Semantic Context-Aware Service Composition for Building Automation System

Son N. Han, Gyu Myoung Lee, *Senior Member, IEEE*, and Noel Crespi, *Senior Member, IEEE*

***Abstract*—Service-oriented architecture (SOA) is realized by independent, standardized, and self-describing units known as services. This architecture has been widely used and verified for automatic, dynamic, and self-configuring distributed systems such as in building automation. This paper presents a building automation system adopting SOA paradigm with devices implemented by device profile for web service (DPWS) in which context information is collected, processed, and sent to a composition engine to coordinate appropriate devices/services based on the context, composition plan, and predefined policy rules. A six-phased composition process is proposed to carry out the task. In addition, two other components are designed to support the composition process: building ontology as a schema for representing semantic data and composition plan description language to describe context-based composite services in form of composition plans. A prototype consisting of a DPWSim simulator and SamBAS is developed to illustrate and test the proposed idea. Comparison analysis and experimental results imply the feasibility and scalability of the system.**

***Index Terms*—Building automation, device profile for web service (DPWS), semantic web, service composition, service-oriented architecture (SOA).**

I. Introduction

THE idea of a smart house or a smart building has been around and well-known for years as highly expected products. A building automation system (BAS) residing at the heart of such smart environments interacts with all of its components, with hardware, software, and communication among them. It involves in several disciplines such as electronics, informatics, automation, or control engineering. BAS, since its debut, has been developed and promoted by a community of developers, technologists, and scientists with plenty of impressive prototypes and products. These products bring in comforts and conveniences to daily life, freeing people from tedious house-works or office-works. Use cases vary from very simple ones, e.g., automatically turning on/off the lights to complex and critical situations, e.g., security surveillance. Furthermore, BAS also provides value-added services by offering intelligent services such as customer tracking in shopping malls or elder healthcare services. All of these options make it a very promising business, attracting the attention of the community to target not only organizational customers but individual end-users.

With such analysis in mind, savvy people in the industry and academia have been developing many new technologies for building automation such as communication protocols, data management, data bus systems, software components, and/or new hardware devices which can be integrated in the new systems. Thanks to all of these efforts, building automation has advanced over the last few decades with several communication protocols and a variety of BAS products from many different vendors. A comprehensive overview of communication protocols in building automation can be found in [1], which also introduces different BAS products and other discussions on building automation. Traditionally, equipment in BAS products are interconnected by proprietary communication protocols such as LonWorks [2], Building Automation and Control Net-work (BACnet) [3], or KNX [4]. These protocols have been used to cover all of the features of building automation, including heating, ventilating and air conditioning (HVAC), lighting, and alarming.

The industry also has been making significant efforts to standardize communication protocols. KNX, for example, is one of the main communication standards used in building automation. It is device-independent and allows monitor/control of lighting, blinds/shutters, security systems, energy management, and HVAC systems. HomeConnex (Peracom Networks) is a home entertainment network which unites PCs, TVs, audio/video components, and set-top devices into an integrated system. X-10 (X10 Inc.) is another industry standard using power lines and radio for communication among electronic devices used for home automation. Other proprietary standards include Easy-Radio (Low Power Radio Solution Inc.), No New Wires (Intellon Corporation), Sharewave (Sharewave Inc.), SoapBox (VTT Electronics), and Z-wave (Zensys).

So far, consumers are overwhelmed by the many appealing BAS solutions offered by the industry and are well aware of the value of such smart systems. However, it is not difficult to recognize the reluctance among customers in adopting available BAS products on the market. The main reasons are identified as the cost and the scalability of these proprietary systems. This normally leads to the suspension or partial deployment of several on-the-table building renovation projects.

In recent years, there has been a recognizable movement from distributed systems controlled by users to a new style of systems built over independent, autonomous, and standardized components or services. This new style of systems called service-oriented architecture (SOA) enables organizations to share logic and data among multiple applications and usage modes. SOA is

Manuscript received November 02, 2012; revised January 08, 2013; accepted February 21, 2013. Date of publication March 12, 2013; date of current version December 12, 2013. Paper no. TII-12-0755.

The authors are with the Department of Wireless Networks and Multimedia Services, Telecom SudParis, Institut Mines-Telecom, Evry, 91011 France (e-mail: son.han@it-sudparis.eu; gm.lee@it-sudparis.eu (corresponding author); noel.crespi@it-sudparis.eu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2013.2252356

also an open concept supporting plug-and-play capabilities of heterogeneous software and hardware components. W3C Web Services [5] using simple object access protocol (SOAP) message [6] and Web Services Description Language (WSDL) [7] is probably the most popular implementation of SOA which is gaining increasing market penetration.

As will be discussed in Section II, there have been many approaches as well as a great deal of research on adopting SOA paradigm into BAS. Several technologies and standards also have been created to support the trend. These works mainly focus on how to deal with the heterogeneity of the devices as well as the scalability of managing and adding devices from different third-party providers in smart building environment. In terms of the functionality at the application layer, there have been only simple prototypes rather than a full-scale system for SOA-based building automation. Current solutions also appear static or semidynamic as they only take into account simple use cases and scenarios in context change, e.g., in temperature or humidity. These efforts can be categorized under the name of enabling technologies for integration of SOA into BAS. However, to successfully enable such integration, it requires more than just connectivity and interoperability of heterogeneous systems.

Therefore, this paper aims to pave the way for the development of full-scale SOA-based BASs by the support of open Web technologies and device profile for web service (DPWS) [8]. There are two problems challenging the development of this kind of BASs. The first problem is to coordinate devices in order to serve the diverse and complex requirements consisting of several services from users. To solve it, the concept of service composition is exploited to compose services based on predefined policy rules with reference to building ontology to choose, bind, and execute appropriate services. The second problem is to deal with the dynamic changes in context of users and building environment, and the solution is to use composite service plans to describe users' requirements by a proposed Composition Plan Description Language (CPDL). User and environment context are modeled and processed efficiently in the system through the context processor to help the decision making process to carry out the service composition.

To put all together, an SOA-based BAS built over DPWS-supported devices is proposed. DPWS devices cover a large range of equipment from highly resource-constrained sensors, full TCP/IP stack devices, to new-trend Android devices, thanks to dedicated open-source DPWS implementations. The database of resource description framework (RDF) [9] graphs is represented semantically by one of the textual syntax alternatives to RDF called Notation3 (N3) [10]. Building ontology containing the description of concepts and relationships in the building environment is designed and used as a reference schema for storing graph data in the database. Context information is modeled, processed, and passed to a service composition engine to coordinate appropriate devices/services based on predefined policy rules and six-phased composition process. The remainder of this paper is organized as follows. Section II provides background and related work on applying SOA paradigm and open Web technologies to building automation. Section III describes architecture of the system. Section IV introduces the design of building ontology and graph database. Semantic context-aware service composition is presented in Section V. Section VI brings in comparison analysis with state-of-the-art and a prototype of the system along with experimental results. Section VII concludes the presentation and discusses future works.

TABLE I
DPWS IMPLEMENTATION

Version	Language	Target	Operating System
DPWS-gSOAP	C	Embedded system	Linux, Windows, Embedded Linux
DPWS-uDPWS	C	Sensor/Actuator	Contiki
DPWS-JMEDS	Java	Embedded system	Embedded Linux
DPWS-Android (JMEDS)	Java	Embedded system	Embedded Linux

II. RELATED WORK

There have been several SOA models in the industry targeting device-to-device communication with technologies as old as Home Audio/Video Interoperability (HAVi) [11] and Java Intelligent Network Infrastructure (Jini) [12] to new and currently developed such as Open Service Gateway Initiative (OSGi) [13], DPWS [8], and/or Universal Plug and Play (UPnP) [14], in which DPWS is getting popular among developers because of its reliance to W3C Web Services. DPWS was debuted in 2004 by a consortium led by Microsoft and natively integrated in Microsoft's Windows Vista platform. It defines a set of implementation constraints to provide secure and effective mechanism for describing, discovering, messaging, and eventing of services for resource-constrained devices. An open-source implementation of DPWS was also developed under several projects funded by Information Technology for European Advancement (ITEA) including SIRENA, SODA, and SOCRATES. Research results of the SIRENA project [15] have been presented widely by industry and academia. It is also available through an open-source software initiative Web Service for Devices (WS4D) [16] which is currently being actively maintained. WS4D has been demonstrated as feasible and powerful for several domains including industrial automation. Up to date, several DPWS stack implementations have been provided by the WS4D initiative in various platforms and languages.

Four core implementations are summarized in Table I. DPWS-gSOAP provides C/C++ toolkits for deploying Web services consumers and providers. It is multiplatform implementation supporting Linux i386, Windows-native, Windows-cygwin, and embedded Linux. DPWS-uDPWS is DPWS implementation in C language designed for Linux systems and especially for highly resource-constrained devices such as sensor nodes. It paves the way for DPWS protocols to be implemented directly into a wireless sensor network (WSN) without any intermediary gateway or so. DPWS-JMEDS is Java framework for DPWS supporting different Java editions. The latest release of DPWS-JMEDS hosts the feature of Android OS which paves the way for implementing services on Android devices.

In addition, academia has been also actively supporting the SOA paradigm as well as DPWS in building automation with a great deal of research and prototypes. Many other solutions for implementing the SOA on networked embedded devices

have been proposed to prove the feasibility of the paradigm regardless of the variety of devices and the heterogeneity of the network with different communication protocols like ZigBee, Bluetooth, or TCP/IP. Distributed operating system (DOS) [17] presents an operating system based on SOA to manage all embedded devices in a home network. It sets to solve four problems regarding the connectivity between devices and central managing point: multiple simultaneous connections to a device, request packet redundancy, inflexibility of direct access to embedded devices, and compact-SOA message. Leong *et al.* [18] try to solve the interoperability problem concerning message exchange between two or more subsystems and performing interoperation without the need for external intervention. The solution involves not only the integration of components in subsystems but also their behavior. It is a rule-based framework with event-condition-action (ECA) pattern inherited from expert systems domain for representing, sharing, and managing data in a smart home environment.

Kyusakov *et al.* [19] introduce an improved DPWS architecture which considers a particular requirements of WSN. Furthermore, for the exchange of SOAP messages, the work used the IETF Constrained Application Protocol (CoAP). The work in [20] discusses the problem of using Web services over IPv6 over low-power wireless personal area networks (6LoWPAN) in WSN. Also, in [21], a modified DPWS protocol stack is proposed to be used in WSNs which comply with 6LoWPAN. They all show that it is applicable for implementing DPWS and SOAP for WSN without any change of existing solution on conventional resource-rich devices.

In the meantime, researches on Web service and open future Web technologies such as Semantic Web have been blooming up in the last decade, making them the most supported and promising technologies. Derived technologies such as service discovery and service composition have been also extensively exploited. Initially, service composition is used in the business processes to manually, semi-automatically, or automatically mobilize component services to create composite services which can satisfy various complex business requirements. The idea of service composition then has been adopted widely into the domain of ubiquitous computing or pervasive computing, especially with respect to context, in which dynamic user and environment context should be taken into account to ensure the successful deployment of the system. The work in [22] proposes a framework for context-aware dynamic service composition in ubiquitous environment. It argues that changes in ubiquitous ambient environments occur frequently in different situations, and the system therefore needs to adapt dynamically. The work in [23] presents a prototype with UPnP devices and OSGi gateway to apply the concept of service composition for building automation. The concept has also penetrated other fields of industrial automation such as the semantic composition model presented in [24] aims to managing production processes in factory automation. The model coordinates three Web services to achieve production goals using the domain Web services.

Semantic Web as envisioned by Berners-Lee *et al.* [25] is another promising technology for building automation. Though it is not yet a full-fledged technology, it does offer a new concept of presenting data in a meaningful way which can improve the communication between human and machine. This means that a certain level of automation can be achieved through Semantic Web. Consequently, Semantic Web is very promising for industrial automation. Runde and Fay [26] from the point of view of software engineering in building automation introduce only a partial adoption of Semantic Web technologies but still hint at many open issues for future work. Another long-term effort for the adoption of Semantic Web into building automation is an ongoing SESAME-S project [27]. The solution concerns the efficient energy consumption in the house which is equipped with sensors, smart meters, and a semantic software module to perform reasoning and control the house energy consumption. Key achievements and technologies of SESAME-S project are described in [28].

To summarize, with several standards, prototypes, as well as products proposed by industry and academia, applying SOA and open Web technologies to building automation has been shown to be possible and promising. The above review also provides an outlook over the readiness of the state of the art of enabling technologies for the development of real BAS based on SOA and Web technologies. Even though several systems have been developed in this way or another, there's still an absence of a full-scale SOA-based BAS products offering some intelligence to cover all the aspects of real and complicated scenarios. Next, the proposed system in such a manner is going to be presented and thoroughly explained.

III. System Architecture

Here, the key functionalities of the system are presented with its architecture in the background. Main explanations include DPWS Service Discovery followed up here. Building ontology and graph database will be described in Section IV, and semantic context-aware service composition will be discussed in Section V. During discussion, devices and theirs hosting services are going to be used interchangeable and sometimes mentioned as devices/services.

The system configuration shown in Fig. 1 depicts the main components and a typical setup of equipments inside a room of a building. There are four groups of devices which are all DPWS-supported consisting of a wide range of building appliances including sensors/actuators, radio-frequency identification (RFID) readers, other appliances with TCP/IP stack and low-power wireless protocols, and Android devices. The first group is sensors/actuators, which are attached to devices to provide networked functionalities. They are implemented by uDPWS over the Contiki OS [29]. The second group is DPWS devices which consists of devices natively support DPWS with full TCP/IP stack. These devices are developed using DPWS-gSOAP and can be connected directly to the IP network. The third group is newborn Android devices which are attracting much attention recently. In 2011, some big electronics companies like Panasonic or Archos released home appliances based on an Android operating system. Google itself introduced the Android@Home at the 2011 Google annual I/O developer conference with the intention of turning the home into a network of Android accessories. Along with that event, Google also announced that it had collaborated with a partner

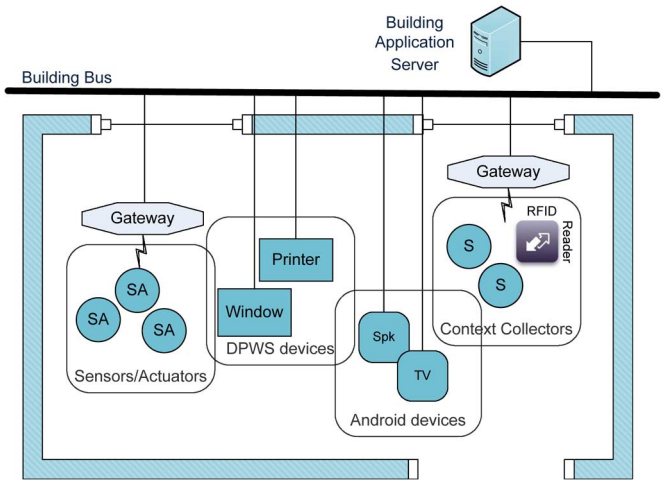


Fig. 1. System configuration. Typical four groups of devices which consist of sensors/actuators, DPWS devices, Android devices, and context collectors are deployed at the same time in a room of a building.

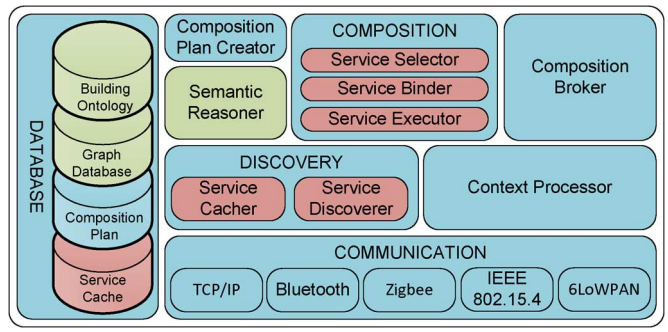


Fig. 2. System architecture consists of four main subsystems DATABASE, COMMUNICATION, DISCOVERY and COMPOSITION and four other modules: composition plan creator, semantic reasoner, composition broker, and context processor.

to launch Android-enabled LED light bulbs. Though there has been some delay in launching the aforementioned Android home appliances, with the constant development of Android platform these days, Android-based home and building devices and appliances are very promising products in the near future. The fourth and last group is classified as context collectors consisting of sensors to provide sensing capacities and RFID readers to receive users' identifiers. All hardware components are connected to Building Application Server (BApS) directly or indirectly via gateways to expose their services. BApS hosts the core functionalities of the system with the details to be discussed in the following sections.

The system architecture shown in Fig. 2 consists of several subsystems and modules. The first subsystem DATABASE is composed of a building ontology, semantic graph database, another database for composition plan, and a caching component of service cache to improve the operation of the service execution process. Building ontology provides the description of concepts and their relationships in building environment. The second subsystem is COMMUNICATION, which represents functionalities over heterogeneous communication methods including TCP/IP and several other low-power wireless protocols. There are different types of hardware devices ranging from sensor nodes, sensor/actuator nodes, DPWS devices, Android devices, and RFID readers which are connected to the network by different types of communication protocols. Note that there are two types of sensors: one attached to device to provide sensing functionality and the other used for collecting context information such as temperature or moisture level in building environment. The third subsystem DISCOVERY consists of two modules service discoverer and service cacher, but they work closely with each other under the DPWS WS-Discovery specification. Service discoverer plays the role of an interface between the core of BAS and devices. It sends a request to the network to discover connected devices and accompanied services and then receives information of available devices/services. Service cacher runs frequently to update the service cache and carries out the update process whenever service discoverer is in operation.

COMPOSITION, the fourth and the central subsystem resides at the center of the architecture. Its six-phased composition process helps to realize and deliver appropriate composite services to user based on the context of user and environment. The subsystem can be functionally divided into selecting services, binding services, and executing services which are reflected in three components of the COMPOSITION: Service Selector, Service Binder, and Service Executor, respectively.

Among the other modules, Semantic Reasoner is also an important part, as it plays a link between building ontology, graph database, and the software component of COMPOSITION. This module is described subsequently along with the building ontology and graph database in Section IV. Composition Plan Creator has access to composition plan database and provides functionalities for users to create, modify, and delete composition plans. The wrapping interface over this module can be a part of Web or smartphone applications providing typical graphical buttons or comboboxes for users to interact with the system. Context processor receives raw context data from context collectors, removes headers and redundant packets to extract important information, and then sends them to the composition broker in form of structured data. Composition broker decides whether to call the COMPOSITION or not with a simple decision-making mechanism based on the received context information.

DPWS service discovery as the main content of DISCOVERY subsystem is going to be first introduced to explain the communication between devices and the core of the system. Then, context-awareness in the context of BAS is also explained later on in this section to give the details of context data and how to process them for the composition process.

A. DPWS Service Discovery

Different types of hardware components built over DPWS specification share the same dynamic mechanism of exposing and discovering in the network which is called DPWS-Discovery as part of DPWS specification. Therefore, the service discoverer module can universally detect and communicate with each service and all of their operators regardless of the origin of the hosting device, hardware, or operating system.

Fig. 3 shows the process of discovering devices. DPWS devices send "Hello" and "Bye" message to join and leave the net-

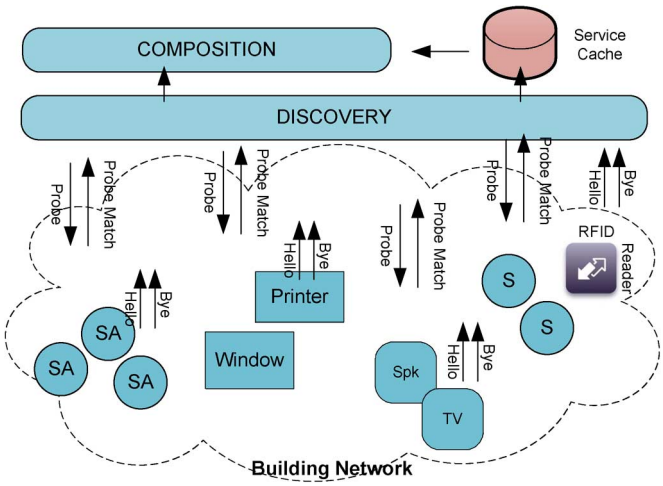


Fig. 3. Service discovery. Devices in different types send hello/bye messages to the network to join/leave the network. A client sends a probe message to request for the service of a specific device and a matching service returns probe match message to confirm its existence.

```
SearchParameter search = new SearchParameter();
search.setDeviceTypes(new QNameSet(
    new QName(''ExamplePrinterDevice'', namespace)));
SearchManager.searchDevice(null, client, null);
```

Fig. 4. Service search.

```
Service Type: {http://www.it−sudparis.eu}Light803Services
Endpoint Reference Address: http://127.0.0.1:5678/Light803Service/
WSDL: http://127.0.0.1:5678/Light803Service/ws4d/resources/description.
      wsdl
```

Fig. 5. Service cache.

work. When a DPWS client wants a DPWS device, e.g., with the identifier as a printer *ns:ExamplePrinterDevice*, it sends a *Probe:ExamplePrinterDevice* message to know if one is connected on the network. A DPWS printer receives this probe and answers back information that it is a printer (by sending a *ProbeMatch*). Other DPWS devices also receive this probe but they do not answer (as not *ExamplePrinterDevice printer*). Fig. 4 shows typical Java codes used to search for the device with information described in *SearchParameter*. The DPWS device speaks directly to the printer about its services metadata and the printer sends its metadata back. In case of discovering all of the devices available in the network, the client puts *null* in the device identifier parameter to get a set of connected devices. A callback function or a handler is called when there is a matching service found.

Information about each service associated with each device is collected through the service discovery process and is frequently updated to a data structure called service cache. Fig. 5 shows an example of a light service in the room 803 stored in the service cache with three pieces of information *Service Type*, *Endpoint Reference Address*, and *WSDL*.

B. Context-Awareness

Context-awareness plays an important role in the pervasive computing architectures to enable the automatic modification of the system behavior according to the current situation with minimal human intervention. Since appeared in [30], context has become a powerful and longstanding concept in human–machine interaction. As human beings, we can more efficiently interact with each other by fully understanding the context in which the interactions take place. It is difficult to enable a machine to understand and use the context of human beings. Therefore the concept of context-awareness becomes critical and is generally defined by those working in ubiquitous/pervasive computing, where it is a key to the effort of bringing computation into daily lives. One major task in context-aware computing is to acquire and utilize information about the context of participating entities of a system in order to provide the most adequate services. The service should be appropriate to the particular person, place, time, event, etc. where it is required. In the scope of this building automation, user, device and environment context are considered in order to bring more efficient service composition.

Context information is collected by sensors and RFID readers which are classified as context collectors. There are two types of context including User (U) and Environment (E). The raw data are sent to and processed by the context broker to yield the structured data in the format of {*Type, Location, Source*}. We present two examples of context here.

1) {*U, Room803, Smith*}
2) {*E, Room803, TempSensor803*}

The processed data then are sent to the composition broker which plays the role as a composition decision maker. It decides whether to call the COMPOSITION or not. For example, if the context information of room temperature is over 10 °C, no composition will be carried out; otherwise, the composition broker checks the temperature with current status of the system to launch the COMPOSITION in case the situation is labeled as context change.

IV. Building Ontology and Graph Database

Building ontology defines concepts and relationships between entities within the building environment. It provides a schema to build up semantic database in the form of graph data. This is a new concept of database for Semantic Web which consumes RDF to present the domain knowledge. RDF is a common acronym within the semantic web community as it creates one of the basic building blocks for forming the Web of semantic data. A graph consists of resources related to other resources, with no single resource having any particular intrinsic importance over another. The RDF database includes of RDF statements or sometimes called an RDF triples. The term triple is used to describe the components of a statement with three constituent parts: the subject, predicate, and object of the statement.

The primary purpose of this ontology is to classify things in terms of semantics, or meaning and especially for describing policies used in composition process. A class in Web Ontology Language (OWL) [31] is a classification of individuals into groups which share common characteristics. If an individual is a member of a class, it tells a machine that it falls under the semantic classification given by the OWL class. This provides the meaning of the data that helps reasoning engine to draw inferred information from the database. Fig. 7 shows a part

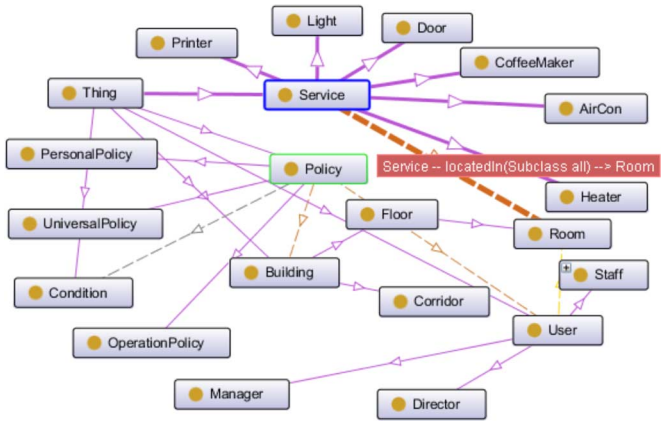


Fig. 6. Building ontology graph. The highlighted blocks in the graph show the hierarchy among class *Service* and its subclasses. The dotted line with a label presents a property called *locatedIn* which takes class *Room* as object meaning a service is located in a room.

```
<rdf:RDF xmlns=''http://www.it−sudparis.eu/bas_ont#''
    xml:base=''http://www.it−sudparis.eu/bas_ont''
    xmlns:rdfs=''http://www.w3.org/2000/01/rdf−schema#''
    xmlns:owl=''http://www.w3.org/2002/07/owl#''
    xmlns:xsd=''http://www.w3.org/2001/XMLSchema#''
    xmlns:rdf=''http://www.w3.org/1999/02/22−rdf−syntax−ns#''
    xmlns:xml=''http://www.w3.org/XML/1998/namespace''>
  <owl:Ontology rdf:about=''http://www.it−sudparis.eu/bas''/>

<!−− http://www.it−sudparis.eu/bas_ont#Policy −−>
<owl:Class rdf:about='' http://www.it−sudparis.eu/bas Policy''>
      <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource=''applyFor''/>
          <owl:someValuesFrom rdf:resource=''Building''/>
        </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource=''applyFor''/>
          <owl:someValuesFrom rdf:resource=''User''/>
        </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource=''hasCondition''/>
          <owl:someValuesFrom rdf:resource=''Condition''/>
        </owl:Restriction>
      </rdfs:subClassOf>
</owl:Class>
...
```

Fig. 7. Building ontology document.

of Building Ontology document in OWL by Protégé-OWL editor [32]. The listing consists of document header and the declaration of the class *Policy* with two properties of *applyFor* and *hasCondition*. These properties also reflex the relationship of class *Policy* with other classes including *Building*, *User*, and *Condition*. Fig. 4 shows the classes of Building Ontology and their hierarchical relationship. An example of the hierarchy between classes of *User* and *Director* can be seen in the figure with the arrow starting from *User* pointing to *Director* which means *Director* is a subclass of *User* and inherits all the properties of *User*.

```
@prefix : <http://www.it−sudparis.eu/bas_data#> .
@prefix bdg: <http://www.it−sudparis.eu/bas_ont#> .

UniversalHeatingPolicy
    a bdg:OperationPolicy ;
    bdg:applyFor bdg:User ;
    bdg:hasCondition :HeatingCondition .
:HeatingCondition
    a bdg:Condition ;
    bdg:conditionType ''Heating'' ;
    bdg:conditionValue 10 .
...
```

Fig. 8. *HeatingCondition* Rule Data.

The above building ontology acts as a schema to define the data among the domain of building automation semantically which is part of the inferring processes. The data are presented in N3 format, a non-XML serialization of RDF. A piece of data is shown in Fig. 8 containing a policy called *UniversalHeatingPolicy* which is an instance of *OperationPolicy* (Building Ontology class). It applies for all users, instances of *User* (Building Ontology class) and has condition *HeatingCondition* (data). *HeatingCondition* is later on described as an instance of *Condition* (Building Ontology class) with "Heating" type and taking the value 10. Previously, two name spaces were defined at the header, one for the data and the other for the ontology.

This kind of graph database built around the building ontology enables Semantic Reasoner to infer additional information from existed data and relationship. A simple example of the reasoning from the data shown in Fig. 8 is explained as follows. In this case, a fact is stated as *UniversalHeatingPolicy* rule if it applies for instances of *User* class. A reasoner with basic capacity can be used to demonstrate the use case, e.g., Jena [33] natively supported reasoner. An inference model is created which takes the reasoner, building ontology, and the Graph Database as input parameters. Data in the form of resources and properties are then created from database. A simple code line can be used to generate an entailed relationship. Specifically, user *Smith* who is an instance of *Director* (Building Ontology class, subclass of class *User*) would be applied by the *UniversalHeatingPolicy* rule as well. This reasoning model helps to reduce the database size and quickly collect all related data of an event or user which are all necessary for the service composition process.

V. Semantic Context-Aware Service Composition

Residing at the heart of the proposed BAS, the COMPOSITION subsystem is in charge of answering composition requests from composition broker with regard to collected context information. It then gets access to all related resources to coordinate appropriate devices/services to serve the request. Previously, building ontology and Graph Database have been discussed to provide the semantic database. Also, context information processed by the context processor is passed to the composition process as the input data. In addition to that, a description language is designed to describe the composition plans and a six-phased composition process is proposed to efficiently and accurately carry out service composition.

```
<?xml version=''1.0'' encoding=''UTF−8'' ?>
<CSDL xmlns:xsi=http://it−sudparis.eu/bas>
        <context type=U location=Room803 source=Smith>
                <service>Window</service>
                <service>Light</service>
                <service>CoffeMaker</service>
        </context>
</CSDL>
<xml>
```

Fig. 9. CPDL.

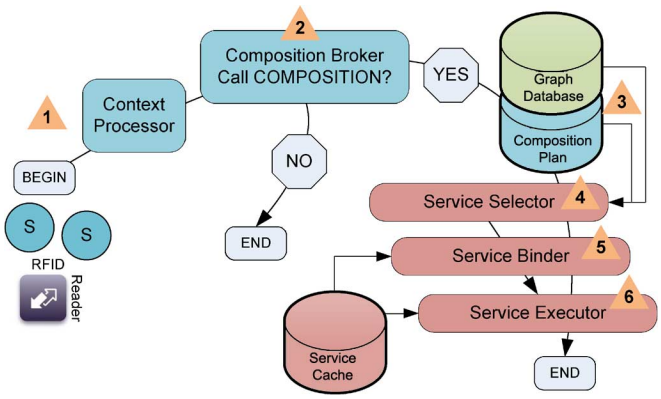


Fig. 10. Composition process. Six phases of the composition process are highlighted in the triangle signs.

A. CPDL

The language CPDL has been designed to describe composition plans associating with each context. An example of a CPDL document is shown in Fig. 9. This document describes a composition plan with the type of *U* or *User* and for user *Smith* with the context of his presence in the room 803. It also describes the composite service consisting of three component services *Window*, *Light* and *CoffeeMaker*.

B. Service Composition

Six-phased service composition process is shown in Fig. 10 which visually depicts six phases of the composition as follows.

- Phase 1: Collect and process context information.
- Phase 2: Make decision to call COMPOSITION.
- Phase 3: Query semantic data.
- Phase 4: Select services.
- Phase 5: Bind services to their operations.
- Phase 6: Execute operations of services

The process starts with signals from Context Collectors when they detect changes in context and send context information in the building environment to Context Processor. Context Processor processes this information to meaningful and machine readable data. These processed and well-structured contextual data are sent to the Composition Broker to decide whether to move on by calling the COMPOSITION or not. In case no action needs to be carried out, the system switches to the sleep mode, otherwise the COMPOSITION is called. Then, resources are collected in the database to support the composition process. Service Selector uses provided context information, CPDL data of the user at that context, and inferred policies from the Semantic Reasoner to select appropriate services and create

Algorithm: Context Matching

Input: Composition Plan, User
Output: Execution Plan
Algorithm:
Execution Plan EP = ∅
For each service AS in Composition Plan
    For each service S in Service Cache,
        If (S.Location = AS.Location)
            EP = EP ∪ S
Return EP

Fig. 11. Context matching algorithm. For each abstract service in the composition plan extracted from CPDL document associated with the User, if the service matches one of the services in Service Cache by location, it is added to the execution plan.

a concrete description of the required composite service. Service Binder follows up by binding with operations of selected services and Service Executor gets access to Service Cache to execute that operations. Fig. 11 explains one implemented algorithm for service matching based on the location context of the user.

VI. Comparison Analysis and Experiments

A. Comparison Analysis

To put the proposed system under the limelight with other close approaches, say, in SOA-based building automation, it offers to some extent intelligence with the use of Semantic Web and context-aware dynamic service composition. The intelligence inspired by Web technologies and the adoption of such open standards in all aspects of the system are important factors to differentiate the work with others. Table II highlights seven features of six systems and products in building automation related to SOA and Web technologies including the proposed. These features cover from what type of devices each system supports, what type of communication protocols among devices, to whether they apply following technologies: service composition, context-awareness, dynamic composition, Semantic Web, and reasoning. Rule-based framework in [18] is stated to deal with heterogeneity of the devices but in fact provide no detail on how it deals with particular types of devices. The framework neither features composition nor reasoning. Kaldeli *et al.* [23] design and develop a prototype of SOA-based home automation system which uses artificial intelligence (AI) planning service orchestration to response to dynamic users' contexts. This work spends much of its space to deal with context-awareness by the presentation and analysis of a comprehensive use case. However, there is no consideration for intelligence or reasoning. Another work [26], from the point of view of software engineering, improves requirements engineering of BAS by means of knowledge-based system and Semantic Web technologies. The prototype based on LonWorks devices and protocols presents no trace of intelligence or automatically controlling the building. In order to improve existing domotic standards to achieve some intelligence for new building automation system, Ruta *et al.* in [34] introduced enhancements of KNX standard to support knowledge-based and

TABLE II
SOA-BASED BAS SOLUTIONS COMPARISON

Solution	Device Type	Communication Protocol	Service Composition	Context-awareness	Dynamic Composition	Semantic Web	Intelligence Reasoning
Leong *et al.* [18]	Heterogeneous	SOAP	No	No	No	No	No
Kaldeli *et al.* [23]	UPnP	SOAP	Yes	Yes	Yes	No	No
Runde *et al.* [26]	LonWorks	proprietary	No	No	No	Yes	Yes
Ruta *et al.* [34]	KNX	proprietary	Yes	Yes	Yes	Yes	No
SEASAME-S [28]	Smart meters	proprietary	No	Yes	No	Yes	Yes
Proposed	DPWS	SOAP	Yes	Yes	Yes	Yes	Yes

context-aware functionalities in home and building automation. The proposed framework though addresses and solves some problems of service composition and context-awareness but due to being based on proprietary protocol, only limited achievements has been reached. SESAME-S project [28] focuses only on smart metering to assist consumers in making decisions and controlling of energy consumption. The communication protocol of smart meters is a proprietary European standard. The work features Semantic Web by using linked data and reasoning models.

To summarize, the proposed system in this paper not only covers all the missing points of other works, but also uses open Web standards in all of the aspects of the system to offer some intelligence to deal with the dynamic environment. In addition, the perspective of third-party companies to manufacture compatible devices and the establishment of new service providers are among important achievements of the proposed approach.

B. Prototype and Experiments

A system prototype was developed to illustrate the operation of the proposed system and to test the feasibility and scalability of the system. The prototype consists of two separate parts: DPWSim and SamBAS. DPWSim is a simulator of various DPWS devices with graphical animation to illustrate their operations. Since all the DPWS devices share the same mechanism of exposing and discovering over the network, without loss of generality, WS4D JMEDS stack version 5 (*ws4d-java-se-full-beta5.jar*) is chosen for installing functionalities of the simulated devices. A class named *BuildingDevice* extending JMEDS *DefaultDevice* class (*org.ws4d.java.service.DefaultDevice*) represents building devices. Services and operations of each device are implemented by inheriting *DefaultService* and *Operation* classes respectively. *DeviceAnimation* class wraps up visuals and animations of simulated devices. A graphical user interface on top of the devices representing an office plan along with its actors: office appliances in their places and a user who can move around the office space to change his context as shown in Fig. 12(a).

The SamBAS consists all of the system components discussed previously. Building ontology is developed using Protégé-OWL editor, Graph Database is represented in N3 format, and the COMPOSITION modules are developed in Java programming language on an application server with Intel processor 2.6-GHz, 6-GB RAM. It uses the Jena library for semantic data manipulation and Jena integrated reasoner for inference functionalities.

Fig. 12(b) illustrates a simple use case when a user Smith enters his office located in the room 803. When he arrives in the office in the morning, he uses his RFID name tag as a security badge to check on the RFID reader located on his office door. This RFID reader, functioning as a Context Collector, sends a context-change notification to the Composition Broker to check with associated policies whether to call up the COMPOSITION or not. In this scenario, it is YES. The system uses the reasoner to collect all the policies constrained to the user to create a concrete appropriate composite service based on the user's CPDL which in this case consists of *CoffeeMaker* and *Light*. Then, the two concrete context-based services *CoffeeMaker803* and *Light803* are selected and bound to their operations and finally executed by Service Executor to serve the user.

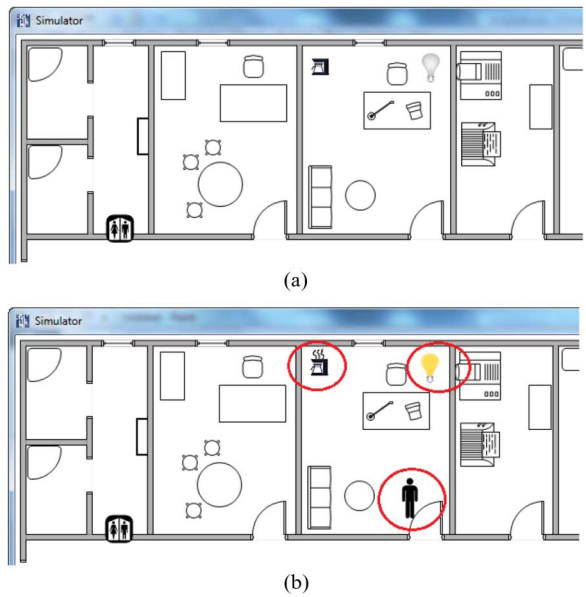


Fig. 12. DPWSim. (a) DPWSim demonstrates the service of user by the context. (b) The composite service consisting of two component services *Light* and *CoffeeMaker* is activated when the user is present in his office.

In order to evaluate the feasibility and scalability of the proposed system, two sets of experiments were carried out to verify the service selection and service execution processes. All experiments were run on the the SamBAS running on the server with Intel processor 2.6-GHz, 6-GB RAM. The first set of experiments aims to measure the time in milliseconds needed for selecting services as the number of devices increaes from 500 to 5000. Ten sets of experiments consisting of 10 runs by each were performed with results of consuming time recorded accordingly and put on a graph shown in Fig. 13. The figure shows the stability of the system with composition time kept under 2.5 s even in a critical situation with the participation of 5000 devices.

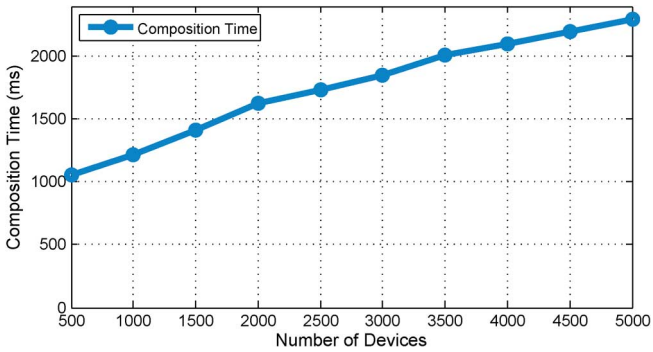


Fig. 13. Service selection. Composition time of service selection process as the number of devices increases from 500 to 5000.

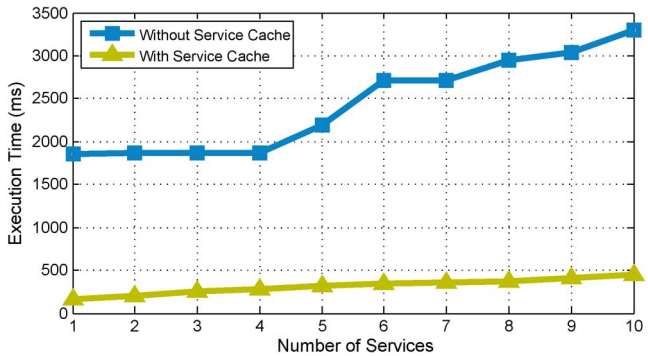


Fig. 14. Service execution. Service execution time in millisecond recorded in two situations, one with and one without Service Cache as the size of Composition Plan increases from 1 to 10.

The second set of experiments aims to compare the execution time of the Service Executor in two conditions, by using Service Cache and not using Service Cache. There is always a bit of mismatch between real services and services available in Service Cache due to the delay of the updating process. However, with an appropriate mechanism and the continuity of service discovery, the reliability of Service Cache is much improved to reach the state of real-time services. Fig. 14 shows the execution time in two situations as number of services in the Composition Plan increases from 1 to 10. Service Cache helps the system to improve the service execution process many times more than the system without it.

VII. Conclusion and Future Work

The design of the SOA-based BAS has been presented as a novel approach to bring Web technologies into building automation. Specifically Web service, SOA and Semantic Web have been integrated into BAS with Web service composition in the background to dynamically coordinate devices/services in accordance with the context. The results from experiments show that it is feasible to deploy the system even on a large scale.

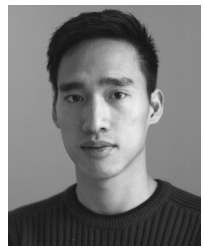
There are still a lot of issues left in the system which need attention in order to deal with further extension of the system such as concurrent multiple requests to devices. This happens when the system grows bigger with complex use cases whereas resource-constrained devices cannot by themselves deal with such simultaneous multiple requests. In the system, there exists a great deal of resource-constrained devices such as sensors or actuators which are only equipped with limited resources, e.g., about 15 kB ROM. This leads to the fact that one device at one time cannot process several requests from multiple users. It cannot be a problem in small systems with less complex scenarios. However, when system expands, the number of concurrent requests to devices also increases accordingly. Resource-constrained devices without enough capacity of handling concurrent requests could end up locked or dead, which subsequently results in the congestion of the system. The technique of multithreading such as dispatcher or scheduling algorithms can be adopted to develop a new module in the system to receive, process, and relay requests to appropriate devices at appropriate time.

Future work will be focused on the real deployment of the system in a physical environment with improvement in three issues. The first one is to reevaluate the system in the real scenario and improve its drawbacks. The second one is for system intelligence. Users' behaviors will be recorded and modeled to understand user's habit which can help to make the composition process smarter. The last issue is to integrate energy saving services into the current BAS to take advantage of the intelligence to even further improve the critical issue of saving energy.

References

[1] D. Dietrich, D. Bruckner, G. Zucker, and P. Palensky, "Communication and computation in buildings: A short introduction and overview," *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3577–3584, Nov. 2010.
[2] *Open Data Communication in Building Automation, Controls and Building Management—Control Network Protocol*, EN 14908-x (1-6), European Committee for Standardization, Brussels, Belgium, 2005–2010, .
[3] *Building Automation and Control Systems—Part 5: Data Communication Protocol*, ISO 16484-5, Int. Org. for Standardization, Geneva, Switzerland, Jul. 2012.
[4] *Information Technology—Home Electronic System (HES) Architecture—Part 4-1: Communication Layers—Application Layer for Network Enhanced Control Devices of HES Class 1*, ISO/IEC 14543-4-1, Int. Org. Standardization, Geneva, Switzerland, Jun. 2008.
[5] "Web Services Architecture," W3C, W3C Working Group Note, Feb. 2004 [Online]. Available: http://www.w3.org/TR/ws-arch/
[6] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, "Simple Object Access Protocol (SOAP) 1.1," W3C, W3C Note, May 2000 [Online]. Available: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
[7] Web Services Description Language (WSDL). ver. 2.0 Pt. 1: Core Language, W3C, Jun. 2007 [Online]. Available: http://www.w3.org/TR/wsdl20/
[8] Devices Profile for Web Services. ver. 1.1, OASIS, , Jul. 2009 [Online]. Available: http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01/
[9] "RDF Primer," W3C, Feb. 2004 [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
[10] T. Berners-Lee and D. Connolly, "Notation3 (N3): A Readable RDF Syntax," Mar. 2011 [Online]. Available: http://www.w3.org/TeamSubmission/n3/
[11] J. Teirikangas, HAVi: Home Audio Video Interoperability Helsinki University of Technology, Technical Report, 2001.
[12] Jini Architecture Specification. ver. 1.2, Sun Microsystems, Dec. 2001.
[13] OSGi Core Release 5 Specification. OSGi Alliance, 2012.
[14] *Information Technology—UPnP Device Architecture Version 1.1*, ISO/IEC 29341-1-1, Int. Org. for Standardization, Geneva, Switzerland, 2011.
[15] "ITEA SIRENA Project," [Online]. Available: http://www.sirena-itea.org/
[16] "Web Service for Devices Initiative," [Online]. Available: http://www.ws4d.org/

[17] A. Sleman and R. Moeller, "SOA distributed operating system for managing embedded devices in home and building automation," *IEEE Trans. Consum. Electron.*, vol. 57, no. 2, pp. 945–952, May 2011.
[18] C. Leong, A. Ramli, and T. Perumal, "A rule-based framework for heterogeneous subsystems management in smart home environment," *IEEE Trans. Consum. Electron.*, vol. 55, no. 3, pp. 1208–1213, Aug. 2009.
[19] R. Kyusakov, J. Eliasson, J. Delsing, J. van Deventer, and J. Gustafsson, "Integration of wireless sensor and actuator nodes with it infrastructure using service-oriented architecture," *IEEE Trans. Ind. Inf.*, vol. 9, no. 1, pp. 43–51, Feb. 2013.
[20] G. Moritz, F. Golatowski, D. Timmermann, and C. Lerche, "Beyond 6LoWPAN: Web services in wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. PP, no. 99, 2013.
[21] I. Samaras, G. Hassapis, and J. Gialelis, "A modified DPWS protocol stack for 6LoWPAN-based wireless sensor networks," *IEEE Trans. Ind. Inf.*, vol. 9, no. 1, pp. 209–217, Feb. 2013.
[22] K. Tari, Y. Amirat, A. Chibani, A. Yachir, and A. Mellouk, "Context-aware dynamic service composition in ubiquitous environment," in *Proc. IEEE Int. Conf. Commun.*, Cap Town, South Africa, May 2010, pp. 1–6.
[23] E. Kaldeli, E. Warriach, J. Bresser, A. Lazovik, and M. Aiello, "Interoperation, composition and simulation of services at home," in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer, 2010, vol. 6470, pp. 167–181.
[24] J. Puttonen, A. Lobov, and J. Martinez Lastra, "Semantics-based composition of factory automation processes encapsulated by web services," *IEEE Trans. Ind. Inf.*, vol. PP, no. 99, 2013.
[25] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific Amer.*, pp. 29–37, May 2001.
[26] S. Runde and A. Fay, "Software support for building automation requirements engineering: An application of semantic web technologies in automation," *IEEE Trans. Ind. Inf.*, vol. 7, no. 4, pp. 723–730, Nov. 2011.
[27] "SESAME-S Project," [Online]. Available: http://sesame-s.ftw.at/
[28] A. Fensel, S. Tomic, V. Kumar, M. Stefanovic, S. V. Aleshin, and D. O. Novikov, "SESAME-S: Semantic smart home system for energy efficiency," *Informatik Spektrum*, vol. 36, no. 1, pp. 46–57, 2013.
[29] "Contiki OS," [Online]. Available: http://www.contiki-os.org/
[30] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Proc. First Workshop Mobile Computing Syst. Applications*, Washington, DC, USA, 1994, pp. 85–90, ser. WMCSA '94.
[31] "OWL 2 web ontology language document overview," W3C, Oct. 2009 [Online]. Available: http://www.w3.org/TR/2009/REC-owl2-overview-20091027/
[32] "Protege-OWL Editor," [Online]. Available: http://protege.stanford.edu/
[33] "Apache Jena Project," [Online]. Available: http://jena.apache.org/
[34] M. Ruta, F. Scioscia, E. Di Sciascio, and G. Loseto, "Semantic-based enhancement of ISO/IEC 14543-3 EIB/KNX standard for building automation," *IEEE Trans. Ind. Inf.*, vol. 7, no. 4, pp. 731–739, Nov. 2011.



Son N. Han received the B.E. degree in applied mathematics from Hanoi University of Technology, Hanoi, Vietnam, in 2006, and the M.S. degree in computer science from The University of Seoul, Seoul, Korea, in 2009. He is currently working toward the Ph.D. degree at the Department of Wireless Networks and Multimedia Services of Telecom SudParis, Institut Mines-Telecom, Paris, France.

His research interests include Web of Things, Web Services, Semantic Web, DPWS and applications of Web Services on resource constrained environment. Previously, he was a Research Engineer with Korea Electronics and Telecommunications Research Institute (ETRI) from 2009 to 2011.

Gyu Myoung Lee (S'02–M'07–SM'12) received the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2000 and 2007, respectively.

He is currently with the Institut Mines-Telecom, Telecom SudParis, Paris, France, and the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, as an Adjunct Associate Professor. He was an Invited Researcher with the Electronics and Telecommunications Research Institute (ETRI), Korea, for international standardization and was also a Guest Researcher with the National Institute of Standards and Technology (NIST), USA. He has contributed more than 250 proposals for standards and published more than 80 papers in academic journals and conferences. His research interests include Internet of Things, Web of Things, cloud-based multimedia services, and energy saving networks.

Prof. Lee has actively participated in standardization meetings including ITU-T SG 13 (Future Networks) as a Rapporteur and IETF.

Noel Crespi (SM'08) received the M.S. degree from the Universities of Orsay and Kent, a Diplome d'Ingénieur from Telecom ParisTech, Paris, France, and the Ph.D. and Habilitation degrees from Paris VI University, Paris.

In 1993, he was with CLIP, Bouygues Telecom, France Telecom R&D in 1995, and Nortel Networks in 1999. He joined Institut Mines-Telecom, Telecom SudParis, Paris, France, in 2002, where he is currently a Professor and Program Director, leading the Service Architecture Laboratory. He is appointed as coordinator for the standardization activities in ETSI and 3GPP. He is also a Visiting Professor with the Asian Institute of Technology and is on the four-person Scientific Advisory Board of FTW, Austria. His current research interests are in service architectures, P2P service overlays, future Internet, and Web-NGN convergence. He is the author or coauthor of more than 230 papers and contributions in standardization.