Science of Computer Programming 95 (2014) 112–134

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Exploiting semantic technologies in smart environments and grids: Emerging roles and case studies

Dario Bonino*, Giuseppe Procaccianti

Politecnico di Torino, Dipartimento di Automatica ed Informatica, Corso Duca degli Abruzzi 24, 10129, Torino, Italy

HIGHLIGHTS

- We provide an updated overview of roles played by semantic technologies in the smart environment and smart energy domain.
- We define general roles played by semantic technologies in these domains.
- We provide a case study on smart environments.
- We provide a case study on smart grids.
- We discuss the current and future landscape of semantic applications in the smart environment and smart grid domains.

ARTICLE INFO

Article history:
Received 31 March 2013
Received in revised form 12 February 2014
Accepted 14 February 2014
Available online 26 February 2014

Keywords:
Semantic technologies
Smart grid
Smart environments
Home automation
Ontologies

ABSTRACT

Semantic technologies are currently spreading across several application domains as a reliable and consistent mean to address challenges related to organization, manipulation, visualization and exchange of data and knowledge. Different roles are actually played by these techniques depending on the application domain, on the timing constraints, on the distributed nature of applications, and so on. This paper provides an overview of the roles played by semantic technologies in the domain of smart grids and smart environments, with a particular focus on changes brought by such technologies in the adopted architectures, programming techniques and tools. Motivations driving the adoption of semantics in these different, but strictly intertwined, fields are introduced using a strong application-driven perspective. Two real-world case studies in smart grids and smart environments are presented to exemplify the roles covered by such technologies and the changes they fostered in software engineering processes. Learned lessons are then distilled and future adoption scenarios discussed.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

In 2001 Tim–Berners Lee proposed the Semantic Web vision: a new generation of the World Wide Web in which content was given well-defined, machine understandable meaning, better enabling cooperation between humans and computers. Almost a decade later, Semantic technologies are recognized as a means to reliably address issues related to information processing, organization, manipulation, visualization and exchange. As a consequence, semantic-based applications are crossing the boundaries of research-level applications and more and more enterprise-grade solutions are now offered on the

* Corresponding author.
E-mail addresses: dario.bonino@polito.it (D. Bonino), giuseppe.procaccianti@polito.it (G. Procaccianti).

http://dx.doi.org/10.1016/j.scico.2014.02.018
0167-6423/© 2014 Elsevier B.V. All rights reserved.

market. This transition contributes to shape the role of semantics in software engineering and fosters previously unforeseen programming patterns.

This paper focuses on changes brought by semantic technologies in software engineering approaches and solutions typical of the home automation (smart homes) and smart grid domains. Starting from a detailed and updated overview of the state of the art in these fields, we analyze how semantic technologies impact the design of smart environments. The analysis follows two parallel paths: firstly we analyze emergent, cross-domain adoption patterns, using a top-down approach (see Section 3); secondly we address the specific domain of smart environments by focusing on two real-world case studies covering the vertical domain of energy monitoring and exchange, from smart environments (Section 4) to smart grids (Section 5). The two selected case studies, stem respectively from a 7-years old research effort carried by the some of the authors, currently part of a larger effort towards standardization of *Energy Using and Producing Products Management* [1], and from an on-going research in the smart grid domain involving one of biggest actors in the smart environment domain, in Italy, i.e. the Loccioni group. As shown in the paper, some clear roles for semantic technologies can be identified, involving both off-line modeling and run-time operation. The latter, in particular, is particularly interesting, from a computer programming standpoint, as it is currently fostering new programming paradigms mainly based on automatic code generation. In general, a tangible effort in finding the right trade-off between "pure" semantic-based operation, supported by full logic inference, and hybrid approaches were semantic information is translated into more operational representations, can be noticed. The first approach exploits the full power of logic to provide advanced data processing/management, but suffers from computationally intensive processes, seldom suited for on-line/real-time operation. The last, instead, is less formal, possibly dealing with, or generating, inconsistencies in modeled information, however it can easily be exploited at runtime, with computation loads typically lower than the full semantics approach. The results of our analysis provide hints on the increasing importance of semantics in smart home and smart grid domains, and, in the next years, we expect an even higher impact on related software engineering processes, particularly for what concerns monitoring and big-data issues.

The reminder of the paper is organized as follows: Section 2 depicts the current state of the art, highlighting typical cases of semantic-based approaches to smart homes and smart grids. Section 3 provides a top-down analysis of emergent roles of semantics in the knowledge domain tackled by the paper and provides an overview of technologies and approaches which are better detailed in the following sections. Sections 4, and 5 exemplify the roles identified in the former section by focusing on two real-world case studies respectively related to smart environments and smart grids. Finally, Section 6 discusses the emerging approaches and provides a view on the future trends in semantics-powered systems for the energy monitoring and exchange domain, while Section 7 concludes the paper and proposes future works.

2. State of the art

Firstly introduced by Tim Berners Lee et al. [2] and lately evolved in one of the main pillars of the next generation web, the Semantic Web and related technologies have quickly crossed the Web boundaries and have been considered as one of the enabling technologies for smart environments, in particular for smart homes and buildings. The earliest applications of semantic technologies (i.e., of technologies such as ontology modeling, reasoning, etc., stemming from the Semantic Web initiative) dealt with *context modeling*, a key technology for smart homes. In this domain, several research efforts [3–6] can be cited dating back to the first Semantic Web years.

2.1. Context modeling

COBRA-ONT [3], for example, is an ontology-based modeling effort for supporting pervasive context-aware systems, and provides a collection of ontologies for describing places, agents and events and their associated properties in an intelligent meeting-room domain. The ontological core of COBRA is SOUPA [7] a Standard Ontology for Ubiquitous and Pervasive Applications. SOUPA is expressed using the Web Ontology Language (OWL) [8] and includes modular component vocabularies to represent intelligent agents with associated beliefs, desires, and intentions, time, space, events, user profiles, actions, and policies for security and privacy.

Another relevant effort to adopt semantic technologies in the context of a pervasive computing environment (GAIA [9]) is reported in [4], where the use of ontologies helped to face some challenges that are not unique to pervasive computing, but that are faced by any multi-agent software system. The work considered three major issues that confront the development and deployment of pervasive computing environments: discovery and matchmaking, interoperability between different entities and context-awareness. From this work, several challenges related to the adoption of semantic technologies in intelligent environments emerged, part of which still have to be solved:

(a) the need to simplify the construction and maintenance of ontologies,
(b) the need to integrate ontologies with software generation and management, for example using ontologies to semi-automatically generate interfaces,[1]

[1] This has been extended to automatic code generation in the last few years.

(c) the need to handle ontology merging at run-time, and consequently the need for upper ontologies defining the smart environments knowledge domain,
(d) the need of languages with different expressive power, as Description Logic (DL) is not suited for some critical aspects of pervasive computing. For example it does not deal well with quantitative concepts; including order, quantity, time, or rates.

2.2. Interoperability

Smart environments are too often "simple environments" filled with smart devices, "isolated" and not collaborating with each other. Semantic technologies assume therefore a key role to bring such components together to provide real Ambient Intelligence (AmI, see [10] for an overview). In this case the main research efforts reported in the literature are about *modeling* and *interoperability*.

Among many relevant domain ontologies, the EHS taxonomy is one of the earliest home appliance classification systems, originally designed by the EHS (European Home System) consortium (now evolved in the Konnex Association[2]) that mainly describes so-called white and brown goods located in a domestic environment. It is deployed along four main classes: Meter Reading, which groups all measurement tools, House Keeping, which groups all household appliances and systems, Audio and Video, which encompasses multimedia appliances, and Telecommunication, grouping all tools able to establish a communication.

In [11] Chen et al. introduce semantic smart homes, a novel concept whose aim is to move from the current state-of-the-art of smart home technologies to the future infrastructure that is needed to fully support the smart home vision. They present a conceptual system architecture for semantic smart homes and elaborate functions, and explore the interplay of constituent components by focusing on the methodology of semantic modeling, content generation and management.

2.3. Domain and upper ontologies

DomoML [12,13] represents one of the first approaches providing a full, modular ontology for representing household environments. It describes operational and functional aspects together with some preliminary architectural and positioning information and is based on three core ontologies: DomoML-env, DomoML-fun and DomoML-core. DomoML-env provides primitives for the description of all "fixed" elements inside the house such as walls, furniture elements, doors, etc., and also supports the definition of the house layout by means of neighborhood and composition relationships. DomoML-fun provides means for describing the functionalities of each house device, in a technology independent manner. DomoML-core provides support for the correlation of elements described by DomoML-env and DomoML-fun constructs, including the definition of proper physical quantities.

More recently, the DogOnt [14] ontology[3] provided one of the few domain models specifically designed to fit real world domotic system capabilities and to support inter-operation between currently available and future solutions. Taking advantage of technologies developed in the context of the Semantic Web, the DogOnt ontology supports device/network independent description of houses, including both "controllable" and architectural elements. States and functionalities are automatically associated to the modeled elements through proper inheritance mechanisms, while automatic device recognition is achieved through classification reasoning.

When crossing the home boundaries, devices are no more "domotic" and tend to be modeled as sensors, somewhat interconnected by a communication mean. The Semantic Sensor Web (SSW) initiative tackles the world of sensors and related issues by encoding sensor descriptions and sensor observation data with Semantic Web languages to enable more expressive representation, advanced access, and formal analysis of sensor resources. The W3C Semantic Sensor Network Incubator group (SSN-XG), in particular, defined an OWL 2 ontology [15] to describe the capabilities and properties of sensors, the act of sensing and the resulting observations.

In the last few years, semantic modeling for smart environments witnessed the development of upper ontologies, as means to provide a solid conceptual framework for emerging domain-level representations. In 2011, as an example, Dobson et al. [16] introduced a top-level ontology for smart environments serving as a formal backbone when designing domain-specific ontologies, linking the meaning implicit in elementary information to higher-level information that is of interest to applications. In this way their ontology aims at providing the common semantics for information at different levels of granularity, which supports the communication, reuse and sharing of ontologies between systems.

2.4. Run-time operation

This increasing availability of modeling solutions grounded on real-devices and extending up to higher conceptual layers, has given birth to a variety ontology-based ambient intelligence systems [17,18]. As a consequence, new roles for semantic technologies emerge, which are currently crossing the domain of static modeling and start to tackle run-time operations [19]

[2] http://www.konnex.org.
[3] http://elite.polito.it/ontologies/dogont.owl.

and related issues, also pertaining previously neglected domains such as: energy measurement [20] and performance certification [21], simulation [22,23], etc.

While it is well known that reasoning has not a good performance for real-time systems, SPARQL-based approaches are used to get a better performance. Current efforts on C-SPARQL or EP-SPARQL, for example, try to bring old-fashioned techniques from the Complex Event Processing and Stream Reasoning areas to the Semantic Web. This should allow better tackling of real-time operations.

Xu et al. [24], for example, propose an ontology-based framework to facilitate automatic composition of applications for smart environments. Their system composes appropriate services depending upon the available equipments in each individual household, automatically. Moreover, it dynamically adjusts environment parameters to match customer needs and available resources. Users are able to specify their preferred behavior templates and to personalize existing ones, thus guiding the services composition towards their own needs.

In [25], Katasonov and Palviainen extend the process of software development for smart environments with a full ontology-driven approach. The goal of their work is, in fact, to raise the level of abstraction of smart application development, for enabling end-user programming, and to partially automate the process to make it easier and faster. A Smart Modeler enables the developer to graphically create a model of a smart space application and, a semantics-based framework provides core interfaces for extensions, supporting both model and ontology-driven development. These extensions enable: ontology-based creation of model elements, discovery and reuse of software components and partial models, through a repository mechanism, and, generation of executable programming code for models.

2.5. Smart grid

The application of semantic technologies to the smart grid domain has been widely discussed in the scientific community in the last few years. The main scientific efforts were directed to the definition of a common ontology for this domain. The first examples of complete software frameworks and architectures, which leverage these technologies to manage electric networks, are currently emerging.

In 2010, the National Institute of Standards and Technology (NIST) published a roadmap for smart grid interoperability [26], where a high-level reference model of the smart grid is defined. This model provides an architectural view of the smart grid domain, that helps in the identification of the actors, use cases and interactions of the next generation energy network. Through this analysis, NIST identified the top technical priorities for the smart grid development, and selected 75 existing standards, related to the smart grid environment, which address these priorities.

The Common Information Model (CIM) by the International Electrotechnical Commission (IEC) is defined in two of those standards, IEC 61970 [27] and IEC 61968 [28]. The first defines the semantic model, at the electrical level, of a power system, in terms of components and relationships between them. The latter provides a more high-level description of the processes involved, such as energy billing, scheduling and monitoring. The conceptual model depicted in the CIM was firstly translated into UML by a research group at ABB [29], then Hughes [30] proposed to adopt the OWL language, in order to solve the problem of the harmonization with another IEC standard, the 61850 [31] and create an Unified Model for the electric systems. This harmonization has been the main subject of many studies: for example, Santodomingo et al. [32] propose to adopt an Ontology Matching approach, developed for the Semantic Web, to perform this matching. They also developed a tool, called ESODAT, which automatically performs the translation between the two ontologies [33].

The CIM ontology, however, may be also considered as a component to a more complex ontology for the smart grids. For example, Grassi et al. [34] proposed an ontology framework, composed of several ontologies: in particular, two of them, the Device and Energy ontologies, combine the concepts of Smart Homes and smart grids. This enables the framework to express control strategies at a fine granularity level. In this contribution, we will present another solution based on the same principle.

Another field of application of semantic technologies in the smart grid domain is data management. Typically, smart grids produce large amounts of data, coming from distributed sensors and monitoring systems. However, rather than storing a continuous flow of data, it would be wiser, from a system manager standpoint, to extract and store only the useful information. In such a case, semantic technologies can be successfully exploited. In this sense, Pena and Penya [35] illustrate how Semantic Web repositories, commonly known as *triplestores*, can be used to store the relevant information and extract it using semantic query languages such as SPARQL.

3. Emergent roles

Semantic technologies lie at the basis of a great variety of applications in the smart environment and smart grid domains, and play different roles depending on the specific application goals. Although the adoption landscape is vast and diversified, as hinted by Section 2, some clear roles of such technologies emerge, encompassing: (a) context-modeling, (b) environment-modeling, (c) interoperability, (d) off-line reasoning, (e) code generation, (f) on-line reasoning and (g) run-time operation. In this section a more systematic overview of these roles is provided, describing each role and relating the roles with the computational and application requirements they typically address.

3.1. Role definition

3.1.1. Context-modeling

This is one of the first roles played by ontologies and semantic techniques in the smart environments domain. Context information is defined by G.D. Abowd, A.K. Dey et al. [36] as

> Any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects.

Representing context information through semantic constructs and operators allows:

(a) easier information processing, as information is given well-defined and machine understandable meaning,
(b) safer information management, as data inconsistencies can be easily detected by means of description logic,
(c) better knowledge handling and discovery, by means of automated inference processing.

Moreover, according to the literature [37,38] ontological models of context provide clear advantages in terms of distributed composition, partial validation, richness and quality of information, incompleteness and ambiguity, formalism, applicability to existing environments, heterogeneity and interoperability. Of course some aspect still deserve investigation, e.g., time modeling[4] for which little support is provided in current ontologies, topological aspects for which approaches [7,39] are available exploiting the Region Connection Calculus (RCC) [40], or performance (typically with NEXPTIME worst case complexity) [39,41].

3.1.2. Environment-modeling

This role might be partially overlapping the previous one, but generally identifies a quite distinct modeling target. To better clarify the interactions between these two roles, we can draw a parallel with the Computation Independent Model (CIM) and Platform Independent Model (PIM) defined in the Model Driven Architecture (MDA). The CIM uses a vocabulary that is familiar to the subject matter experts (SMEs), and presents exactly what the system is expected to do, but hides all information technology related specifications to remain independent of how that system will be (or currently is) implemented. The Platform Independent Model (PIM), instead, exhibits a sufficient degree of independence so as to enable its mapping to one or more platforms, but a much more specific representation of technical details, typically in terms of services. While Context modeling can be mapped to CIM models, in this parallel, Environment modeling can be mapped to PIM.

In other words, Environment modeling aims at *representing the environment configuration and capabilities in detail*, thus enabling applications, be they middleware (e.g., home gateways) or end-user applications (i.e., user interfaces), to exploit this information to define interaction patterns, to simulate the environment behavior, to support interoperability. Typical examples of semantic technologies playing this role are the DogOnt [14] and the EEOnt ontology [21].

3.1.3. Interoperability

In the interoperability case, semantic technologies, and in particular ontologies, are used as a *shared cross-language bridging communication between devices and automation networks having different native technologies and protocols*.

In the software agents domain, interoperability is defined as the ability of cooperating through data or processes, and can only be achieved if agents can interact and communicate with other agents using a common (shared) vocabulary. Semantic-based interoperability involves the adoption of the same vocabulary (ontology) and data-model (RDF). In the smart environment domain, instead, interoperability involves different home automation technologies, i.e., subsystems and has often to be ensured by a (de)centralized controller, as devices are typically “dumb” (they seldom carry enough computational power on board to run a software agent).

In this scenario, semantics-based interoperability can be deployed both statically, i.e., off-line, or dynamically, i.e., on-line. In the former case, ontologies define cross-technology semantics and primitives; a suitable software infrastructure takes care of translating low-level protocols in such higher-level representations. The translation might include some interaction paradigm conversion, e.g., master-slave to peer-to-peer, or polling-based operation to event-based interaction. In on-line approaches, instead, static ontology information is exploited to: (a) generate run-time inter-operation rules to automatically bridge different technologies (e.g., as in [42,43]), with the aim of composing cross-technology home services, e.g., switching on a smart oven through the television remote control, (b) correctly face faulty or anomalous situations as in [44], etc.

3.1.4. Off-line reasoning

When semantic technologies are used to extract implicit information contained in ontology-based environment and context models, they play the role of off-line reasoning. In this role, semantics and, in particular automated reasoning

[4] E.g., http://www.w3.org/TR/owl-time/.

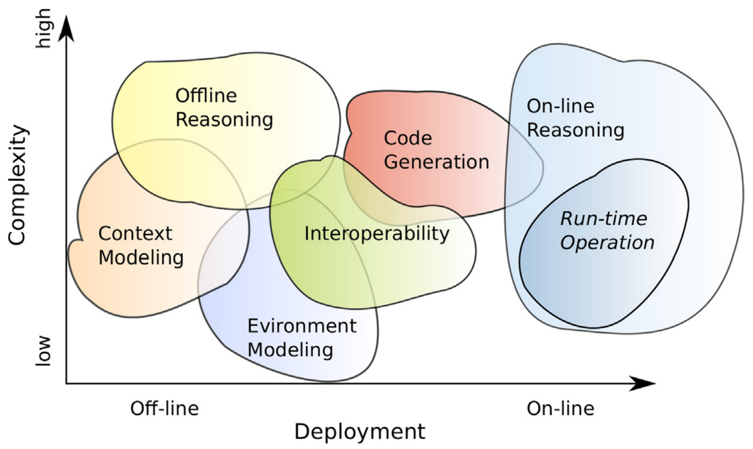

Fig. 1. Overview of the roles played by semantic technologies in smart grid and environments.

techniques, enable off-line computation of relevant environment and device properties typically related to design, usability and normative issues [21,45–47]. These properties can be asserted on the sole basis of the environment structure, e.g., the layout of walls or the location of specific devices, and they are usually checked off-line. They are particularly useful for validating design choices taken by architects and home automation engineers against architectural, operational and configuration constraints.

3.1.5. Code generation

Stemming from the validated model-driven development approach, and fostered by the inherent complexity of smart environments, new approaches are emerging, in literature [48–51], where ontologies assume the role of normative reference for application development. Automatic code generation techniques then enforce consistency of application code and formal representations, and support quick and error safe development of relevant portions of the smart environment logic. Some application of this new role is also emerging in assistive robot research [52], which is tightly interconnected with smart environments and ambient intelligence research.

3.1.6. On-line reasoning

Context-aware and ambient intelligence systems proactively interact with users by proposing actions or alternatives based on updated knowledge of the ambient state and capabilities. In the last few years several research works have shown that such a knowledge can be effectively modeled through semantics, by using ontologies, and by providing systems with the ability of performing inference at run-time [6,53,54], through incremental reasoning techniques and/or reasoning with ambiguity. These approaches define a new role for semantic techniques that we name "On-line reasoning." While reasoning performance issues are still to be successfully addressed, the approach is promising and is worth of consideration in smart environments design as it allows to uniformly tackle a wide variety of design concerns, mostly related with the previously cited roles.

Run-time operation can be seen as an extension of the on-line reasoning role where updated information triggers actions by means of inference processes. Typical examples of technologies playing this role can be found in [55,47].

3.2. Role peculiarities

Roles played by semantic technologies in the smart environments and smart grid domain can be characterized from several standpoints, e.g., application scenarios, involved complexity, performance, etc. Fig. 1 roughly summarizes the complexity facet (bubble shapes are used to show complexity levels and possible intersections between roles) with a very high-level categorization of the above-defined roles in terms of complexity vs. real-time requirements.

Approaches involving logic inference typically imply higher computational complexity and are best suited for off-line deployment whereas approaches involving code generation or simple ontology querying can be successfully exploited at run-time. Clearly run-time approaches based on logical inference are subject to stricter requirements as they feature high complexity (typically NEXPTIME) and near real-time requirements.

4. Semantic technologies in smart environments

Top-down roles identified in Section 3 are integrated at different degrees in current smart home systems. In this section we provide deeper insights on the technology roles and applications by focusing on a case study which integrates semantic

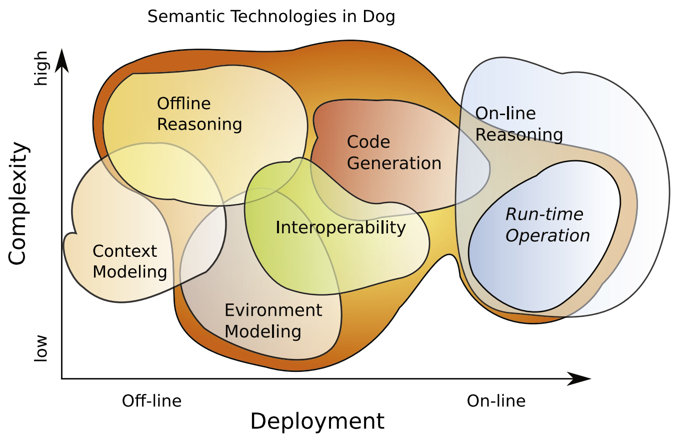

Fig. 2. Roles played by semantic technologies in Dog.

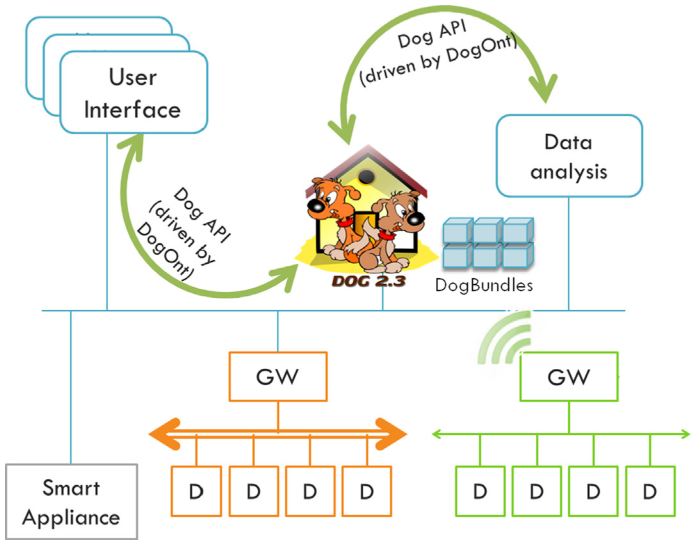

Fig. 3. The logic architecture of a Dog-based Intelligent Domotic Environment (GW stands for Gateway, D for device; simple lines identify an IP-based network, whereas horizontal double arrows are used to represent low-level communication infrastructures, be they using field buses or wireless connections. Curved green arrows represent high-level protocols).

technologies in almost all discussed roles (see Fig. 2). The case study involves the Domotic OSGI Gateway (Dog,[5] firstly presented in literature in 2008 [18], and still very active), and analyzes the system architecture from the semantic technology standpoint. As a consequence this section does not provide full details on the software design and architecture but concentrates on semantic-based features, limiting other aspects to a very high-level overview.

4.1. Dog from 30'000 feet

Dog (Domotic OSGi Gateway) is a home gateway based on the OSGi framework and designed to provide smart home capabilities to environments equipped with commercial home automation systems. It was firstly presented in literature in 2008 [18], by building on a former effort on home gateways [56], and it is still evolving to include new features and bridge new technologies. The latest release is Dog2.5 and includes support for most bus-based technologies (except X10), and for wireless home automation solutions such as ZigBee (Home Automation v1.0 [57]), Z-Wave and EnOcean (under development).

The Dog design aims at overcoming typical inter-operation issues related to diverse and often competing market stakeholders, and at offering high-level, semantics-based foundations for Intelligent Domotic Environments (IDEs), i.e., for "Environment settings in which existing automation technologies are interfaced by a low cost device (gateway) providing *neutral* access to the environment for inter-operation, intelligent automation scenarios, energy saving, etc.". Fig. 3 reports the logic

[5] http://domoticdog.sourceforge.net.

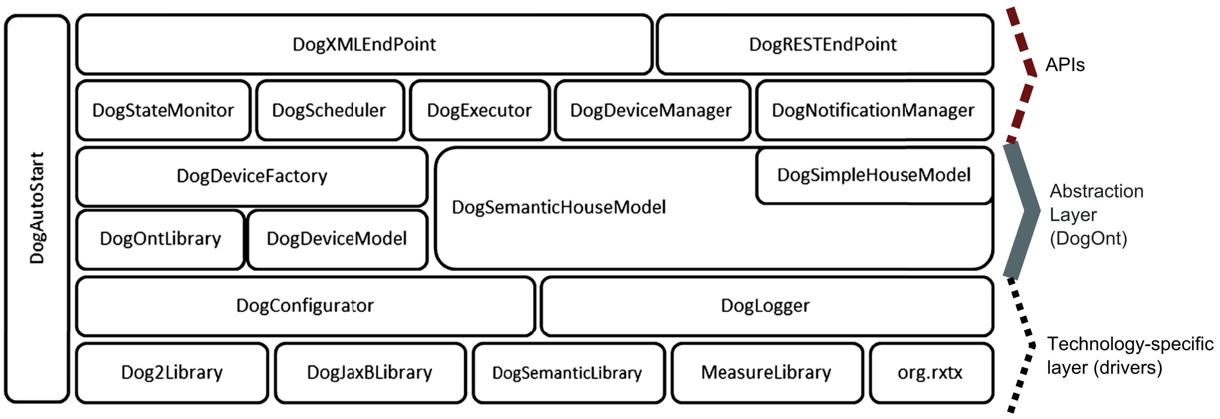

Fig. 4. The inner Dog architecture from 30 000 feet (angle brackets on the right group modules into different abstraction layers).

architecture of the approach where Dog acts as a bridge between low-level technology specific protocols and high-level requirements of applications, be they “simple” interfaces or complex data-processing systems (e.g., for energy monitoring).

The key aspect of the approach is *neutral* access which entails the ability to interface, operate and query devices independently from the specific technology with which they are realized. Such a neutrality is based on a cross-language whose main terms and operators are defined by means of a domain-specific ontology (DogOnt [14]) and on a driver-based technology abstraction layer (Fig. 4 shows the Dog architecture from 30 000 feet, while more details can be found in [18]) based on the Equinox OSGi framework.

4.2. DogOnt: a quick overview

DogOnt is a domain ontology specifically designed to model smart homes equipped with off-the-shelf domotic plants and intelligent appliances (for a complete description of the DogOnt design and modeling capabilities see [14]). It is organized along 5 main hierarchies of concepts (Fig. 5, hierarchy roots in bold) supporting the description of:

- the environment structure (rooms, walls, doors, etc.), by means of concepts descending from *BuildingEnvironment*;
- the type of domotic devices and of smart appliances (concepts descending from the *Controllable* subclass of the *BuildingThing* main concept);
- the working configurations that devices can assume, modeled by *States* and *StateValues* (see the following paragraphs for more details);
- the device capabilities (*Functionalities*) in terms of accepted events and generated messages, i.e., *Commands* and *Notifications*;
- the technology-specific information needed for interfacing real-world devices (*NetworkComponent*), and
- the kind of furniture placed in the home (concepts descending from the *UnControllable* subclass of the *BuildingThing* main concept).

DogOnt models domotic devices in terms of functionalities and states.

Functionalities. They describe the device under the viewpoint of device interaction capabilities, i.e., they describe how a given device can be controlled, queried and whether it can autonomously generate “events.” For example, while a lamp can only be switched on and off, a light sensor can either be queried for the current measured luminance or can autonomously send luminance change events at regular time intervals, or upon threshold trespassing. DogOnt functionalities include:

- *ControlFunctionalities*, modeling the ability of a device to be controlled by means of some message or command,
- *QueryFunctionalities*, modeling the ability of a device to be queried about its current state, and
- *NotificationFunctionalities*, modeling the ability of a device to issue notifications about state changes, in an event-driven interaction model.

Functionalities are either associated with commands (for *ControlFunctionalities*) or with notifications (*NotificationFunctionalities*) that further detail the specific operations supported by DogOnt device instances.

States. They describe the various stable configurations that a device can assume during its working life-cycle. From the modeling point of view, each device may include one or more different simultaneous behaviors. If we refer to a CD player, it can either be on or off, it can be playing a CD track with a given number, and it may have a specific output volume.

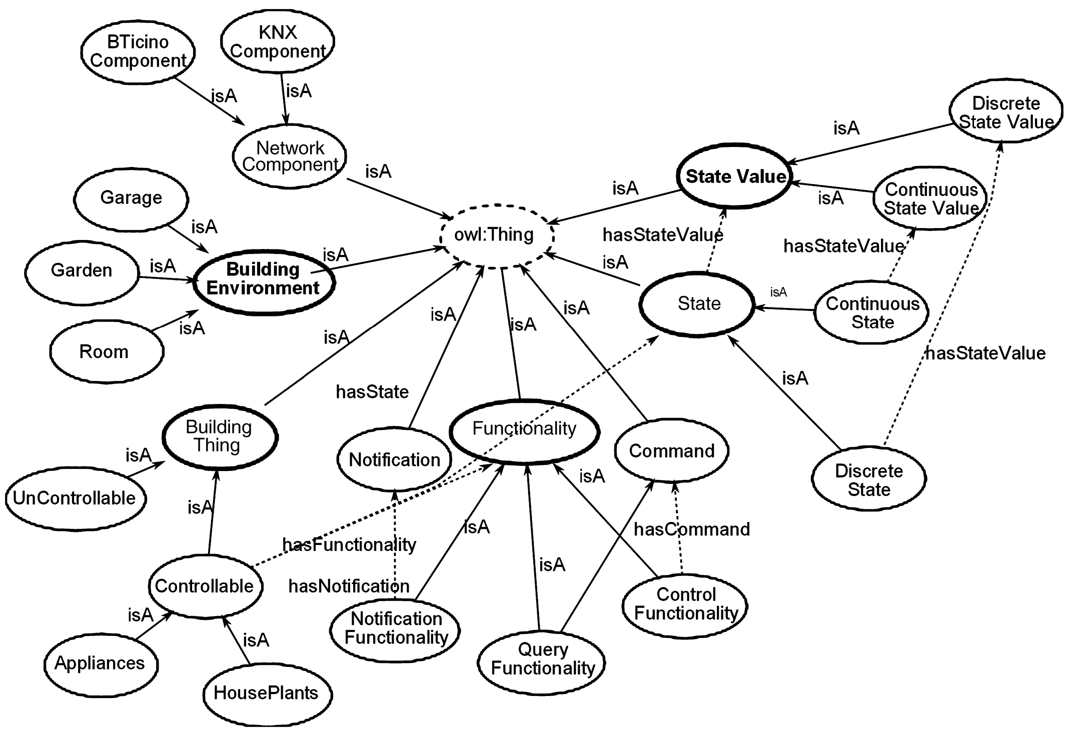

Fig. 5. DogOnt in a nutshell (Solid lines represent inheritance relations, i.e., *isA*, whereas dotted lines identify other relationships; the dashed shape around `owl:Thing` highlights the implicit presence of such a concept, as dictated by the OWL specification [8]).

In DogOnt such behaviors are called `dogont:States`. The description of each `dogont:State` is represented by a set of identifiers, called `dogont:StateValue`, that model each operating condition. For example the CD player is modeled as having three independent `dogont:States` (with discrete or continuous values): a `dogont:OnOffState`, a `dogont:PlayingState` and a `dogont:VolumeLevelState`. Each of these three states includes a specific set of possible state values (for example, the first state includes a `dogont:OnStateValue` and a `dogont:OffStateValue`). The current state of a device is therefore defined by a list containing one `dogont:StateValue` per each `dogont:State`.

4.3. Roles of semantic technologies in Dog

Since discussing the Dog architecture design and implementation is out of the scope of this paper, we discuss what are the roles played by semantic technologies in Dog and the issues solved by them. Dog exploits semantic technologies at different degrees:

(a) DogOnt classes and relationships define the API-level interfaces through automatic code generation,
(b) DogOnt instances define the configuration of specific environments, both at the gateway and at the API level,
(c) Basic ontology-merging allows plug-in modules (bundles) to dynamically add information to the house model, which can then be exploited by all gateway bundles,
(d) Ontology querying/inference supports automatic generation of inter-operation rules,
(e) Off-line ontology querying/reasoning supports generation of emulation drivers,
(f) Off-line ontology inference permits to check structural properties of the environment,
(g) On-line ontology inference supports checking of properties depending on the current state of devices (run-time).

4.3.1. Automatic code generation (off-line)

In Dog, all the Java classes defining devices and the respective commands, states and notifications (Fig. 4, DogOntLibrary and DogDeviceModel), are generated off-line by extracting the relevant knowledge from the DogOnt ontology (Fig. 6).

Starting from the DogOnt ontology classes and relationships, a tool (DogOnt2Dog, included in the Dog source release), generates the device representation classes (Java) used by Dog as inner model. The code generation process is deployed along the following steps:

(1) *Ontology query*. The ontology is analyzed and queried to extract device classes (inheriting from `dogont:Controllable`) and related information. This process exploits the Apache Jena framework and is accomplished by executing

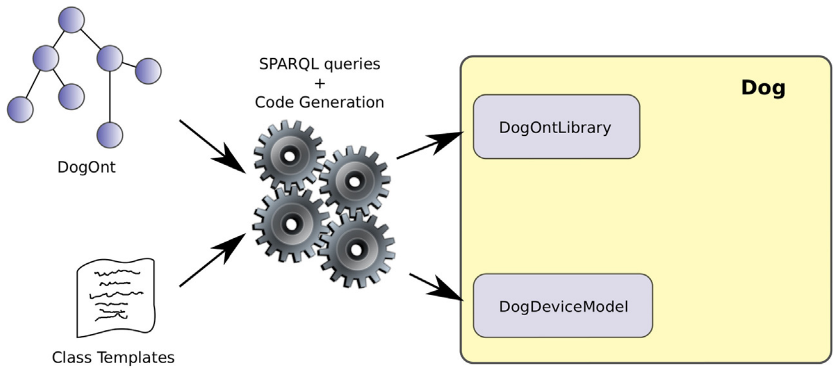

Fig. 6. Semantics-based library generation in Dog.

```
SELECT DISTINCT ?notification ?notifValue ?notifParamValue
  ?functionality WHERE {{
 <URI_of_the_Device_Class>
  rdfs:subClassOf [rdf:type owl:Restriction;
 owl:onProperty dogont:hasFunctionality;
 owl:someValuesFrom ?functionality] .
 ?functionality rdfs:subClassOf dogont:NotificationFunctionality.
 ?functionality rdfs:subClassOf [rdf:type owl:Restriction;
 owl:onProperty dogont:hasNotification;
 owl:someValuesFrom ?notification] .
 ?notification rdfs:subClassOf dogont:Notification .
 ?notification rdfs:subClassOf [rdf:type owl:Restriction;
 owl:onProperty dogont:notificationName;
 owl:hasValue ?notifValue] . OPTIONAL
 {?notification rdfs:subClassOf [rdf:type owl:Restriction;
 owl:onProperty dogont:notificationParamName;
 owl:hasValue ?notifParamValue]}}
 UNION
 {
 <URI_of_the_Device_Class>   rdfs:subClassOf ?ancestor.
 ?ancestor rdfs:subClassOf dogont:Controllable.
 ?ancestor rdfs:subClassOf [rdf:type owl:Restriction;
 owl:onProperty dogont:hasFunctionality;
 owl:someValuesFrom ?functionality] .
 ?functionality rdfs:subClassOf dogont:NotificationFunctionality.
 ?functionality rdfs:subClassOf [rdf:type owl:Restriction;
 owl:onProperty dogont:hasNotification;
 owl:someValuesFrom ?notification] .
 ?notification rdfs:subClassOf dogont:Notification .
 ?notification rdfs:subClassOf [rdf:type owl:Restriction;
 owl:onProperty dogont:notificationName;
 owl:hasValue ?notifValue] . OPTIONAL
 {?notification rdfs:subClassOf [rdf:type owl:Restriction;
 owl:onProperty dogont:notificationParamName;
 owl:hasValue ?notifParamValue]}}};
```

Fig. 7. A sample of the SPARQL queries needed to extract device information from DogOnt, in particular for extracting notifications originating from a given `<URI_of_the_Device_Class>`.

(through the ARQ system), and combining, several SPARQL queries (see Fig. 7 for an example) to finally get a complete specification of each device, including functionalities, states, commands, etc.

(2) *Java Class generation*. For each complete device description, DogOnt2Dog loads the needed Java class templates defined in the Velocity VTL expression language and merges the template and the ontology extracted information thus generating a Java source file representing the device. The same process is applied for generating library classes for notifications, states and state values defined in DogOnt. Generated classes are organized into homogeneous packages including:
- `it.polito.elite.dog.model.devicecategory` defines the interfaces describing the device (allows for instances inheriting from more than one device class)
- `it.polito.elite.dog.model.device` model defines the base implementation of a device class, in dog.
- `it.polito.elite.dog.model.notification` contains the OSGi event classes used to represent notifications generated by domotic devices
- `it.polito.elite.dog.model.state` represent the current state of a device as a set of state values

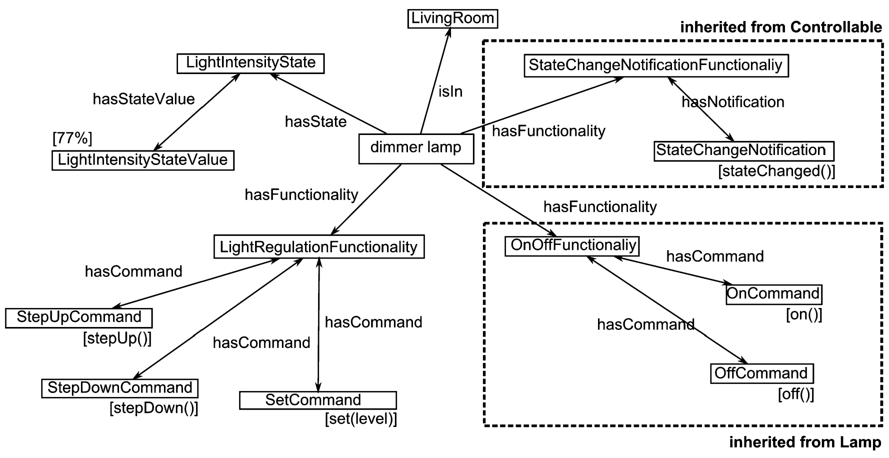

Fig. 8. Sample device (dimmer lamp) description through instances of DogOnt classes. Squares represent concept instances, which are specifically connected to the sample dimmer lamp.

- `it.polito.elite.dog.model.statevalue` defines classes for holding device state values; available Java classes reflect the `dogont:StateValue` classes defined in DogOnt.

(3) *Runtime Compilation*. Generated source files are compiled at runtime by exploiting the java compiler (`javac`) through the Java Virtual Machine (JVM) tools library (`tools.jar`, available as part of the standard Java Development Kit) and result in a folder structure mirroring the source one and containing compiled class files (`*.class`). Classes and libraries needed for successful compilation are part of the DogOnt2Dog tool and they are made available to the JVM by properly setting-up the classpath environment variable.

(4) *Bundling*. Compiled classes are integrated with a suitable manifest file (generated through a similar VTL template) and bundled in 2 different jar files, i.e., bundles which can then be used in Dog. The 2 bundles contain the device model and all the other packages, respectively. Such a separation allows for easier maintenance of Dog versions, enabling separate evolution of device models and related interfaces (be they device categories, notifications, states or state values).

This solution allows to tackle two distinct issues in the Dog development process:

(a) error-prone manual coding of over 500 gateway classes;
(b) full consistency of the Dog implementation with respect to the DogOnt ontology model;

and requires a negligible time to be executed: around 90s vs several days in the manual coding case.

4.3.2. *Instance-based configuration of specific environments*

Dog is designed to run as the "core" software of an home gateway enabling neutral access to real devices in real environments. How the environment is configured (architecturally speaking) and what devices are available is a property of the specific setup and must be configured during the installation process. In Dog this operation is based on an environment description given as a set of DogOnt class instances. Each device is described by (manually) creating instances for the relevant classes defined in DogOnt, including the device type, functionalities and possible states (see Fig. 8 for an example).

4.3.3. *Basic run-time ontology merging*

The core device modeling provided by DogOnt and exploited by Dog can be extended to include aspects pertaining different, related domains, e.g., energy, context modeling, etc. While the main house description (core ontology classes and instances) is managed by a dedicated bundle, the SemanticHouseModel, additional modeling concerns can be "plugged" at run-time by other bundles, through run-time ontology merging. For example, in the energy domain, a dedicated Dog bundle (PowerModel) provides support to consumption estimation by merging the DogOnt environment model and the device class power consumption information, at run-time. The latter, in particular, is modeled by means of a light-weight ontology (DogPower) designed to model power consumption of electrical devices and appliances in (automated) homes (see Fig. 9). A minimal modeling approach is adopted, reducing primitives (classes and relations) to those strictly needed to support power consumption modeling. Relations to described devices and appliances are left "open," i.e., their descriptions shall be completely formalized depending on the ontology-based home/device model to which the DogPower ontology is connected.

This "openness" is exploited to attach power descriptions to DogOnt instances (Fig. 10), at runtime. While the environment configuration (discussed in the previous section) is loaded by Dog at start-up, the power model of available devices is loaded lately by the PowerModel bundle. Then such a model is merged to the configuration model (by the Semantic-HouseModel bundle, see Fig. 4) to associate power consumption information to managed devices. Whenever a device gets

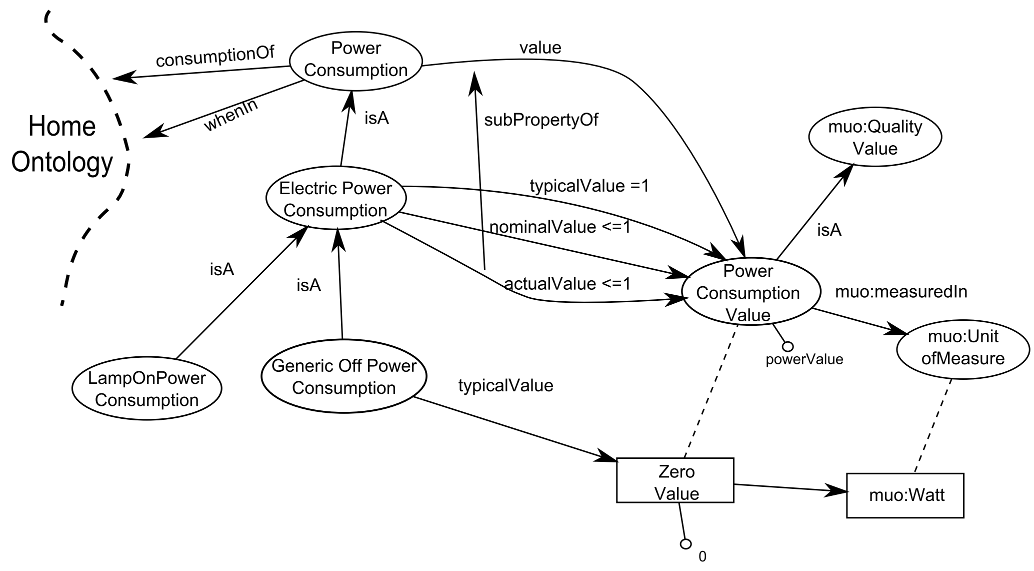

Fig. 9. The DogPower ontology.

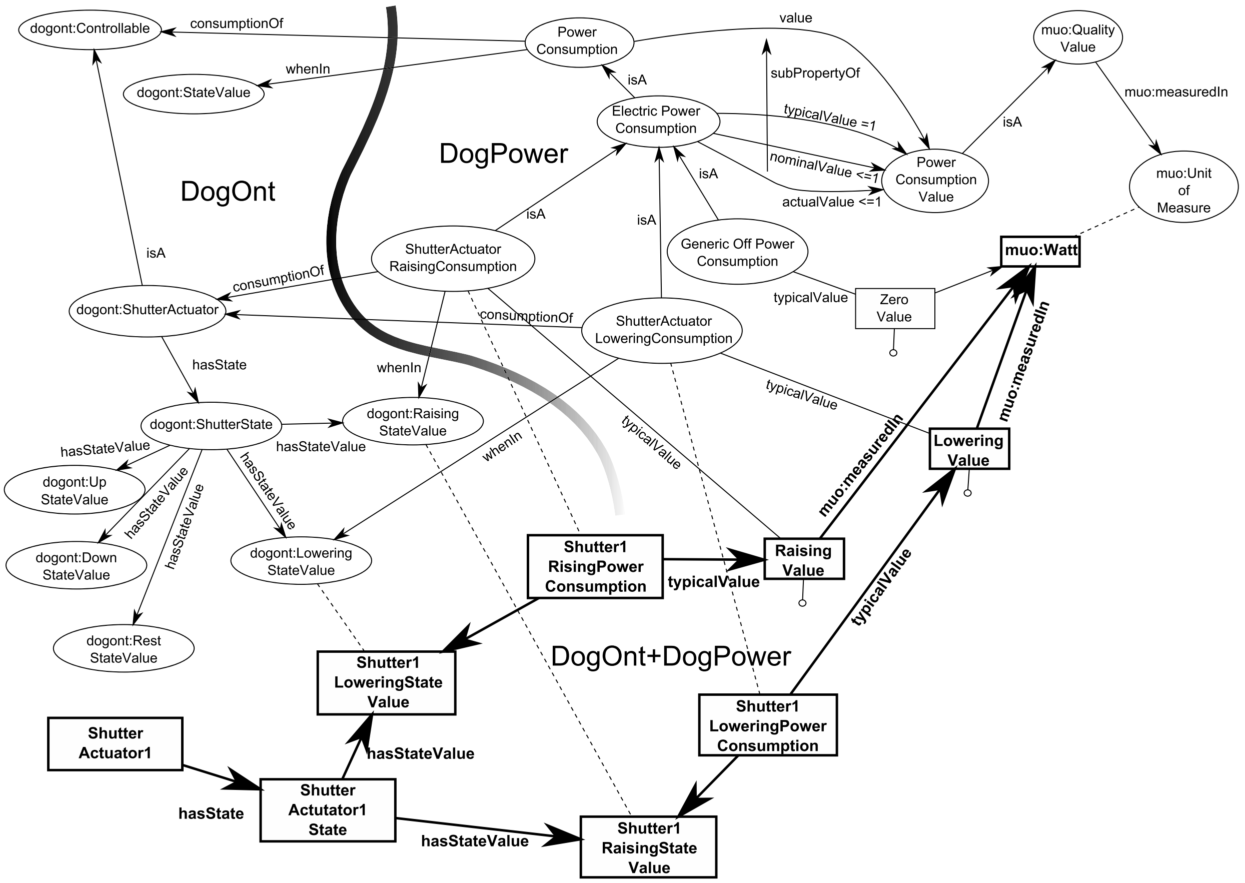

Fig. 10. Integration between a DogOnt device model and the corresponding DogPower consumption description. The dark, thick, line draws the separation between DogOnt-defined classes and DogPower consumption descriptions, whereas rectangles identify instances used to represent a shutter actuator consumption. Arrows with solid, tiny, lines identify relationships while dashed lines identify the `rdf:type` of each represented instance.

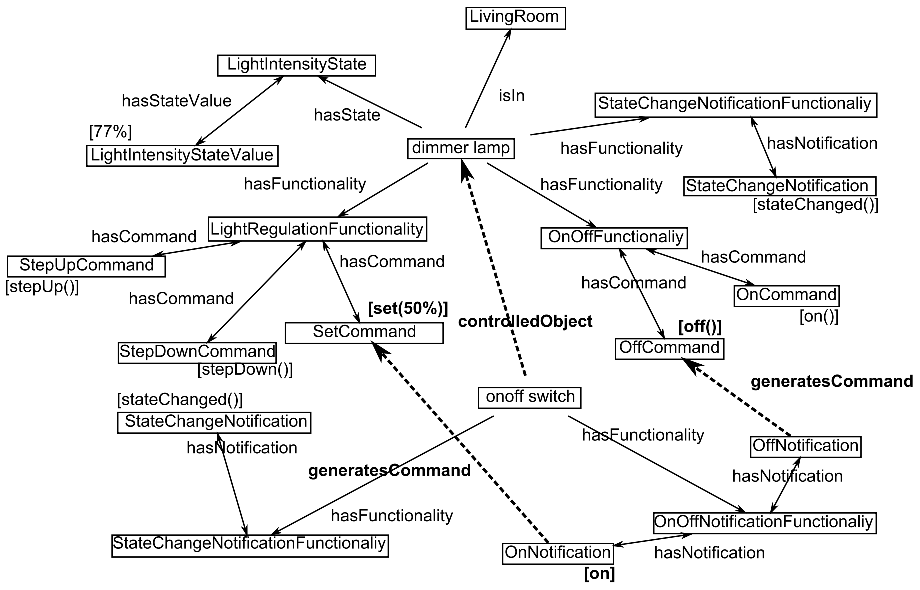

Fig. 11. Inter-communication between a 2-way switch and a dimmer lamp (thick, dashed lines represent relationships allowing to mine the information needed to generate suitable interoperation rules).

activated (either by the final user or by some internal process) the corresponding consumption information is retrieved (through SPARQL) and combined with any runtime information coming from real meters, to provide the best possible estimation of the device power absorption.

4.3.4. Automatic generation of inter-operation rules

Inter-operation between domotic devices, either connected to the same or to different domotic plants (or networks), is modeled in DogOnt by associating notifications of "controller" devices (`dogont:Control`) with commands of "controlled" ones. This association is formally represented by two relationships instantiated at the device level (`dogont:controlledObject`) and at the functionality level (`dogont:generatesCommand`), respectively.

At the device level, inter-operation is modeled with a very light master-slave approach: every device can control or can be controlled by another device, and may also change its role in time due to home configuration changes. As an example, an Hi-Fi amplifier usually acts as controller for the Hi-Fi components, e.g., the CDPlayer and the MP3Player. Nevertheless, it can also assume the role of controlled device, e.g., if it is remotely controlled by a PC. In general, while some devices (e.g., thermostats, Hi-Fi systems, etc.) can play both as controller and controlled device, other, simpler, domotic devices can only behave as "controlled" devices, e.g., Lamps.

To enable automatic inference on device-to-device inter-operation, e.g., to automatically generate inter-operation rules, a finer grained relationship is defined between `dogont:Notification` and `dogont:Command` instances: `dogont:generatesCommand`. This relation allows defining specific command(s) generated by specific notification(s) associated to functionalities of a specific `dogont:Control` device. For example it permits to associate the `dogont:OnNotification` of a simple `dogont:Switch` to the `dogont:OffCommand` of a Home Entertainment System.

Example 1 *(Controlling a Dimmer Lamp).* Fig. 11 shows how we can model in DogOnt a dimmer lamp controlled by a 2-way switch (`dogont:OnOffSwitch`). Two-way switches always have an `dogont:OnOffFunctionality` instance, describing their ability of being switched on or off. The `dogont:OnOffFunctionality` instance is in turn associated to two `dogont:Notification` instances: a `dogont:OnNotification` and a `dogont:OffNotification`, respectively. If the Dimmer Lamp shall be lit at 50% when the switch is in the on position and shall be switched off when the switch is off, then the `dogont:OnNotification` of the switch shall be connected to the `dogont:SetCommand(50%)` of the DimmerLamp. Accordingly, the `dogont:OffNotification` of the 2-way switch shall be connected to the `dogont:OffCommand` of the dimmer lamp.

Dog achieves automatic inter-operation through rule-based reasoning (as in [42], where the Drools[6] engine is exploited to execute rules at runtime), and leverages suitable communication primitives, included in the rule language, to listen for real device notifications and to send commands. Fig. 12 shows a prototypical rule (using the Drools rule syntax), where the

[6] http://www.jboss.org/drools/, last visited on September 09, 2013.

```
when
 device:?x
 receivedNotification(?x, ?v)
then
 device: ?y
 sendCommand(?y, ?cn, [?cv])
```

Fig. 12. An sample interoperation rule, values in brackets are optional.

```
SELECT DISTINCT ?x ?n ?v ?c ?d ?class ?cn WHERE
{
?x a dogont:Controllable .
?y a dogont:Controllable .
?x dogont:controlledObject ?y .
?x dogont:hasFunctionality ?f .
?f dogont:hasNotification ?n .
?n dogont:notificationValue ?v .
?n dogont:generateCommand ?c .
?d dogont:hasFunctionality ?f2 .
?f2 dogont:hasCommand ?c .
?c rdf:type ?class .
?class rdfs:subClassOf dogont:DiscreteCommand .
?class rdfs:subClassOf [rdf:type
owl:Restriction; owl:onProperty
dogont:realCommandName;
owl:hasValue ?cn]
}
```

Fig. 13. A sample SPARQL query for extracting inter-operation data.

`when` clause represents the rule antecedent and the `then` clause the rule consequent. It is important to notice that the rule structure is completely independent from the device type and technology.

To enable automatic interoperation the `?x`,`?v`, `?y`, `?cn` and `?cv` variables are automatically extracted from DogOnt device, notification and command instances. This requires to first compute the transitive closure of the DogOnt model instantiation (environment model, Section 4.3.2) to derive all direct and inherited information involving declared device instances, e.g. to infer that an *OnOffSwitch* instance is also an instance of *Switch*, *ElectricComponent*, *Controllable*, etc. Secondly, SPARQL querying is exploited to extract, for each controller device `?x`, the list of controlled devices `?y` and to further retrieve the controller device notifications `?v` and the commands accepted by the controlled device `?cn` (and their values `?cv`, when available). Fig. 13 shows the query for devices accepting *DiscreteCommands*, a similar query is defined for *ContinuousCommands*, also retrieving the command values.

4.4. Generation of emulation drivers

In Dog, the operational knowledge needed to simulate the behavior of a real device is automatically associated to DogOnt device instances by means of ontology querying and by customizing a library of state diagram templates. Device interconnections, modeled in DogOnt as semantic relations, are translated into event-remapping machines exploiting the event messaging framework defined by the state chart formalism [58]. Simulation of device behavior is supported at run-time through the DogSim API [59], which embeds a state machine execution engine based on the Apache Commons SCXML library,[7] and by providing event-driven interaction with the IDE model.

This integration allows exploiting the simulation abilities of DogSim in a mixed environment where state machine events can both be synthetic (as in the DogSim approach) or can stem from real device interactions thanks to the Dog messaging framework. During emulation, the control and intelligence parts of Dog deal uniformly with real and simulated devices, and which devices are actually simulated is unknown to the upper layers of Dog. In fact, emulated devices are handled by a special "network driver" (EmuDog) that, from the point of view of the gateway, is indistinguishable from other network drivers used to control real plants.

Deployment of emulated devices involves two distinct phases: a first initialization in which emulator machines are created and a second phase of run-time operation where emulators are executed. In the initialization phase (Fig. 14), the DogSemanticHouseModel loads the DogOnt ontology and the corresponding instances and performs the transitive closure of the ontology. Then, the EmuDog driver discovers devices needing emulation, i.e., devices for which no network driver has been found or devices that explicitly require emulation through a special property `dogont:emulateThis` being set.

[7] http://commons.apache.org/proper/commons-scxml.

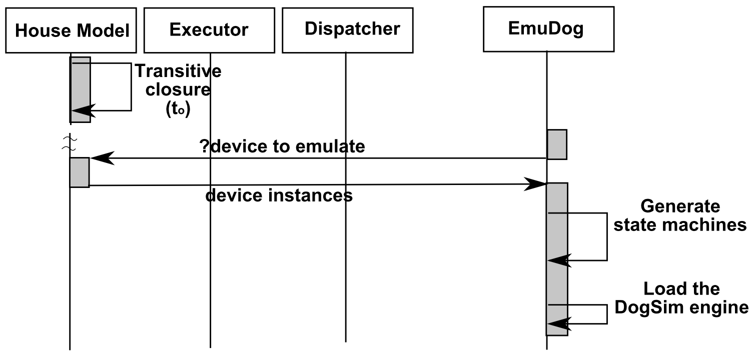

Fig. 14. The EmuDog initialization sequence diagram.

Table 1
Examples of structural properties.

Domain	Name	Description
Architectural, normative	accessibleControls	In Italy a law (DPR n. 503 24/07/1996) imposes that every button, switch or control available in a home shall be accessible to people on a wheel chair, i.e., that every control must be positioned at a height between 40 and 120 cm from the floor.
Operational	usableLamp	All the lamps in a given *Room* must be controllable at least by a switch either located in the same *Room* or in an adjacent *Room*.
Architectural	reachability	Every *Room* shall be reachable from any other *Room* in a finite number of room traversals.

For each found device, EmuDog builds the corresponding state machine, exploiting the DogOnt2SCXML module. Generated state machines are loaded in the DogSim Engine integrated in EmuDog and the bundle becomes ready.

At runtime, the EmuDog works as any other Network Driver in Dog: it receives messages requesting device actions (e.g., to switch a lamp on) and injects them as new events in the DogSim Engine. Running machines possibly change their states as a consequence of the injected event and detected state changes are then packed into one or more messages and sent back to the rest of the platform.

4.5. Structural properties checking

Structural environment properties encompass features related to design, usability and normative issues. They can be asserted on the sole basis of the environment structure, e.g., the layout of walls or the location of specific devices, and they are usually checked off-line.

Structural properties are typically extracted from Building Information Modeling (BIM) data, defined at design time, and are particularly useful for validating choices taken by architects and domotic engineers against architectural, operational and configuration constraints. Examples are reported in Table 1.

Representing devices and environment features through ontology concepts (as in DogOnt) enables effective and automatic checking of these properties, thus improving the overall design of smart environments [47]. As shown by the following example, semantic technologies, and in particular rule-based ontology inference (based on production rules handled by the Apache Jena rule reasoner[8] or on SWRL), enable, in fact, programmatic verification of conformance between the actual design being carried and the normative references (expressed as structural properties), and supports early detection and correction of any discrepancy or violation.

Example. The *usableLamp* property is a direct structural property that aims at checking the correct configuration of *Lamps* and *Controls* inside an IDE. It is formalized by the logic predicate shown in Fig. 15, labeling a *Lamp* as *usable* if and only if the *Lamp* is controlled by a *Control* located in the same *Room* in which the *Lamp* is located, or by a *Control* located in a *Room* directly adjacent to the *Room* containing the *Lamp*.

4.6. State properties checking

State properties estimate the current, or the possible, value(s) of specific environment characteristics, given the environment structure and the current state of the installed devices, in a time-independent manner. Differently from structural

[8] More flexible with respect to SWRL due to currently available libraries, i.e., Apache Jena vs. Jess.

$$\begin{aligned}&Lamp(x) \wedge Control(y) \wedge Room(r) \wedge Room(r1) \wedge \\ &hasControl(x,y) \wedge ((isIn(x,r) \wedge isIn(y,r)) \vee \\ &(isIn(x,r) \wedge isIn(y,r1) \wedge adjacentTo(r,r1))) \\ &\Rightarrow usableLamp(x)\end{aligned}$$

Fig. 15. The Usable Lamp formalization, x, y, r, $r1$ are instances of DogOnt classes defined for a specific environment. The *Lamp(x)* notation in figure matches an instance of either the DogOnt class *Lamp* or of one of its subclasses, e.g., *DimmerLamp*. The same principle holds for *Control(y)* and *Room(r)*.

Table 2
Examples of State properties.

Domain	Name	Description
Operational	roomIllumination	A *Room* is illuminated if at least a *Lamp* located inside the same *Room* is lit or if there is an active lamp in an adjacent *Room*.
Comfort	mosquitoFree	Given the current *State* of *Doors* and *Windows*, label all *Rooms* that can be reached by mosquitoes as *not-mosquito-free* and all the remaining rooms as *mosquito-free*.
Safety	smokeFree	Given the current *State* of *SmokeSensors*, *Doors* and *Windows*, evaluate where smoke can propagate (supposing an infinite speed of propagation), and label *Rooms* reachable by smoke as *not-smoke-free* and rooms isolated from smoke as *smoke-free*.
Safety	validEscape	Given the labeling provided by the Smoke Free find all the *Rooms* located on a safe exit path, i.e. find all smoke-free *Rooms* that possess a *Window*, an *ExternalDoor* or that are connected to a valid *Room* by means of an open *Door*.

properties, state properties are usually checked on-line supporting the detection of relevant/critical configurations, possibly related to comfort, energy saving or safety policies (Table 2).

Example. The *validEscape* property (Fig. 16) is a complex rule identifying safe paths towards house exits, in case of smoke and/or fire. It is a state property based on the results of another property evaluation (*smokeFree*). It is formalized as follows: first, the *smokeFree* property of *Rooms* is evaluated, labeling *Rooms* with smoke inside, or reachable by smoke through open doors, as ¬*smokeFree*. Secondly, after labeling all remaining *Rooms* as *smokeFree*, it evaluates *Room* features to discover connections to the house exteriors either through *Windows* or *Doors*. *SmokeFree Rooms* with *Windows* (or *Doors*) and *smokeFree Rooms* connected to them by an open *Door* are labeled as belonging to a *validEscape* path. On the converse, isolated *smokeFree Rooms* and ¬*smokeFree Rooms* are labeled as ¬*validEscape*s.

5. Semantic technologies in smart grids

As introduced in Section 2, the smart grid conceptual model is easily expressed through ontologies. Nevertheless the role of semantic technologies in such a knowledge domain, also covers other requirements as pointed out by Wagner et al. [60]. In this section, the general roles previously identified (see Section 3) are mapped to the smart grid domain and a real case study is described, where a prototype for a smart grid management infrastructure was developed using Semantic Web technologies.

The smart grid domain, as stated before, is characterized by a strong demand for interoperability and standardization (role 3.1.3, Interoperability). Through the use of standard Semantic Web languages (RDF, OWL) and Linked Data principles, this demand can be satisfied. Moreover, since the smart grid is an interdisciplinary concept, ontology reuse is significantly encouraged: an holistic smart grid ontology should be able to represent both the high-level, management aspects (the new market scenario created by smart grids [61], efficiently depicted in Fig. 17) and the low-level aspects (Building Automation concepts, energy transmission networks, energy generation).

Semantic technologies also address another very common issue in smart grid development: the integration of different data sources. A typical example occurs when forecasting the energy production of Renewable Energy Sources (RES): several parameters must be taken into account, such as climate conditions, source plant location, plant type and technology. A plethora of sensors, often organized in Wireless Networks, takes care of data acquisition. However, to take informed decisions and to react to unexpected events, the management system must have not only the bare values, but also a structured knowledge of the quantities involved (roles 3.1.1 and 3.1.2, Context and Environment modeling). This is where semantic technologies come in handy.

Ontology-powered smart grid systems should be able to handle complex operation scenarios, where events must be inferred from large amounts of heterogeneous information. For example, real-time pricing of energy can be achieved through forecasting of energy production from RES and energy market simulations, using concepts from different ontologies. Another example may be to determine whether a portion of the smart grid network (a so-called microgrid) is autonomous in terms

Structural part – stage 1
$\mathit{Room}(x) \land \mathit{Room}(y) \land \mathit{Door}(d) \land \mathit{Wall}(w) \land$
$\mathit{hasWallOpening}(w, d) \land \mathit{isOpen}(d) \land$
$\mathit{hasWall}(x, w) \land \mathit{hasWall}(y, w)$
$\Rightarrow \mathit{adjacentTo}(x,y)$
$\mathit{Room}(x) \land \mathit{Door}(d) \land \mathit{Wall}(w) \land$
$\mathit{hasWallOpening}(w, d) \land \mathit{isOpen}(d) \land$
$\mathit{hasWall}(x, w)$
$\Rightarrow \neg \mathit{doorClosed}(x)$
$\mathit{Room}(x) \land \mathit{Window}(wnd) \land \mathit{Wall}(w) \land$
$\mathit{hasWallOpening}(w, wnd)$
$\Rightarrow \mathit{hasWindow}(x)$
$\mathit{Room}(x) \land \mathit{ExternalDoor}(wnd) \land \mathit{Wall}(w) \land$
$\mathit{hasWallOpening}(w, wnd)$
$\Rightarrow \mathit{hasWindow}(x)$

Structural part – stage 2
$\mathit{Room}(x) \land \mathit{notAssigned}(\mathit{doorClosed}, x)$
$\Rightarrow \mathit{doorClosed}(x)$
$\mathit{Room}(x) \land \mathit{notAssigned}(\mathit{hasWindow}, x)$
$\Rightarrow \neg \mathit{hasWindow}(x)$

Smoke Free – stage 1
$\mathit{Room}(x) \land \mathit{hasSmoke}(x)$
$\Rightarrow \neg \mathit{smokeFree}(x)$
$\mathit{Room}(x) \land \mathit{Room}(y) \land$
$\mathit{adjacentTo}(x,y) \land \neg\ \mathit{smokeFree}(y)$
$\Rightarrow \neg \mathit{smokeFree}(x)$

Smoke Free – stage 2
$\mathit{Room}(x) \land \mathit{notAssigned}(x, \mathit{smokeFree})$
$\Rightarrow \mathit{smokeFree}(x)$

Valid Escape – stage 1
$\mathit{Room}(x) \land \neg \mathit{smokeFree}(x) \land \mathit{notAssigned}(\mathit{validEscape}, x)$
$\Rightarrow \neg \mathit{validEscape}(x)$
$\mathit{Room}(x) \land \mathit{smokeFree}(x) \land$
$\mathit{notAssigned}(\mathit{validEscape}, x) \land \mathit{hasWindow}(x)$
$\Rightarrow \mathit{validEscape}(x)$
$\mathit{Room}(x) \land \mathit{validEscape}(x) \land$
$\mathit{Room}(y) \land \mathit{smokeFree}(y) \land \mathit{adjacentTo}(x, y) \land$
$\mathit{notAssigned}(\mathit{validEscape}, y)$
$\Rightarrow \mathit{validEscape}(y)$
$\mathit{Room}(x) \land \mathit{smokeFree}(x) \land \mathit{notAssigned}(\mathit{validEscape},y)$
$\neg \mathit{hasWindow}(x) \land \mathit{doorClosed}(x)$
$\Rightarrow \neg \mathit{validEscape}(x)$

Valid Escape – stage 2
$\mathit{Room}(x) \land \mathit{notAssigned}(\mathit{validEscape}, x)$
$\Rightarrow \neg \mathit{validEscape}(x)$

Fig. 16. The *validEscape* formalization.

of energy generation and consumption, in order to minimize exchanges of energy with the whole grid (roles 3.1.4 and 3.1.6, Off-line and On-line reasoning).

5.1. Leaf Islands, an industrial case study

In order to experiment all the previously described synergies, a prototype software solution for microgrids management was developed and called Leaf Islands. The Leaf Islands project was carried out in partnership with Loccioni Group, an Italian mid-sized company, with the aim of developing an energy management software system for a pioneering project called "Leaf Community", a sustainable community encompassing houses, offices and renewable energy power plants. The prototype is powered by the Leaf Ontology (in OWL), which comprises both smart grid reference models and Building Automation concepts. The goal of this solution is to assess the energetic independence of a microgrid and eventually acting on the appropriate devices to enforce it, thus minimizing energy exchanges with the bulk energy supply. This system allows to design high-level strategies for energy efficiency, starting from the available metering information, and to enforce those strategies through the control devices available in the network, regardless of their particular technology and communication protocols.

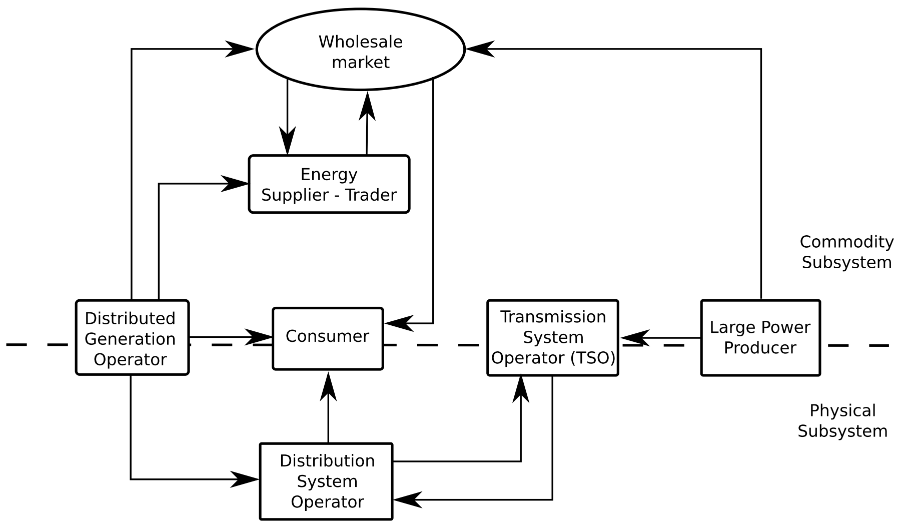

Fig. 17. Overview of transactions within the electricity market.

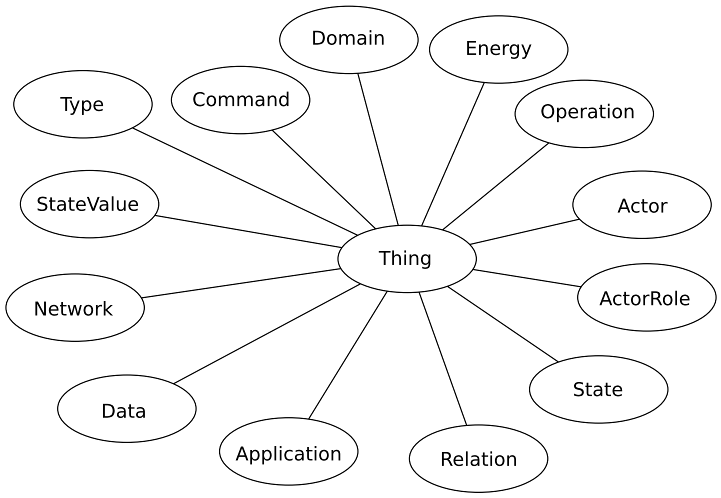

Fig. 18. Leaf Ontology: main classes and concepts.

5.2. *Leaf Ontology engineering process*

The Leaf Ontology is the final outcome of an ontology-engineering process strongly based on reuse. We combined the Building Automation concepts provided by the DogOnt ontology (see Section 4) with the GridPedia ontology [62].

GridPedia is a collaborative effort, started and maintained by the Karlsruhe Institute of Technology, to develop an ontology for the smart grid domain. Its main sources of information are the NIST Framework and Roadmap for smart grid Interoperability [26] and the Common Information Model [28,27]. Through the GridPedia classes and relationships, we are able to express the complex relationships among smart grid stakeholders and actors in technical, economical and organizational terms. Various usage examples of this ontology are directly available from its website.[9]

GridPedia has been integrated with DogOnt by linking the concepts in a common ad meaningful hierarchy: for example, the *BuildingEnvironment* class in DogOnt has been declared as a subclass of the *Actor* of GridPedia, and all the *Controllable* devices of DogOnt have been declared as subclasses of the *Device* class of GridPedia.

The final result of the integration process is the Leaf Ontology, which is summarized in Fig. 18.

For a better understanding of how the ontology integration works, consider this simple usage scenario involving two Actors of a smart grid: a house and a photovoltaic roof, respectively. These two actors have different roles: the house is

[9] http://gridpedia.org/wiki/Help:Ontology_Usage, last visited February 11, 2013.

```
:LeafCommunity a gridpedia:Smart_Grid .
:LeafHouse a dogont:Building .
:LeafRoof a dogont:Building .
:LeafHouse gridpedia:actorNetwork :LeafCommunity .
:LeafRoof gridpedia:actorNetwork :LeafCommunity .

:LeafHouse gridpedia:role :DefaultEnergyConsumer .
:LeafRoof gridpedia:role :DefaultEnergyProvider .
:LeafRoofPowerPlant a gridpedia:SolarPowerPlant .
:LeafRoof dogont:contains :LeafRoofPowerPlant .
:LeafRoof gridpedia:powerFlow :LeafHouse .
```

Fig. 19. The OWL representation of a usage scenario for the Leaf Ontology.

Table 3
Mapping of IEC 61850 to RESTful services.

HTTP method	Corresponding function	IEC 61850 reference
GET	The node returns its self-description in RDF/XML format.	Self-description
POST	The node inserts the resource contained in the request body inside its ontology A-Box	Peer-to-peer data exchange
GET	The node modifies its configuration with the data contained in the request body.	Online configuration

an energy consumer, while the roof is an energy provider. Both actors are linked through a bidirectional power flow. The semantic definition of this scenario is provided in Fig. 19.

5.3. Prototype description

The prototype was developed as a distributed Java application. The application runs at two different levels: a **node** level, where we identify a node as a physical building connected to our power grid, and a **grid** level. Both levels refer to the same ontology as concepts (T-Box) but differ in terms of individuals and assertions (A-Box).

Our definition of **node** resembles the definition of *substation* given in the IEC 61850 standard [31]. More appropriately, we identify each node with a *BuildingEnvironment*, thus an *Actor* of our smart grid system containing a certain number of *Devices*. Each local version of our prototype has a complete knowledge of the devices present in the node itself, expressed through the Building Automation concepts of DogOnt. Moreover, it is able to interact with, and to control, the devices, regardless of their technology and communication protocols, via an appropriate lower-level framework (developed by Loccioni). This enables every node in the network to enact local strategies, according to its knowledge of the network. An example of local strategy for a residential building might be the optimization of appliance usage through appropriate scheduling based on variable energy tariffs.

The **grid** level of our application, instead, is able to aggregate information from different nodes, in order to enact network-level strategies. This component is supported by a persistent version of the ontology, containing brief, aggregated description of the nodes in the network, useful to determine relevant information for the decision making process. Example of network-level decisions may include logic topology modifications, with the creation of islands composed by one or more nodes, or predictive RES production management through external applications.

5.4. Communication between nodes

Each node in the network exchanges semantic information with the other nodes, by means of REST-based Web Services. These Web Services provide a mapping of the following services, as described in the IEC 61850 [31]: peer-to-peer data exchange, self-description, and online configuration. The REST mapping was inspired by the work of Pedersen et al. [63] and the actual mapping is described in Table 3.

Through this simple communication schema, the nodes exchange semantic information and accordingly modify their inner state. Moreover, at the grid level, a more advanced component is able to query the description of the nodes and take more informed decisions, specifying different constraints for the local control logic of the single nodes.

5.5. Control logic

The control logic of the prototype is implemented in the Java language. The Java classes representing the concepts expressed in the Leaf Ontology are automatically generated from the ontology itself into a single Java package. When accessing those classes, it is possible to reason with high-level, real-world elements such as buildings, devices, energy quantities and power flows. This allows to express very complex rules through very simple and natural Java instructions. The translation of these instructions into low-level commands to enact the corresponding control actions is delegated to the underlying framework. This approach enables programmers to immediately translate and implement the specifications provided by the

```
for (actor:leafCommunity.listActors())
{
    if(actor.is(EnergyProvider))
        leafHouse.addPowerFlow(actor);
}
```

Fig. 20. The Java code for implementing a simple high-level strategy.

Table 4
Summary of roles played by semantic technologies, and obtained advantages/solved issues in the 2 presented use cases.

Role	Smart environment use case	Smart grid use case
Context modeling	• Not covered	• Grid-level modeling to support decision-making through aggregation of local information
Environment modeling	• Technology-independent reference model for smart environments • High-level, common API for interacting with smart environment devices (responds to the application layer abstraction needs)	• Technology-independent reference model for smart environments • Interaction with a lower-level framework controlling devices and appliances, to enact local strategies for each node in the grid
Interoperability	• Technology-independent reference model for smart environments (addresses a currently open problem as witnessed by current standardization efforts) • Automatic interoperation by means of off-line reasoning and rule-generation (solves low-level interoperation issues, abstracting from technology details) • Interoperability between currently isolated technologies (solves a still open issue affecting most commercial systems)	• Conceptual model referencing most of the efforts done to ensure Smart Grid interoperability [26,28,27] • Standard-based communication services between nodes • Usage of RESTful technologies
Off-line reasoning	• Design-time validation of architectural constraints (currently done manually, at least partially) • Design-time validation of norms and regulations (e.g., safe exits) (currently done manually) • Design time validation of the smart environment behavior through simulation (seldom addressable with current commercial solutions)	• Design-time planning of loads and energy distribution (currently simulated) • Explicit modeling of legal/economical aspects (energy supply contracts, energy tariffs, etc.) allowing market analyses and customer profiling operations
Code generation	• Coherence between the semantic environment model and gateway implementation (hard to keep with traditional manual development) • Improvements in the development process both in required time (seconds vs. days) and code quality	• Allows domain experts and energy managers to be involved during development of complex strategies, through the usage of high-level languages
On-line reasoning	• Integration/Merge of different models, allowing to tackle several aspects of the smart environment operation/management	• Aggregation of local nodes with grid-level models • Encapsulation of local node details with respect to grid-level decision-making

energy managers. A simple example of energy strategy is given in Fig. 20, stating that the `leafHouse` node should connect to every energy provider available in the network.

6. Discussion

The adoption of semantic technologies in the smart environment and smart grid domains is fostering relevant changes in modeling and programming techniques typically adopted in these fields. New roles and new solutions based on such techniques are emerging on a daily basis and an increasing interest in their application is pervasively emerging from the current literature. The discussed case studies, whose results are summarized in Table 4, provided an hint on how wide could be the impact of semantic solutions in such domain and on which issues can be addressed by exploiting semantics and related technologies along the whole design and implementation work-flow of smart environments and grids. Covered issues range from support to modeling activities, to development automation, coherence enforcement and inter-operation between different solutions/technologies.

Recent initiatives promoted by the European Commission (EC),[10] ETSI, CENELEC, and the main stakeholders involved in energy-related businesses, confirm this vision. The European Commission, in particular, is promoting interoperability between energy using and producing devices (EUPD) in the home domain[11] and is supporting (through the Adapt4EE project) the definition of standard vocabularies (ontologies) for Building Information Models.

[10] http://ec.europa.eu/information_society/activities/sustainable_growth/index_en.htm, last visited March 31, 2013.
[11] As in the 2nd WORKSHOP OF M2M SEMANTICS FOR SMART EEAPPLIANCES, Bruxelles, 2013.

More specifically, the European Commission is coordinating workshops between main market stakeholders with the aim of defining a shared, semantic, interoperability layer for energy using and producing devices enabling a minimum core of inter-operation between smart appliances, energy management systems and the smart grid. This effort builds upon current standardization activities carried in the context of the EU mandate M490 by CENELEC (TC205 WG18, TC57 WG17), and is strictly connected to current ETSI initiatives with the intent of promoting real interoperability without hampering current standardization efforts and avoiding proliferation of conflicting standards and guidelines. At the same time, the EC is working on a European-level labeling scheme (integrated with the energy labeling effort) aimed at providing a normative reference to smart appliance interoperability, with a focus on energy information. A preparatory study is currently being carried to:

(a) define essential features of smart appliances,
(b) make sure that smart appliances have access to necessary information (smart metering and pricing information),
(c) support interoperability,
(d) make sure that "smartness" does not bring unreasonable energy consumption (concerns smart meters + smart appliances),
(e) create the basis for incentives for smartness and/or interoperability through the Energy Label.

Also in this case, semantic technologies are considered crucial to define a shared, formal and sound support to interoperability between smart appliances.

Parallel to the current EC efforts on semantics-based interoperability, the Adapt4EE European project is building the foundations for a European-level semantic model of building-related informations. According to the project summary, Adapt4EE aims to develop and validate an holistic energy performance evaluation framework that incorporates architectural metadata (BIM), critical business processes (BPM) and consequent occupant behavior patterns, enterprise assets and respective operations as well as overall environmental conditions. The proposed approach is strongly exploiting semantics, as highlighted in the project deployment statement: "Adapt4EE will develop an *enhanced semantic enterprise model* that treats, learns and manages the enterprise environment as an intelligent agent, perceives environmental state using multi-type sensors and information modalities."

Thanks to the organization of so-called vocabulary camps (VoCamp), the project is systematically tackling the representation of building information through semantics including: Energy Efficiency in Buildings (1st VoCamp), Building Information Models (BIM, 2nd VoCamp), Energy using and producing devices (3rd VoCamp, held in June 2013), etc. This will likely provide a new driving force to semantics adoption and to semantics-based programming techniques in the context of smart environments and grids, both at the European and at the global level.

7. Conclusions

Semantic technologies are currently fostering new programming paradigms and new design methodologies in almost all domains. Thanks to their peculiar features they are currently seen as reliable and consistent means to address several design challenges, from modeling issues to run-time operation. In this paper we provided an overview of typical application cases in the smart environments and smart grids domain. While providing a top-down characterization of roles covered by these technologies, we provided a strong application-oriented view of the advantages and issues that can be effectively tackled through semantics. Several research challenges are still open, including:

(a) efficiency at run-time in terms of complexity and response times,
(b) application of Linked Open Data solutions to smart environments, e.g., for tackling interoperability problems and data sharing,
(c) validation and verification of achieved behaviors with respect to modeling, and design, assumptions, exploiting semantics (e.g., by using reasoning and/or SPARQL querying).

Nevertheless, application of semantic technologies in both smart environment and smart grid domains is increasingly playing a pivotal role as shown by European and international initiatives involving these themes.

We believe that semantics-based solutions will drive the change in smart environments and smart grids in the next years, and we envision an even more pervasive adoption if complexity issues traditionally involved in semantics handling could be overcome.

Rather than providing a final remark, we'd like to conclude the discussion with an open "call to arms" for the semantic community: the technology is now mature enough to support complex domains and to solve remarkable issues, however current issues on performance, quantitative reasoning, effective querying and inference need to be successfully tackled to move to the next generation semantic-based systems.

Acknowledgements

The authors of this contribution would like to thank Loccioni Group and in particular the MyLeaf R&D team for their support during the development of the prototype described in Section 5.

References

[1] C. Grimm, D. Bonino, Towards standardization of M2M communication in smart appliances, in: ICT for Sustainable Places, 2013, p. 9.
[2] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Sci. Am. 284 (5) (2001) 34–43.
[3] H. Chen, T. Finin, A. Joshi, An ontology for context-aware pervasive computing environments, Knowl. Eng. Rev. 18 (3) (2003) 197–207.
[4] A. Ranganathan, R.E. McGrath, R.H. Campbell, M.D. Mickunas, Use of ontologies in a pervasive computing environment, Knowl. Eng. Rev. 18 (3) (2003) 209–220.
[5] D. Preuveneers, J.V. den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, K.D. Bosschere, Towards an extensible context ontology for Ambient Intelligence, in: Second European Symposium on Ambient Intelligence, in: Lect. Notes Comput. Sci., vol. 3295, Springer, Eindhoven, The Netherlands, 2004, pp. 148–159.
[6] T. Gu, H. Pung, D. Zhang, Toward an OSGi-based infrastructure for context-aware applications, IEEE Pervasive Comput. 3 (4) (2004) 66–74.
[7] H. Chen, T. Finin, A. Joshi, The SOUPA ontology for pervasive computing, in: V. Tamma, S. Cranefield, T. Finin, S. Willmott (Eds.), Ontologies for Agents: Theory and Experiences, Whitestein Series in Software Agent Technologies, Birkhäuser, Basel, 2005, pp. 233–258.
[8] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, L.A. Stein, M. Dean, G. Schreiber, OWL web ontology language reference, W3C recommendation, 2004.
[9] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, K. Nahrstedt, Gaia: a middleware platform for active spaces, ACM SIGMOBILE Mob. Comput. Commun. Rev. 6 (4) (2002) 65–67.
[10] D.J. Cook, J.C. Augusto, V.R. Jakkula, Review: Ambient intelligence: Technologies, applications, and opportunities, Pervasive Mob. Comput. 5 (4) (2009) 277–298.
[11] L. Chen, C. Nugent, M. Mulvenna, D. Finlay, X. Hong, Semantic smart homes: Towards knowledge rich assisted living environments, in: S. McClean, P. Millard, E. El-Darzi, C. Nugent (Eds.), Intelligent Patient Management, in: Stud. Comput. Intell., vol. 189, Springer, Berlin, Heidelberg, 2009, pp. 279–296.
[12] L. Sommaruga, A. Perri, F. Furfari, DomoML-env: an ontology for human home interaction, in: Proceedings of SWAP 2005, the 2nd Italian Semantic Web Workshop, Trento, Italy, December 14–16, 2005, in: CEUR Workshop Proceedings, 2005.
[13] F. Furfari, L. Sommaruga, C. Soria, R. Fresco, DomoML: the definition of a standard markup for interoperability of human home interactions, in: Proceedings of the 2nd European Union Symposium on Ambient Intelligence, EUSAI '04, ACM, New York, NY, USA, 2004, pp. 41–44.
[14] D. Bonino, F. Corno, DogOnt – Ontology Modeling for Intelligent Domotic Environments, in: A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin, K. Thirunarayan (Eds.), International Semantic Web Conference, in: Lect. Notes Comput. Sci., vol. 5318, Springer-Verlag, 2008, pp. 790–803.
[15] M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W.D. Kelsey, D.L. Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth, K. Taylor, The {SSN} ontology of the {W3C} semantic sensor network incubator group, J. Web Semant. 17 (0) (2012) 25–32.
[16] J. Ye, G. Stevenson, S. Dobson, A top-level ontology for smart environments, Pervasive Mob. Comput. 7 (3) (2011) 359–378.
[17] V. Miori, L. Tarrini, M. Manca, G. Tolomei, An open standard solution for domotic interoperability, IEEE Trans. Consum. Electron. 52 (2006) 97–103.
[18] D. Bonino, E. Castellina, F. Corno, The DOG gateway: Enabling ontology-based intelligent domotic environments, IEEE Trans. Consum. Electron. 54 (4) (2008) 1656–1664.
[19] G. Stevenson, S. Dobson, Sapphire: generating Java runtime artefacts from OWL ontologies, in: C. Salinesi, O. Pastor (Eds.), Advanced Information Systems Engineering Workshops, in: Lect. Notes Business Inform. Process., vol. 83, Springer, Berlin Heidelberg, 2011, pp. 425–436.
[20] S. Tomic, A. Fensel, M. Schwanzer, M.K. Veljovic, M. Stefanovic, Semantics for energy efficiency in smart home environments, in: Applied Semantic Technologies: Using Semantics in Intelligent Information Processing, Taylor and Francis, 2011, pp. 429–455.
[21] J.J. Vinagre Díaz, M. Wilby, A.B. Rodriguez Conzalez, J. Garcia Munoz, EEOnt: an ontological model for a unified representation of energy efficiency in buildings, Energy Build. 60 (2013) 20–27.
[22] D. Bonino, F. Corno, DoMAIns: domain-based modeling for Ambient Intelligence, Pervasive Mob. Comput. 8 (4) (2012) 614–628.
[23] P. Moreaux, F. Sartor, F. Vernier, An effective approach for home services management, in: 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2012, pp. 47–51.
[24] J. Xu, Y.-H. Lee, W.-T. Tsai, W. Li, Y.-S. Son, J.-H. Park, K.-D. Moon, Ontology-based smart home solution and service composition, in: International Conference on Embedded Software and Systems, ICESS '09, 2009, pp. 297–304.
[25] A. Katasonov, M. Palviainen, Towards ontology-driven development of applications for smart environments, in: 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010, pp. 696–701.
[26] National Institute of Standards and Technology, NIST framework and roadmap for smart grid interoperability standards, U.S. Dept. of Commerce, National Institute of Standards and Technology, Office of the National Coordinator for Smart Grid Interoperability, Gaithersburg, MD, 2010.
[27] International Electrotechnical Commission, IEC 61970: Energy management system application program interface (EMS-API) — Part 301: Common information model (CIM) base, International Electrotechnical Commission, Geneva, Switzerland, 2009, http://webstore.iec.ch/webstore/webstore.nsf/ArtNum_PK/42807.
[28] International Electrotechnical Commission, IEC 61968: Application integration at electric utilities — System interfaces for distribution management — Part 13: CIM RDF Model exchange format for distribution, International Electrotechnical Commission, Geneva, Switzerland, 2008, http://webstore.iec.ch/webstore/webstore.nsf/ArtNum_PK/41598.
[29] T. Kostic, O. Preiss, C. Frei, Towards the formal integration of two upcoming standards: IEC 61970 and IEC 61850, in: Large Engineering Systems Conference on Power Engineering, 2003, pp. 24–29.
[30] J. Hughes, Harmonization of IEC 61970, 61968, and 61850 models, Tech. rep., EPRI, 2006.
[31] International Electrotechnical Commission, IEC 61850: Communication networks and systems in substations — Part 5: Communication requirements for functions and device models, International Electrotechnical Commission, Geneva, Switzerland, 2003, http://webstore.iec.ch/webstore/webstore.nsf/ArtNum_PK/30929.
[32] R. Santodomingo, J. Rodriguez-Mondejar, M. Sanz-Bobi, Ontology matching approach to the harmonization of CIM and IEC 61850 standards, in: First IEEE International Conference on Smart Grid Communications (SmartGridComm), 2010, pp. 55–60.
[33] R. Santodomingo, J. Rodriguez-Mondejar, M. Sanz-Bobi, S. Rohjans, M. Uslar, Towards the automatic alignment of CIM and SCL ontologies, in: IEEE International Conference on Smart Grid Communications (SmartGridComm), 2011, pp. 422–427.
[34] M. Grassi, M. Nucci, F. Piazza, Towards an ontology framework for intelligent smart home management and energy saving, in: IEEE International Symposium on Industrial Electronics (ISIE), 2011, pp. 1753–1758.
[35] A. Pena, Y. Penya, Distributed semantic repositories in smart grids, in: 9th IEEE International Conference on Industrial Informatics (INDIN), 2011, pp. 721–726.
[36] G.D. Abowd, A.K. Dey, P.J. Brown, N. Davies, M. Smith, P. Steggles, Towards a better understanding of context and context-awareness, in: Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing, HUC '99, Springer-Verlag, London, UK, 1999, pp. 304–307.
[37] T. Strang, C. Linnhoff-Popien, A context modeling survey, in: UbiComp 2004 – The Sixth International Conference on Ubiquitous Computing, in: Workshop on Advanced Context Modelling, Reasoning and Management, Nottingham/England, 2004.
[38] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, D. Riboni, A survey of context modelling and reasoning techniques, Pervasive Mob. Comput. 6 (2) (2010) 161–180.

[39] X. Wang, D. Zhang, T. Gu, H. Pung, Ontology based context modeling and reasoning using OWL, in: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004, pp. 18–22.
[40] D.A. Randell, Z. Cui, A.G. Cohn, A spatial logic based on regions and connection, in: Proceedings 3rd International Conference on Knowledge Representation and Reasoning, 1992.
[41] A. Agostini, C. Bettini, D. Riboni, Hybrid reasoning in the CARE middleware for context awareness, Int. J. Web Eng. Technol. 5 (1) (2009) 3–23.
[42] D. Bonino, E. Castellina, F. Corno, Automatic domotic device interoperation, IEEE Trans. Consum. Electron. 55 (2) (2009) 499–506.
[43] G. Niezen, Ontologies for interaction: enabling serendipitous interoperability in smart environments, Ph.D. thesis, Technische Universiteit Eindhoven, September 2012.
[44] A. D'Elia, L. Roffia, G. Zamagni, F. Vergari, A. Toninelli, P. Bellavista, Smart applications for the maintenance of large buildings: How to achieve ontology-based interoperability at the information level, in: IEEE Symposium on Computers and Communications (ISCC), 2010, pp. 1–6.
[45] M. Bhatt, F. Dylla, J. Hois, Spatio-terminological inference for the design of ambient environments, in: K. Hornsby, C. Claramunt, M. Denis, G. Ligozat (Eds.), Spatial Information Theory, in: Lect. Notes Comput. Sci., vol. 5756, Springer, Berlin, Heidelberg, 2009, pp. 371–391.
[46] J. Hois, M. Bhatt, O. Kutz, Modular ontologies for architectural design, in: Proceedings of the Conference Formal Ontologies Meet Industry, IOS Press, Amsterdam, The Netherlands, 2009, pp. 66–77.
[47] D. Bonino, F. Corno, Rule-based intelligence for domotic environments, Autom. Constr. 19 (2) (2010) 183–196.
[48] A. Soylu, P. De Causmaecker, Merging model driven and ontology driven system development approaches pervasive computing perspective, in: 24th International Symposium on Computer and Information Sciences, ISCIS 2009, 2009, pp. 730–735.
[49] Y.-H. Lee, W. Li, W.-T. Tsai, Y.-S. Son, K.-D. Moon, A code generation and execution environment for service-oriented smart home solutions, in: IEEE International Conference on Service-Oriented Computing and Applications (SOCA), 2009, pp. 1–8.
[50] N. Rodriguez, A framework for context-aware applications for smart spaces, in: IEEE/IPSJ 11th International Symposium on Applications and the Internet (SAINT), 2011, pp. 218–221.
[51] S.M. Mohsin, D.R. Natalia, S. Espen, L. Johan, P. Ivan, Ontology Driven Smart Space Application Development, River Publishers, 2012, pp. 1–25 (Ch. 5).
[52] Y. Gavshin, J. Shumik, Runtime generation of robot control code from ontology file, in: Proceedings of the Second International Conference on Adaptive and Intelligent Systems, ICAIS'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 157–167.
[53] G. Bin, D. Zhang, M. Imai, An ontology-based context-aware infrastructure for smart homes, in: Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives, IGI Global, 2011, pp. 272–299.
[54] A. Munoz, J.A. Botea, J.C. Augusto, Using argumentation to understand ambiguous situations in intelligent environments, J. Ambient Intell. Smart Environ. 5 (2009) 35–42.
[55] W. Zhang, K. Hansen, Semantic web based self-management for a pervasive service middleware, in: Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO '08, 2008, pp. 245–254.
[56] P. Pellegrino, D. Bonino, F. Corno, Domotic house gateway, in: Proceedings of the 2006 ACM Symposium on Applied Computing, SAC '06, ACM, New York, NY, USA, 2006, pp. 1915–1920.
[57] ZigBee Home Automation Profile Specification, ZigBee Alliance – public specification, Oct. 2007.
[58] D. Harel, Statecharts: A visual formalism for complex systems, Sci. Comput. Program. 8 (3) (1987) 231–274.
[59] D. Bonino, F. Corno, Dogsim: A state chart simulator for domotic environments, in: Eighth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2010, pp. 61–66.
[60] A. Wagner, S. Speiser, A. Harth, Semantic web technologies for a smart energy grid: requirements and challenges, in: A. Polleres, H. Chen (Eds.), ISWC Posters Demos, CEUR Workshop Proceedings, vol. 658, CEUR-WS.org, 2010.
[61] C. Warmer, K. Kok, S. Karnouskos, A. Weidlich, D. Nestle, P. Selzam, et al., Web services for integration of smart houses in the smart grid, in: Grid-Interop – The Road to an Interoperable Grid, Denver, Colorado, USA, 2009.
[62] A. Wagner, GridPedia, an RDF ontology for the Smart Grid, http://www.gridpedia.org, 2012.
[63] A. Pedersen, E. Hauksson, P. Andersen, B. Poulsen, C. Træholt, D. Gantenbein, Facilitating a generic communication interface to distributed energy resources: mapping IEC 61850 to RESTful services, in: First IEEE International Conference on Smart Grid Communications (SmartGridComm), 2010, pp. 61–66.