Context Based Adaptation of Semantic Rules in Smart Buildings

Vikash Kumar[†]
Telecommunication Research Center Vienna (FTW), Austria
kumar@ftw.at

Anna Fensel[§]
Telecommunication Research Center Vienna (FTW), Austria
fensel@ftw.at

Peter Fröhlich
Telecommunication Research Center Vienna (FTW), Austria
froehlich@ftw.at

[†]Inland Revenue
Auckland, New Zealand
vikash.kumar@ird.govt.nz

[§]Semantic Technology Institute (STI) Innsbruck
University of Innsbruck, Austria
anna.fensel@sti2.at

ABSTRACT

This paper presents a semantic policy adaptation technique and its applications in the context of smart building setups. We study how semantic rules which are created for one set of contextual conditions are affected when one of the context parameters changes. In particular, rules for triggering alerts and monitoring appliances and their adaptation with changing contexts have been studied in detail. We then describe how this changing context triggers changes in other related context parameters. This technique has been implemented to demonstrate its feasibility, evaluated and positively accepted during trials with users of a real-life semantically empowered smart building setup.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Representations (procedural and rule-based)*; H.2.4 [**Database Management**]: Systems—*Rule-based databases*; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Semantic Technology

Keywords

Semantic Rules, Rule Adaptation, Context Management, Smart Buildings

1. INTRODUCTION

Rules are increasingly being used in semantic applications as well as in traditional IT systems to provide a formal and powerful way of representing information such as individual preferences and privacy constraints [4, 5, 6]. To make such systems more versatile, there have been many initiatives that aim to transform a rule created in one semantic format to one created in another [8, 7, 9, 10]. However, in all these cases there is no provision that allows a rule to be shared or reused with other agents of the same application which could facilitate a sharing of intelligence. This would also save any extra effort needed by the users in creating same or similar rules from scratch. The application developers also would not need to spend too much effort in adapting their rules to different installation sites and for different end users. This paper proposes a technique for an intelligent semantic policy adaptation that enables the agents of an application across different settings, context, environments, etc. to share and reuse semantic policies amongst themselves. The investigations are carried out primarily in the use case of a smart building system with live installations in two sites. The markets for energy efficiency, building automation, smart homes are growing rapidly, almost at an exponential rate in the renewable energies sector [27].The topic of this research, to the best of our knowledge has not been pursued before. Considering that our work is among the initial steps in this direction, we start from a more scenario specific experimentation and validation of our technique. However, the foundations of the approach itself lies in the concept of context interdependency that has been studied and explained in a use case independent manner. Further validation of this technique in the use case of location aware advertising for mobile users can be found in [3] where user policies have been adapted based on this method.

Research Question: The primary research questions addressed in this paper are:

1. How dependent are semantic policies on the contextual conditions in which they operate?
2. Which context parameters are included in a typical policy?
3. How dependent are those context parameters on each other?

The main contribution of this paper is to study the extent of automated adaptation that can be achieved in semantic rules and policies triggered by a change in one of the adjoining context parameters in which the rule was defined. A technique for adaptation of policies is proposed and applied

iiWAS2013, 2-4 December, 2013, Vienna, Austria.

in the use case of two real setups for energy efficient smart building where semantic rules are used for triggering alerts and monitoring of idle devices.

Since a large part of our automation is triggered by semantic rules, we also evaluate how much adaptation is required in these automation rules for installing such a system in different site. The adaptations required when the context of an installation changes will also be discussed in detail. The contributions of this paper can be summarized as:

i) Application of semantic rules in two real smart building installations;

ii) A policy adaptation technique on the basis of context interdependency;

iii) Application of context interdependency for adapting building automation rules across different contexts and evaluations validating its acceptance.

This paper is structured as follows. In Section 2 we present a review of the related work in rule interchange and home automation, in Section 3 we present our technique towards context based adaptation of semantic policies. The details of smart building setup is described in Section 4, followed by evaluations in Section 5 and conclusions in Section 6.

2. RELATED WORK

We present the related work discussion first in the context of rule interchange and then for home automation techniques.

2.1 State of the art in Rule Interchange

There has been considerable interest in the areas of rule interchange and profile matching in the semantic web community. Some of the most notable ones are mentioned below:

The RuleML (Rule Markup Language) [32] initiative was first such example of efforts in the direction of creating a unifying family of XML-serialized rule languages that included all the Web rules. The aim was to create inter operation bridges between the common rule languages. A discussion about the importance of Rule languages on the web and the problems and opportunities of exchanging rules in a common standardized format is described in [7]. It also provides interesting discussions into the works of the W3C working group on the Rule Interchange Format, its results, use cases and future work directions.

The Rule Interchange Format(RIF) [10, 31] is a W3C recommendation aimed at producing a core rule language using which rules can be represented across all systems. The RIF framework for rule-based languages consists of a set of *dialects* which formally describes information about the syntax, semantics and XML serialization. Existing and upcoming rule languages can be mapped into this format which serves as an interlingua for mapping rules from one application to be shared, published and re-used in other applications.

An interesting related work on semantics-enabled layered policy architecture ("policy layer cake") has been proposed in [9]. This architecture is aimed at facilitating the exchange and management of policies created in multiple languages across the web. The proposed architecture consists of four layers: Unifying Logic layer, Policy Interchange Format (PIF) layer, Privacy Protection/DRM layer and Domain Specific applications layer. The architecture has been proposed as an extension of W3C's Semantic Web architecture to enable the reuse of existing work.

While our work takes inspirations from the existing works in the direction of rule interchange and sharing, the focus is more on *application independence* of such an architecture rather than *language independence* (as done by other approaches discussed). Moreover, the unique feature of the proposed infrastructure is the *adaptation* of a policy created in one set of conditions into that in another set of conditions for the "same" application.

2.2 State of the art in Home Automation

Home automation has been defined in [16] as *"The introduction of technology within the home to enhance the quality of life of its occupants, through the provision of different services such as telehealth, multimedia entertainment and energy conservation."* There has been extensive work in making home automation systems by using different technologies and utilizing innovative techniques [12, 13, 14]. The aim of authors in [12] is to achieve a more efficient remote control and monitoring of networked enabled devices in a house. They investigate the use of ZigBee towards creating a flexible automation system. The system has however not been tested with real installations and hence the scalability cannot be ascertained. Another example of such an implementation using knowledge representation techniques for smart university application is shown in [16]. This approach is based on semantic web service middleware enriched with capabilities like dynamic composition adapted specifically for university buildings and similar educational facilities. Such a system is therefore far from generic to be implemented across the board in all kinds of buildings. Besides this, expansions in such installations are themselves constantly vulnerable to changes like appliance manufacturers, types of energy used in different buildings, etc. notwithstanding the changes required in the user interfaces [15]. After an initial study into such systems, we believe that our approach provides a unique mix of the qualities (mentioned in detail in [17, 20]) that can be practically used in mainstream smart building systems. By basing most of our decision based reasoning on semantic policies, and thereafter by proposing the policy adaptation technique as detailed in this paper, we ensure that even the services for such a system can be easily adapted across several installation sites and/or user preferences. This will be a great improvement over the excel based complex system described for some practical installations in [15].

3. THE POLICY ADAPTATION TECHNIQUE

Before describing context based rule adaptation, we will first revisit one of the widely accepted definitions of context in computer science. According to Dey et. al. [22]: *"Context is 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."* We have relied on this definition to define our context parameters on which rule adaptation is based. It should also be noted that all the rules considered for our experiments were production type semantic rules [1, 2] and in this paper are referred to as simply "rules" or "policies".

The following subsections are structured as follows: in Subsection 3.1 we define the specific context parameters that are used for policy adaptation. In Subsection 3.2, we explain the interdependency of the defined context parameters on each other along with respective explanations. Finally, in Subsection 3.3 we explain our technique for the context-based adaptation of a policy based on the interdependency of context parameters described before.

3.1 Adaptation specific context parameters

Even though we describe the use case specific implementation of rule adaptation in a later section, for the ease of explanation, it is useful to mention that semantic rules are used to trigger alerts in the smart building setup. This formed the basis of experimentation of the technique described below.

For our application scenario, we consider the context information with respect to sensors being its central entity. The context parameters that we specifically employ in policy adaptation are defined below:

1. Type: Type of sensor (e.g. light, heat, movement, etc.);
2. Identity: The unique identity of the sensor (its ID, name, etc.);
3. Location: The physical location of the sensor installation;
4. States: Measurement/Reading of the sensors (e.g. ON/OFF, temperature, luminosity, etc.);
5. Time: The instant at which the above observations are made.

We did not consider the parameter "People" in our study as our aim was to explore a sensor centric approach towards the context parameters and adding "People" puts another dimension to the other parameters making the whole setup quite complex. This being a relatively unexplored direction of research, we started with adaptations in simple use cases first.

3.2 Interdependence of context parameters

As mentioned in the Introduction, one of the main contributions of this work is to observe how the change in one of the above mentioned context parameters of a semantic policy affects the other context parameters. Using these results, we describe in the next section how we achieve an adaptation of the original policy for a situation where one of the context parameters have changed. For example, we explore a case where certain policy needs to be applied to a different location, i.e. how a change in the location context triggers changes in other context parameters.

The steps involved in policy adaptation can be enumerated as:

Step 1: Identify the context that triggers the need for adapting a rule

Step 2: Determine the context parameters that are affected as a result of the triggering context change

Step 3: Replace the variables corresponding to the changed parameters in the rule

The first step to achieve this adaptation is by finding the effect of change in one context parameter on other parameters. This interdependence of parameters is envisaged for a generic scenario, but for ease of understanding is described keeping in mind our smart building use case. The examples provided will therefore refer to sensors located in a smart building setup.

Table 1 explains how the change of one context parameter affects other parameters. The parameters in bold (non-italicized) are the ones that *definitely* get affected by a change in the corresponding *triggering context parameter*. The other parameters in the *Affected Context Parameters* column (represented in *italics*) are the ones that *may* get affected by change in the corresponding triggering context. Below we provide the semantics of how a context parameter sometimes *definitely* and sometimes *may* get affected. The explanation of why certain parameters *definitely* get affected, others get affected only *possibly* while others *do not* get affected is also explained in the following subsections.

3.2.1 Semantics of *definite* and *possible* change of parameters

"Parameter P gets *definitely* affected by change in another context P_1" $\rightarrow$ there is NO situation where the value of P_1 changes and the value P remains unchanged.

"Parameter P *may* get affected by P_1" $\rightarrow$ there is at least one occurrence of a situation where each of the following statements are respectively true:

1. a change in value of P_1 causes a change in value of P,
2. a change in value of P_1 DOES NOT cause a change in value of P,

"Parameter P *does not* get affected by P_1" $\rightarrow$ there is NO occurence of a situation where change in value of P_1 causes a change in value of P.

3.2.2 Location

When the location of application of the policy changes, it is natural that the rule will now be applied on a new set of sensors specific to that location. Considering the fact that each sensor has a unique ID and one sensor cannot be at two places simultaneously, a change in location also means that the involved sensor ID also changes. Also, we keep the *Type* of the sensor unchanged as changing this might change the whole meaning of the rule. For example, in Rule 1, replacing the readings of temperature sensor by another sensor type e.g. light would change the complete meaning of the rule. Similarly, again taking the example of Rule 1, the time ($17-23$ hours) or state (airconditioner ON) do not necessarily need to change if the location changes. However, the set of constraints on state and time are open to change as well without affecting the core meaning of the rule as shown in Rule 2

$$\text{IF } X \text{ is TemperatureSensor} \wedge X \text{ isLocatedAt } L \wedge X \text{ reads } Y \wedge Y \leq 23°C \wedge \text{currentTime is } T \wedge 17:00 \leq T \leq 23:00 \rightarrow \text{Turn the Airconditioner } AC \text{ ON} \quad (1)$$

Triggering Context	Affected Context Parameters	Un-Affected Context Parameters
Location	**ID**, *Time, State*	Type
Time	*ID, State*	Location, Type
Identity (ID)	*Location, Time, State*	Type
Type	**State, ID**, *Location, Time*	
State	*Time, ID, Location, State*	Type

Table 1: Context Interdependence

$$\text{IF } X' \text{ is TemperatureSensor} \wedge X' \text{ isLocatedAt } L' \wedge X' \text{ reads } Y' \wedge Y' \leq 23.5^{\circ}C \wedge \text{currentTime is } T' \wedge 17:45 \leq T' \leq 23:01 \rightarrow \text{Turn the Airconditioner } AC \text{ ON} \quad (2)$$

3.2.3 Time

When the time parameter associated with a constraint in a rule changes, that alone may be enough for transformation of the policy without affecting other context. Again taking the example of Rule 1, we can see that a change in the time parameter can take place in the policy keeping all other context parameters unaffected and at the same time preserving the semantic meaning of the policy. So, another user can in effect want the same Airconditioner (same ID) located at the same place to be switched ON between 19-23 hours instead of the 17-23 mentioned. Similarly, the *ID* and *State* parameters also do not *necessarily* depend on Time. The argument for *Type* being an unaffected context parameter remains the same as in case of Location explained above. The parameter *Location* remains unaffected because a change in Location causes a *definite* change in *ID* as described above and we want to avoid cyclic dependencies as explained later in Section 3.3.

3.2.4 Identity(ID)

When the identity of a sensor associated with a policy changes, the new sensor with the new ID can be located in the same location (e.g. room) and the policy can still be valid for the set of constraints involving same *time, state* and *type* of constraints as already contained in the rule. This can be verified again from Rule 1 where changing the ID of temperature sensor from X to X' does not necessitate any change in other parameters. So no parameter *definitely* gets affected by change in ID. Looking back to Rule 1 we can see that the parameters for *location, time* and *state* can still be changed while maintaining the semantic meaning. Rule 3 presents an example where the change in ID in a rule leads to change in other context parameters (except *type*) and the inherent meaning of the rule is preserved. The reasoning for *Type* being an unaffected context parameter remains the same as in case of Location above.

$$\text{IF } X' \text{ is TemperatureSensor} \wedge X' \text{ isLocatedAt } L' \wedge X' \text{ reads } Y' \wedge Y' \leq 28^{\circ}C \wedge \text{currentTime is } T' \wedge 19:00 \leq T' \leq 22:00 \rightarrow \text{Turn the Airconditioner } AC \text{ ON} \quad (3)$$

3.2.5 Type

When a change in the *type* of a sensor is the trigger, then the parameters *State* and *Identity* also *definitely* change along with the type. For example, if a light sensor is changed to heat sensor, its *unique* ID is bound to change. Also, the states of heat sensor (°C) will not be the same as those of light sensor (lux) even though the plain numerical readings themselves might look similar (e.g. similar looking float numbers 39.1 (°C) and 44.4 (lux)). *Location* and *Time* are the parameters that *may* also get changed triggered by a changing *Type* of sensor. For example, in Rule 4, the location and time also change and the rule effectively represents the same *semantic* policy (as Rule 1) even with the change.

$$\text{IF } X' \text{ is HumiditySensor} \wedge X' \text{ isLocatedAt } L' \wedge X' \text{ reads } Y' \wedge Y' \leq 50\% \wedge \text{currentTime is } T' \wedge 16:00 \leq T' \leq 23:05 \rightarrow \text{Turn the Airconditioner } AC \text{ ON} \quad (4)$$

Another point of argument in Rule 4 could be that its meaning has changed with the change in *Type* parameter of the sensor (i.e. by replacing heat sensor by humidity sensor). While this may be true depending on interpretation, it should be noted that we are observing the rule in context of the trigger of *Type* change of a sensor. To avoid ambiguity based on interpretation, for this case, we take the rule *after* the *Type* of sensor (the triggering context) has been changed. The reference rule being adapted will therefore change from Rule 1 to Rule 5. Although this rule might make no syntactic sense (e.g. HumiditySensors do not give readings in °C), one can safely assume it as a reference for adapting the policy to eventually what it looks like in Rule 4.

$$\text{IF } X' \text{ is HumiditySensor} \wedge X' \text{ isLocatedAt } L \wedge X' \text{ reads } Y \wedge Y \leq 23^{\circ}C \wedge \text{currentTime is } T \wedge 17 \leq T \leq 23 \rightarrow \text{Turn the Airconditioner } AC \text{ ON} \quad (5)$$

3.2.6 State

When change in *state* of a rule is the trigger for rule adaptation, it can be argued that none of the other parameters *necessarily* need to change as a result of change in state. Consider the case of Rule 1 adapted to a situation where only the state of the triggering sensor changes. In such a situation, only the temperature constraints of the temperature sensor will change and the rule turns into Rule 6.

$$\text{IF } X \text{ is TemperatureSensor} \wedge X \text{ isLocatedAt } L \wedge X \text{ reads } Y' \wedge Y' \leq 27^{\circ}C \wedge \text{currentTime is } T \wedge 17 \leq T \leq 23 \rightarrow \text{Turn the Airconditioner } AC \text{ ON} \quad (6)$$

As can be seen in this adapted rule, none of the other parameters need to be *necessarily* affected to maintain the semantic and syntactic conformance with the original rule.

However, there *might* be situations where other parameters also need to be changed when triggered by a change in *state*. These situations might affect the other three context parameters namely *Time, Identity* and *Location* as shown in Rule 7.

$$\text{IF } X' \text{ is TemperatureSensor } \wedge\ X' \text{ isLocatedAt } L' \wedge \\ X' \text{ reads } Y' \wedge\ Y' \leq 27°C \wedge \text{ currentTime is } T' \wedge \\ 19:30 \leq\ T'\ \leq 22:30 \rightarrow \text{ Turn the Airconditioner } AC \text{ ON} \quad (7)$$

The respective parameter for the *Type* of sensor is a special case because the parameter *State* of a sensor itself is a property of the *Type* of the sensor. For example, the *State* of temperature sensor is measured in $°C$ while that of a humidity sensor (another sensor *Type*) is measured in percent (%). A change in the *State* of the sensor can therefore, never trigger a change in the *Type* of the sensor in a rule because this changes the semantics (and probably the syntax) of the whole rule as shown in example Rule 8 which is both semantically and syntactically incorrect.

$$\text{IF } X' \text{ is HimiditySensor } \wedge X' \text{ isLocatedAt } L' \wedge X' \text{ reads } Y' \\ \wedge\ Y' \leq 27°C \wedge \text{ currentTime is } T' \wedge 19:30 \leq\ T'\ \leq 22:30 \\ \rightarrow \text{ Turn the Airconditioner } AC \text{ ON} \quad (8)$$

3.3 Rules for policy adaptation

The main actions of the rules created for smart building setup were:

1. Changing the status of an appliance from IDLE to OFF or from IDLE to ON based on certain conditions,

2. Checking a combination of parameters and decide whether to send an alert if some criteria are met or do not send an alert otherwise.

These policies fall in the category of "Device dependent" and "Threshold dependent" policies respectively for Smart Homes as defined in [21]. The rules to be followed when choosing which corresponding parameter should be substituted first are enumerated below:

1. The parameter that *definitely* affect the most number of parameters has the highest priority,

2. The parameters that *potentially* affects other parameters comes afterwards,

3. Only the parameters triggered by initial parameter change gets affected, i.e. there is no cyclic interdependency between parameter changes.

The protocol to be followed when rule adaptation is triggered by each of the context parameters is described in detail in the flow diagram of Figure 1.

4. DETAILED SETUP DESCRIPTION

While the general architecture of the Policy Adaptation Framework is shown in Figure 2, in the following subsections we describe the hardware and semantic software parts of the implemented smart energy efficient building deployment.

4.1 Hardware - Physical Layer Settings

The installation was carried out on two separate locations and different types of buildings [20]. The installation at a school in Kirchdorf, Austria, consists of sensors, two smart meters for measuring consumption of electrical energy in three classrooms, plugs for measuring electrical consumption of individual appliances, and a power management service (PMS), for monitoring and controlling the state of all computers installed in these three rooms. The rooms were named E012, E014 and E015 while the smart meters were called 1Q0 and 2Q0 as shown in a figure later. The installation at a factory floor in Chernogolovka in Russia includes only sensor devices for measuring room temperatures and temperatures of incoming and outgoing airflow of heating system in factory. In the complete version of our system, the following set up has been applied.

4.1.1 Sensors

Sensor devices for measuring temperature, humidity and light were battery powered autonomous systems pushing their data over wireless LAN internet connection to a cloud service exposing collected data via REST interface.

4.1.2 Meters

Power consumption meters were installed in the housing of already existing electricity distribution cabinets at school. Collected data was exposed over a serial interface on metering devices and transferred to persistent storage through a wireless bridge to be finally exposed via REST interface for external use.

4.1.3 Smart Plugs

Smart plugs from Plugwise [23] were used to measure the consumption of individual devices at school in Kirchdorf, consisting of one beverage machine, one coffee machine and a wall socket where pupils in school connect their laptops. The data of smart plugs is exposed over Web Service interface for external use.

4.1.4 Power Management Service (PMS)

The PMS is a service for monitoring the state of computers installed in the three classrooms as well as for shutting down and restarting them remotely.

4.1.5 Hardware Setup Overview

The whole trial setup consists of sensors, meters, plugs and PMS where all delivered data is accessed either by REST or Web Service interface processed by connector software where it is prepared for semantic storage in an OWLIM semantic repository [18].

4.2 Data and Software Layer - Semantics

An OWLIM-Lite [18] based semantic repository is central to the data layer of our smart building architecture. OWLIM is a Sesame SAIL (Storage and Inference Layer) where Sesame[24] itself is a framework for analyzing and querying RDF data stored in a triple store. This serves as the triple store for all the data received from various sensors and also provides RDFS-optimized reasoning capabilities necessary for our use cases.

The connector software itself reads data from school via REST and Web Services on the one side and prepares and stores that data in OWLIM repository by using native Java

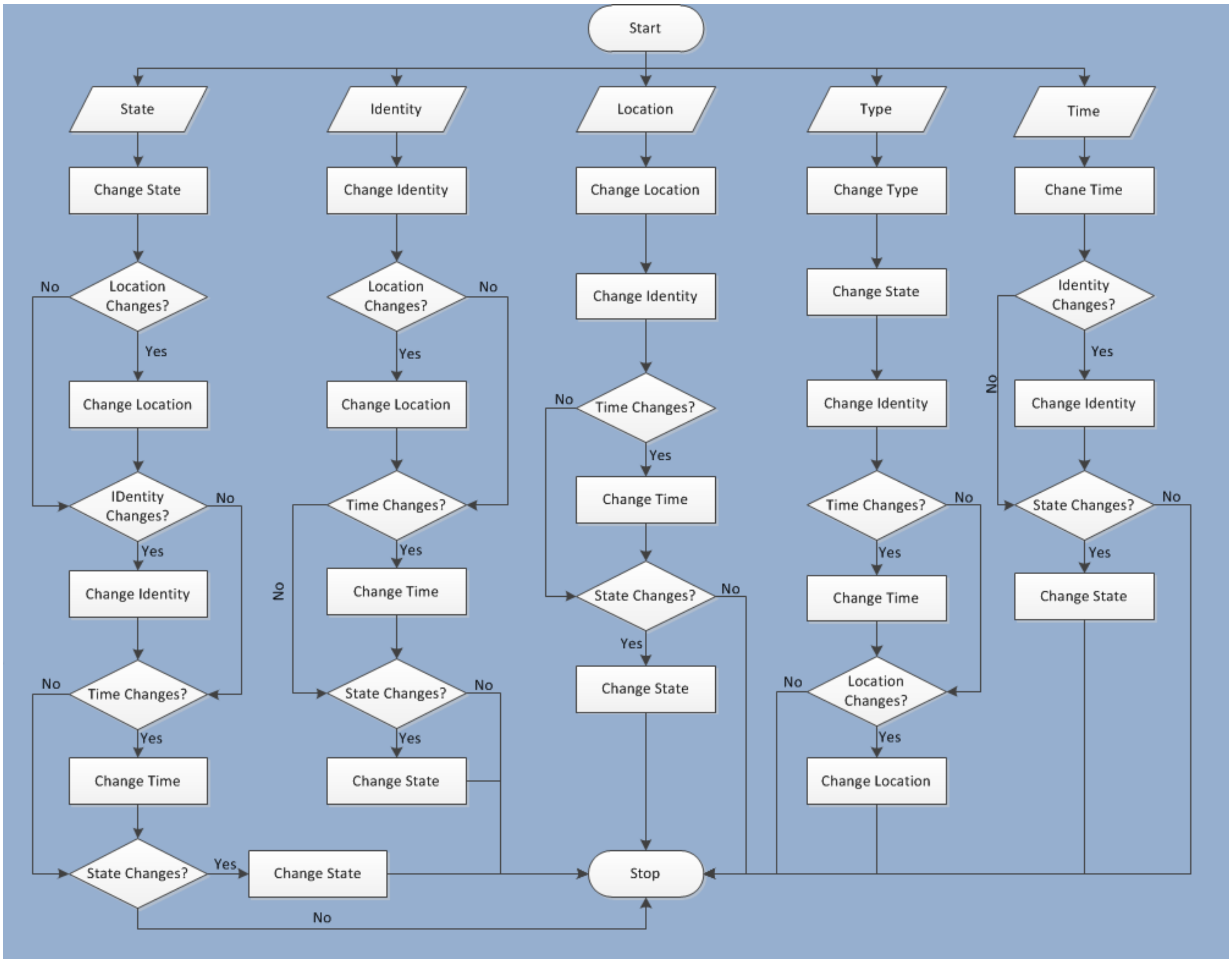


Figure 1: Flow Diagram Representing the Procedure to be Followed for each Context Trigger

Sesame 2.0 API on the other side. Besides this functionality and due to limitations of OWLIM for implementing custom rules, the connector software is doing custom reasoning over data in OWLIM store via SPARQL Construct queries implementing rules for our use cases. Client side devices like tablets and phones query the repository utilizing REST interface offered by OWLIM store. These devices contain services for the users to monitor and control energy consumption in a house.

The SESAME-S [11, 30] Smart Building ontology is publicly available for download from the CKAN directory at [29] and details of individual entities described in [20]. Quantitative details of the data collected in triple store from the school setup over a period of around 6 months (February-August 2012 with some breaks) have been provided in Table 2.

Parameter	Count
Triples	9631432
Entities	1203864
Distinct Classes	33
Distinct Predicates	34
Distinct Subject Nodes	1203868
Distinct Object Nodes	1203786
Distinct Resource URIs	1203870

Table 2: Ontology and Triple Store Details

4.2.1 Policy Based Approach Towards Intelligent Decision Making

A policy based approach towards providing personalized and flexible services to users lies at the core of our design that enables easy adaptation of the framework to different environments. With the data arriving from heterogeneous sources structured and updated in the central repository almost in real time, we utilize the feature to create complex rules and policies to administer this setup. Several services and end user interfaces were created in which rule based modeling of user side constraints were utilized in the moni-

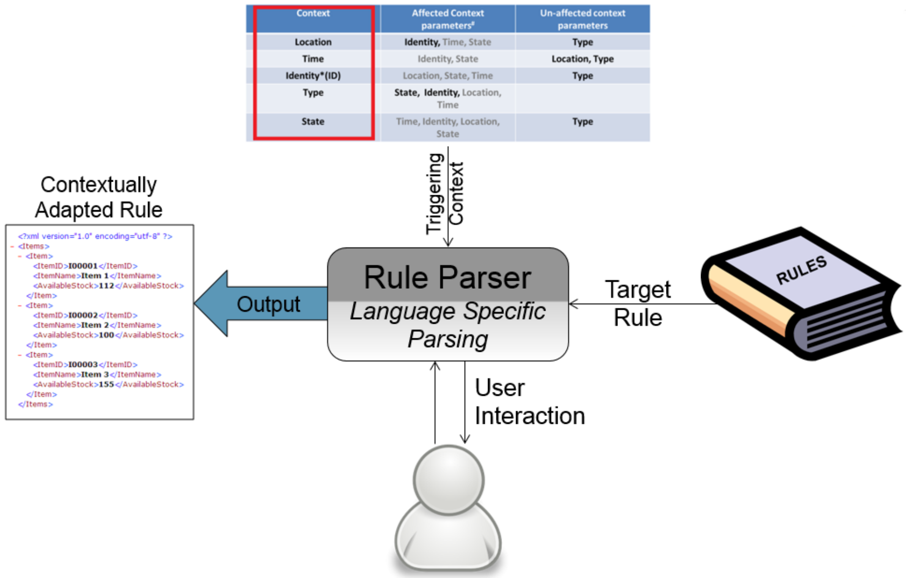


Figure 2: The Architecture of the Policy Adaptation Framework

toring and administration of the system. In the first phase of implementation, most of our tasks involved monitoring of the system and only one use case takes care of administrative jobs.

The OWLIM uses TRREE (Triple Reasoning and Rule Entailment Engine) which is a forward-chaining R-Entailment [19] for providing in-memory reasoning and query evaluation functionalities. Existing rulesets could be extended with custom rules expressed in the native PIE format. However, several limitations in this format (like not supporting mathematical operators) greatly restrict the potential advantages one could harness with custom rules.

We therefore resorted to using SPARQL [25] Construct queries as a workaround for this handicap. The end user services in our setup do a constant querying of the status of changes in repository at regular intervals. These intervals are flexible and in all our installations, fixed at 15 minutes. Each time an application sends a query to the repository, a set of SPARQL Construct queries are executed by the connector software. Each of these Construct queries contains the logic and constraints relating to the user and/or system policies. If the constraints are matched with existing environmental conditions represented in the repository, new triples pertaining to respective alerts are generated and added to the triple store. In our setup, all these triples are added to the class "`Alert`", connecting them to the device and time at which they were generated along with a message. The Application queries the "`Alert`" class each time to check if a new alert was generated in the system since the last check.

It should be noted here that Construct queries are just used as an implementation specific optimization in this setup and for another setup, its own matching rule language could be used without affecting the policy adaptation technique. The intervals of 15 minutes was also chosen after several experiments. The system was capable of having a granularity from 15 seconds and upwards, but it was observed that for optimal performance and avoiding data overload, 15 minute interval was ideal for our use case to achieve almost all intended functionalities. The policy adaptation technique itself was not dependent on collection of home data in real-time.

4.3 Policy Adaptation Implementation

The implemented architecture of the policy adaptation is shown in Figure 2. The SmartBuilding ontology is extended with the concept-context relationship, i.e. various concepts belonging to specific context parameters like location, sensor type, etc. are appropriately tagged. Rules in the code are written using parametrized SPARQL Construct queries [28]to make it easier for substituting variables corresponding to individual context parameters.

The central parser performing language specific parsing has been implemented using Jena's ARQ utility [26]. The inputs to the system are the context parameter that triggers adaptation and the target rule to be adapted. On getting the inputs, the parser decides the parameters that *definitely* get affected and the ones that *may* get affected and using the concept-context mapping in the ontology ask the user interactively for new values for respective concepts that need to be substituted. After interactively substituting the values for affected parameters, the final adapted policy is produced as a result.

The degree of automation of the overall system is very high despite the user interaction involved. Tasks like rule parsing, identification of dependent parameters, look-up of their respective values in triple store and eventual substitution in adapted rule is done automatically. The user only provides the triggering parameter and their choice/preference of values for the affected parameters.

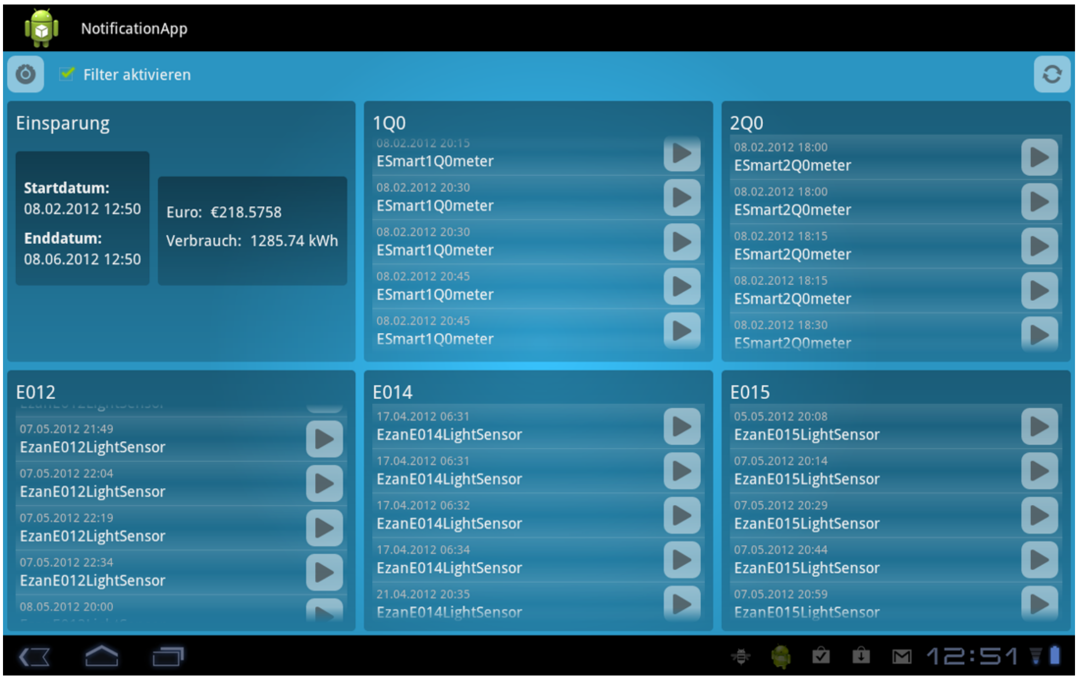


Figure 3: The Alert Monitoring App

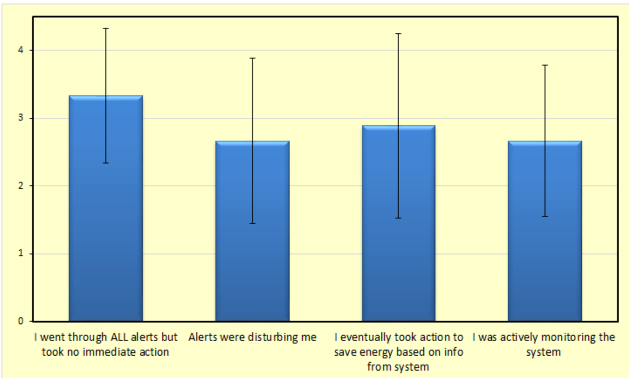


Figure 4: User Feedback on Alert Monitoring App

5. EVALUATIONS

The evaluations of the system were carried out in several phases with different sets of users and end user services using semantic policies for different purposes. We describe the evaluation outcomes of the policy adaptation incorporated in two apps for "Alert Monitoring" and "Power Management Service" respectively.

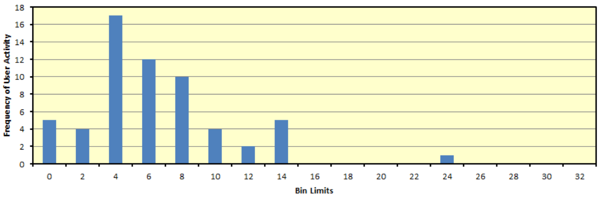
Also, the evaluations presented here do not address issues like scalability of the system, its ability to handle larger amounts of data, performance, etc. Details of the system setup and its performance based on parameters like adding new data sources from more buildings and devices and aligning the new data coming from these sources are described in [20].

We stick to the main focus of the paper which is to prove effectiveness of the policy adaptation technique and its acceptance among users by way of logging in and studying their usage data.

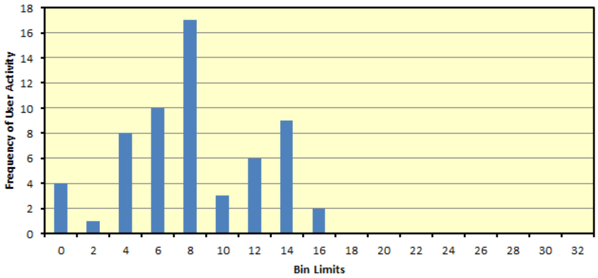
5.1 Alert Monitoring Policies

The initial setup consisted of an Alert monitoring app that uses policies for generating alerts based on anomalous circumstances. The app was installed in 2 Android based tablets and given to pupils of a class for testing during 1 week of trials in June 2012. Analysis of the data collected by a smart home installation in a school over ca. 6 months was used to come up with thresholds which were considered *normal* and beyond which we could call the situation as *anomalous*. The underlying system policies of the app were then *adapted* using our technique based on these threshold values obtained from data analysis. Figure 3 shows the app that would generate these alerts based on the thresholds provided.

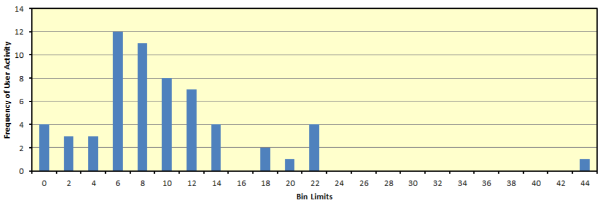
During the trial period, users could check the app to obtain the alerts generated by the system in a specific temporal duration by setting appropriate *filters* for the duration. They could further click on specific alerts and get details about the timing and source of the alerts. At the end of our one week of trials, the results in the form of user experiences were gathered through a questionnaire from the class. The results obtained as feedback from the questionnaire were compiled as shown in Figure 4. In the graph, with readings ranging from 0 to 5, value 0 represents *strongly disagree* while 5 represents *strongly agree*. The feedback shows that 9 users were using the tablet for 1 week and most of them were viewing the alerts and eventually taking action to save energy based on them. They were also fairly actively monitoring the system throughout the test period even with the limitation that only 2 tablets were shared in a class of 9 pupils. Apparently the number of alerts were too many for some users and we learned the lesson that similar alerts may be clubbed together to reduce the number that is shown eventually to users.



(a) Frequency graph for Phase 1



(b) Frequency graph for Phase 2



(c) Frequency graph for Phase 3

Figure 5: User activity as logged during the trial

5.2 Policies for Power Management Services

In another setup, several Android based smart phones and tablets were distributed among pupils and teachers of two classes in user trials conducted over a duration of 1 week for each class (i.e. a total of 2 weeks). This app was used to monitor and control the power management service installed in 3 of the computer rooms of the school (more details of the app in [20]). The app would also allow a user to change the state of the computers in computer rooms from IDLE to AWAKE or OFF. Here we provide results of some of the observations made during these trials at the school. The evaluations were made by an analysis of their actions as logged by them when using the system. Policies of the PMS were adapted afresh after every few days of evaluation. The results presented represent the user behavior alterations during those evaluation periods caused due to policy changes.

During the first week of trials (Phase 1), we presented the users with alerts based on an energy saving policy that first shows them alerts from individual computers in computer rooms. First alert from a computer is generated when it is idle for a predefined time interval T_1 minutes and another alert is generated after further T_2 minutes. If no action was taken by the user, then after waiting for another T_3 minutes, the respective computer would automatically turn off.

The policies were adapted and changed after every few days by changing the *Time* and *Location* context parameters and the respective user behavior was recorded. As shown in Figures 5 (a) and (b), the frequency of user activity remains comparable and in fact increase slightly in the second week (Phase 2) showing the compatibility of new policies with user interests. Bin limits in these graphs represent the range of frequency values in ascending order; i.e. a frequency of 4 having bin limit 2 would mean 4 is the number of observations having values more than 0 (previous limit) and less than or equal to 2 (current limit). Therefore, it can be seen in Figure 5 (b) that the frequency of the number of observations in range 3-4 is 8, 5-6 is 10 and 7-8 is 17 respectively. This can be considered an improvement over Phase 1 data represented in Figure 5 (a) where the frequency of the number of observations in range 3-4 is 17, 5-6 is 12 and with 7-8 is 10 respectively.

In the third week (Phase 3) of trials, the users were given same service installed on Android phones rather than the tablets in the previous phases. Appropriate policy adaptations based on location of computers were made and the results showed no significant negative change in user behavior as seen in Figure 5 (c).

6. CONCLUSIONS & FUTURE WORK

In this paper, we introduced the challenge of automatic adaptation of semantic policies to various contexts and proposed a technique that utilizes interdependence of context parameters towards achieving policy adaptation. With this aim, we identified a set of five context parameters in a typical policy and studied the effects of changes in each of these on other parameters respectively. This helped us in answering the research questions (2) and (3) defined in Section 1 earlier. Having described this interdependence at different levels (i.e. definite, possible or no effect) for each pair of context parameters, we explained the policy adaptation technique developed by us. We also showed how this technique was implemented in a real-life semantically powered smart building system along with details of hardware and software level setups. This helped us in answering the research question (1) defined in Section 1. In the end, we provided empirical evidence of positive feedback for this implementation from the users of various apps using these adapted policies of the smart building system.

In future, this technique can be extended to take into account multiple context parameters as a trigger. Furthermore, adding "People" as another parameter will be a very important extension for such adaptations to work pervasively. Implementation of this technique for other common rule systems like SWRL, SQWRL, RIF will also be interesting.

7. ACKNOWLEDGMENTS

The Competence Center FTW Forschungszentrum Telekommunikation Wien GmbH is funded within the program COMET - Competence Centers for Excellent Technologies by BMVIT, BMWA, and the City of Vienna. The COMET program is managed by the FFG. This work has been partially supported by the FFG programs COIN with SESAME-S project and "IKT der Zukunft" with OpenFridge project. We would like to thank the SESAME-S and OpenFridge project teams for their contributions to this work.

8. REFERENCES

[1] de Sainte Marie, C., Hallmark, G., Paschke, A., editors. RIF Production Rule Dialect. W3C Recommendation,

22 June 2010. http://www.w3.org/TR/2010/REC-rif-prd-20100622/

[2] Damasio, C. V., Alferes, J. J., Leite, J. 2010. Declarative semantics for the rule interchange format production rule dialect. In Proceedings of the 9th international semantic web conference on The semantic web - Volume Part I (ISWC'10), Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, and Lei Zhang (Eds.), Vol. Part I. Springer-Verlag, Berlin, Heidelberg, 798-813.

[3] Kumar, V., Zeiss, J., Happenhofer, M.: Rule Based Preferential Context Sharing in Location Aware Mobile Advertising. In The 10th International Conference on Mobile Web Information Systems (MobiWIS), Springer LNCS 8093: 64-78, Cyprus, 26-28 August 2013.

[4] Toninelli, A., Bradshaw, J., Kagal, L., and Montanari, R.: Rule-based and ontology-based policies: Toward a hybrid approach to control agents in pervasive environments. In Proceedings of the ISWC2005 Semantic Web and Policy Workshop, 2005.

[5] Grosof, B. N.: Representing e-business rules for the semantic web: Situated courteous logic programs in ruleml. In Proceedings of the Workshop on Information Technologies and Systems (WITS, 2001).

[6] Grosof, B. N., Poon, T. C.: Representing agent contracts with exceptions using xml rules, ontologies, and process descriptions. pages 340-349. ACM Press.

[7] Boley, H., Kifer, M., Patranjan, P. L., Polleres, A.: Rule interchange on the web. In Reasoning Web, pages 269-309, 2007.

[8] Boley, H., Osmun, T. M., Craig, B. L.: Social semantic rule sharing and querying in wellness communities. In Proceedings of the 4th Asian Conference on The Semantic Web, ASWC '09, pages 347-361, Berlin, Heidelberg, 2009. Springer-Verlag.

[9] Hu, Y. J., Boley, H.: Sempif: A semantic meta-policy interchange format for multiple web policies. In Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01, WI-IAT '10, pages 302-307, Washington, DC, USA, 2010. IEEE Computer Society.

[10] Kifer, M.: Rule interchange format: The framework. In Diego Calvanese and Georg Lausen, editors, Web Reasoning and Rule Systems, volume 5341 of Lecture Notes in Computer Science, pages 1-11. Springer Berlin / Heidelberg, 2008.

[11] Tomic, S., Fensel, A., Pellegrini, T.: Sesame demonstrator: ontologies, services and policies for energy effciency. In Proceedings of the 6th International Conference on Semantic Systems, I-SEMANTICS 2010, pages 24:1-24:4, New York, NY, USA, 2010. ACM.

[12] Gill, K., Yang, S. H., Yao, F., Lu, X.: A zigbee-based home automation system. Consumer Electronics, IEEE Transactions on , vol.55, no.2, pp.422-430, May 2009.

[13] Byun, J., Park, S.: Development of a self-adapting intelligent system for building energy saving and context-aware smart services. IEEE Transactions on Consumer Electronics. IEEE, 2011.

[14] Bromley, K., Perry, M., Webb, G.: Trends in Smart Home Systems, Connectivity and Services. www.nextwave.org.uk, 2003.

[15] Capehart, B. L., Middelkoop, T.: Handbook of Web Based Energy Information and Control Systems, 565 pages, Fairmont Press (July 26, 2011).

[16] Stavropoulos, T. G., Tsioliaridou, A., Koutitas, G., Vrakas, D., Vlahavas, I.: System architecture for a smart university building. In Proceedings of the 20th international conference on Artificial neural networks: Part III (ICANN'10), Konstantinos Diamantaras, Wlodek Duch, and Lazaros S. Iliadis (Eds.). Springer-Verlag, Berlin, Heidelberg, 477-482, 2010.

[17] Fensel, A., Tomic, S., Kumar, V., Stefanovic, M., Aleshin, S. V., Novikov, D. O.: SESAME-S: Semantic Smart Home System for Energy Efficiency. Informatik Spektrum 36(1): 46-57 (2013)

[18] Kiryakov, A., Ognyanov, D., Manov, D.: OWLIM - A pragmatic semantic repository for OWL. In Proceedings of the 2005 international conference on Web Information Systems Engineering (WISE'05), Mike Dean, Yuanbo Guo, Woochun Jun, Roland Kaschek, and Shonali Krishnaswamy (Eds.). Springer-Verlag, Berlin, Heidelberg, 182-192, 2005

[19] Horst, H. J.: Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity, volume 3729 of Lecture Notes in Computer Science, chapter 48, pages 668-684. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005

[20] Kumar, V., Fensel, A., Lazendic, G., and Lehner, U.. Semantic Policy-based Data Management for Energy Efficient Smart Buildings. In On the Move to Meaningful Internet Systems: OTM 2012 Workshops, Springer LNCS 7567, pages 272-284, Rome, Italy, September 2012.

[21] Kumar, V., Fensel, A., Tomic, S., Mayrhofer, R., Pellegrini, T.: User created machine-readable policies for energy efficiency in smart homes, in Proc. UCSE 2010, co-located with Ubicomp 2010, September 2010.

[22] Dey, A.K., Abowd, G.D., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction 16 (2-4), 97-166, 2001.

[23] Plugwise Plugs: `http://www.plugwise.com`

[24] Sesame Framework: `http://www.openrdf.org/`

[25] SPARQL Query Language: `http://www.w3.org/TR/rdf-sparql-query/`

[26] Jena ARQ: `http://jena.apache.org/documentation/query/`

[27] Clean Energy Trends 2012: `http://www.cleanedge.com/reports/clean-energy-trends-2012`

[28] Parameterized SPARQL: `http://jena.apache.org/documentation/javadoc/arq/com/hp/hpl/jena/query/ParameterizedSparqlString.html`

[29] The SESAME-S Smart Building ontology: `http://datahub.io/dataset/smartbuilding-sesames`

[30] The SESAME-S project: `http://sesame-s.ftw.at/`

[31] Rule interchange format. `www.w3.org/2005/rules/`

[32] Ruleml. `http://www.ruleml.org/`