'qol:partOf', 'hom
    ↪ :house']);
});
// generate triples for sensors
_.each(pN3.sensors, function(sensor) {
var sensorUri = 'hom:'+sensor.id.toLowerCase();
triples.push([sensorUri, 'rdf:type', 'hom:'+sensor
    ↪ .sensapp]);
triples.push([sensorUri, 'qol:id', '"'+sensor.id
    ↪ +'"^^xsd:string']);
var bindingPredicate = (sensor.bindingType === '
    ↪ Room') ? 'deployedIn' : 'attachedTo';
triples.push([sensorUri, 'qol:'+bindingPredicate,
    ↪ 'hom:'+sensor.binding]);
});
//Load triples into the KB
kb.store.addTriples(houseId, triples);
```

Code 2: Background routine populating the *instance* model from the information entered into the home description service.

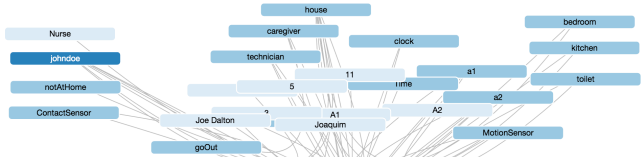
5. VALIDATION AND RESULTS

The validation UbiSMART was divided into two parts: controlled and real scenarios. First, the technical validation and proof of concept in controlled scenarios. Over two weeks, we involved researchers from the lab to deploy UbiSMART without any prior knowledge. The experiences were performed within a living lab facility. The volunteers received a close box with UbiGATE and the credentials to the web app. The aim of this experiment was to detect the blocking process and improve the user experience and web design.

Second, we deployed our system in more than 40 real and distant deployments. We counted on the collaboration of Nursing Homes and Senior Activity Centers to recruit real participants. The system was delivered in a box and we tested the connectivity with the server from different locations and connectivity (LAN or 3G). In both parts, simulated and real, the knowledge creation works correctly most of the times. The system detected the UbiGATE and the sensors discovery worked as expected. The background routines created the triples correctly and the inclusion in the KB was satisfactory. Moreover, table 1 shows the improvements in terms of time investment when using the novel approach. These improvements are compared to our experience when



(a) Knowledge Base status after the creation of user profile and environment.



(b) Knowledge Base status when the system is ready to run the inference engine.

Figure 4: Ontology evolution. The specific values are instantiated based on the *abstract* model to populate the *instance* model. E.g., sensor *a2* is the instance of the *motionSensor* type *A2*.

deploying the old version of UbiSMART. The preparation work, where the technician created the user, profile and the user environment, has been reduced from one (1) hour to five (5) minutes. The deployment time has notably dropped from four (4) hours to 20 minutes because of the use of *abstract* model and the friendly completion of the *instance* model. The installation does not need the presence of an expert anymore and the upgrades can be tested in controlled scenarios and performed remotely. Figure 4 represents the ontology before and after using our services for the knowledge construction.

	New UbiSMART	Old UbiSMART
Preparation Work	5 min	1h
Deployment Time	20 min	4h
Installer	Non-Expert	Expert
Upgrade Process	0 min	Up to 20 min

Table 1: Improvements in terms of time using the new version of UbiSMART. In the table we compare the current time spans with the time spans we experienced in our old deployments.

6. CONCLUSIONS

We introduce in this paper the use of an agile web framework to discover context and construct the semantic Knowledge AAL environments. The final goal of the framework is to enable detecting Activities of Daily Living and provisioning adaptable services for aging people.

We present, first our new approach for AAL cloud-based web App. This new approach allows populating the Knowledge Base of the semantic reasoning through user-friendly services. Second, the models we use to manage many environments at the same time. The *abstract* model, which contains the common properties and the *instance* model, which considers the peculiarities of each environment. Third, the technical implementation of the services presented to the end-user to gather the knowledge. Finally, we present the benefits of our new framework in terms of time and human resources involvement.

The results show that using UbiSMART allows us to move forward towards a quick large and remote deployment. The web platform provides a simple way to turn any space into a smart environment in five (5) minutes and the provided services notably help its installation. UbiSMART is particularly easy to setup because of the automation of most of the process and the friendly interaction. This new approach has been tested in controlled environments to prove the concept and then move forward to real and distant deployments. We can prepare and deploy the system in less than 20 minutes per space without technical experts involved. This approach has allowed us to collect more than 462,284 sensor events and inferred 84,351 activities over a period of 386 days. Currently, we are working to deploy more spaces.

7. REFERENCES

[1] B. Abdulrazak and A. Helal. Enabling a plug-and-play integration of smart environments. In *2006 2nd International Conference on Information & Communication Technologies*, volume 1, pages 820–825. IEEE, 2006.

[2] H. Aloulou, B. Abdulrazak, R. Endelin, J. Bentes, T. Tiberghien, and J. Bellmunt. *Simplifying Installation and Maintenance of Ambient Intelligent Solutions Toward Large Scale Deployment*, pages 121–132. Springer International Publishing, Cham, 2016.

[3] H. Aloulou, M. Mokhtari, T. Tiberghien, J. Biswas, and L. J. H. Kenneth. A semantic plug&play based framework for ambient assisted living. In *Impact Analysis of Solutions for Chronic Disease Prevention and Management*, pages 165–172. Springer, 2012.

[4] H. Aloulou, M. Mokhtari, T. Tiberghien, R. Endelin, and J. Biswas. Uncertainty handling in semantic reasoning for accurate context understanding. *Knowledge-Based Systems*, 2015.

[5] J. Bellmunt, T. Tiberghien, M. Mokhtari, H. Aloulou, and R. Endelin. Technical challenges towards an aal large scale deployment. In *Inclusive Smart Cities and e-Health*, pages 3–14. Springer, 2015.

[6] A. Helal, M. Mokhtari, and B. Abdulrazak. *The engineering handbook of smart technology for aging, disability and independence*. John Wiley & Sons, 2008.

[7] Y. Hu, D. Tilke, T. Adams, A. S. Crandall, D. J. Cook, and M. Schmitter-Edgecombe. Smart home in a box: usability study for a large scale self-installation of smart home technologies. *Journal of Reliable Intelligent Environments*, 2(2):93–106, 2016.

[8] T. Tiberghien, M. Mokhtari, H. Aloulou, and J. Biswas. Semantic reasoning in context-aware assistive environments to support ageing with dementia. In *The Semantic Web–ISWC 2012*, pages 212–227. Springer, 2012.

[9] J.-b. Woo and Y.-k. Lim. User experience in do-it-yourself-style smart homes. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 779–790. ACM, 2015.