Intelligent Environments 2016
P. Novais and S. Konomi (Eds.)

doi:10.3233/978-1-61499-690-3-33

Smart Office Automation Based on Semantic Event-Driven Rules

Sergio Muñoz [1], Antonio F. Llamas, Miguel Coronado, and Carlos A. Iglesias
Intelligent Systems Group, Universidad Politécnica de Madrid, E.T.S.I. Telecomunicación, Avda. Complutense, 30, 28040 Madrid, Spain

Abstract. The emerging Internet of Things opens endless possibilities to the contemporary notion of smart offices, where employees can benefit for automations so that the workplace can maximize employees productivity and enterprise performance. However, usually the integration of new components in smart environments is not straightforward. In this article we propose the use of a semantic vocabulary to define this automation, and an architecture consisting of a web task automation server and mobile task automation components that enable contextual services. One the one hand, the architecture exhibits flexibility to interconnect internet services and devices. On the other side, the use of semantic technologies provides semantic interoperability and expressivity for the automation definition.

Keywords. smart office, task automation, ontology, EWE, ambient intelligence

1. Introduction

The increasing adoption of Information and Communication Technologies (ICT) has transformed workplaces into networked workplaces. The application of Ambient Intelligence (AmI) principles to the workplace turns out in the notion of *smart offices*, that can be defined as "workplaces that proactively, but sensibly, support people in their daily work" [1]. Smart offices should be aligned to the business objectives of the enterprise [2], and should enable a productive environment that maximizes employee satisfaction and company performance. Thus, smart offices should manage efficiently and proactively the Internet of Things (IoT) infrastructure deployed in the workplace as well as the enterprise systems. Moreover, smart offices should be able to interact with smartphones and help employees to conciliate their personal and professional communications.

A recent shift in AmI is to move from the model of full and transparent automation to smart collaboration, since autonomous often leave users feeling out of control [3]. A popular approach to interconnect and personalise both IoT as well as Internet services is the use of Event-Condition-Action (ECA) rules, also known as trigger-action rules. A number of now prominent web sites, mobile and desktop applications feature this rule-based task automation model, such as Ifttt [2] or Zapier. These systems, so-called Task Automation Service (TAS) [4], are typically web platforms or smartphone applications, which provide an intuitive visual programming environment where unexperienced users seamlessly create and manage their own automations.

[1] Corresponding Author.
[2] http://ifttt.com and http://zapier.com

The great number of users these platforms accumulate, speak for itself in terms of usefulness. However, they suffer from two major drawbacks: (i) the only incoming data streams available are those the platform is prepared for; and, (ii) they lack of a mechanism to use and reason over large scale data outside their platform such as the Linked Open Data (LOD) cloud or context data. These shortcomings decrease TASs flexibility, narrowing rule capabilities.

Before introducing the complete architecture, lets consider the following scenario. A technological company uses the EWETasker framework to automate common processes inside corporate logistics with their clients and employees as main actors. The smart office has installed several beacons along the entire perimeter, covering every department of the enterprise. Each employee has installed the EWETasker application on his smartphone and is able to define customized automated rules using their local resource channels combined with company channels.

In order to overcome the shortcomings identified above and realize the proposed smart office scenario, this paper provides the following contributions. We propose to use a semantic vocabulary, Evented WEb (EWE), for modeling automation rules and a general task automation architecture where users can customize their automation rules. The article presents an implementation of this architecture, so-called EWETasker, that provides both a web platform as well as a mobile application for Android devices. Rule execution has been developed based on EYE inference engine that provides support for Notation3 rules. In addition, this framework also integrates external devices as channels, like Estimote Beacons, which uses Bluetooth Low Energy (BLE) technology to detect the position of the user inside the beacon range among other things.

The rest of the article is structured as follows. In the next section, we review briefly the EWE ontology [4] that provides the foundation for expressing task automation for web services and devices. Then we present the architecture of EWE Tasker, a task automation service for enabling the smart office in Sect 3. The implementation of this architecture in a smart office scenario is presented in Sect. 4. Sect. 5 discuss related work. Finally, we discuss lessons learnt and future work in Sect. 6.

2. Evented WEb (EWE)

Evented WEb (EWE) ontology [4][3] is a standardized data schema (also referred as “ontology" or “vocabulary") designed to model, in a descriptive approach, the most significant aspects of Task Automation Services. It provides a common model to define and describe TASs, representing its rules and enabling rule interoperability. In addition, it provides a base vocabulary in order to build domain specific vocabularies, and also enables the publication of raw data from TASs online (in conformity with Internet trends).

Four major classes make up the core of EWE: Channel, Event, Action and Rule, as shown in Fig. 1. In order to create new classes that are specific to particular use case scenarios in TASs, these classes may be extended [4].

First, the class *Channel* defines individuals that either generate Events, provide Actions, or both. In the smart office context, sensors and actuators such as a presence sensor or a light switch are described as channels, thus they produce events or provide actions. The class *Event* defines a particular occurrence of a process, and allows users to describe

[3] Available at http://www.gsi.dit.upm.es/ontologies/ewe/

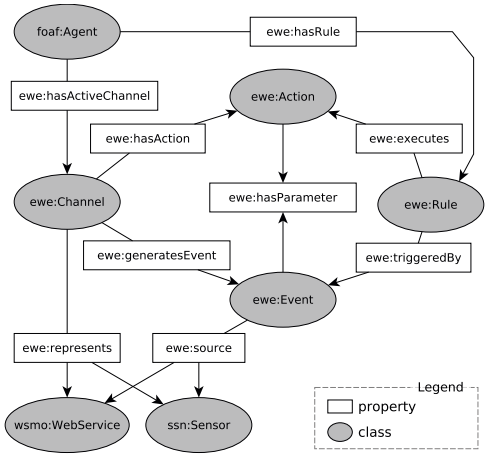

Figure 1. EWE main entities

under which conditions should Rules be triggered. Event individuals are generated by a certain Channel. The presence detection generated by the presence sensor is an example of entity that belongs to this class. The class *Action* defines an operation provided by a Channel that is triggered under some conditions. Following the smart office context mentioned above, to switch on the light is an example of Action generated by the light switch Channel. Finally, the class *Rule* defines an *Event-Condition-Action* (ECA) rule, triggered by an Event that produces the execution of an Action. An example of rule is: "*If presence is detected, then switch on the light.*". In order to model the execution logic of the rules, EWE employs the EYE reasoner.

3. EWETasker Architecture

The *smart office* scenario combines events coming from different sources, such as Internet services, smartphones, and connected devices. These services and devices are managed by channels. Channels are defined as abstractions for receiving events or emitting actions to Internet services (i.e. Twitter or GCalendar) and connected devices (i.e. device channels). These channels are registered in a channel directory provided by the TAS.

A reference TAS architecture must provide a visual rule editor for managing rules; include both web and device channels; feature channel discovery; enable multi-event, multi-action and chain rules; manage group channels and group rules; detect collisions with rules that involve group channels; and support mixed execution profile [5]. Furthermore, in a smart environment, it is essential the ability to enable easy including of new components that expand the capabilities of the system. These reference requirements have been followed in order to successfully design the EWETasker architecture.

EWETasker architecture, shown in Fig. 2, consists of two main modules: *Task Automation Server* and *Mobile Client*. The former aims to handle events and to trigger accordingly an action generated by the rule engine. The latter has as main role to handle events coming from contextual sources (such as presence or temperature sensors) or from the own device (such as battery, Wifi, or Bluetooth), and to send them to the TAS. Furthermore, both modules include several functions for managing rules. In this way,

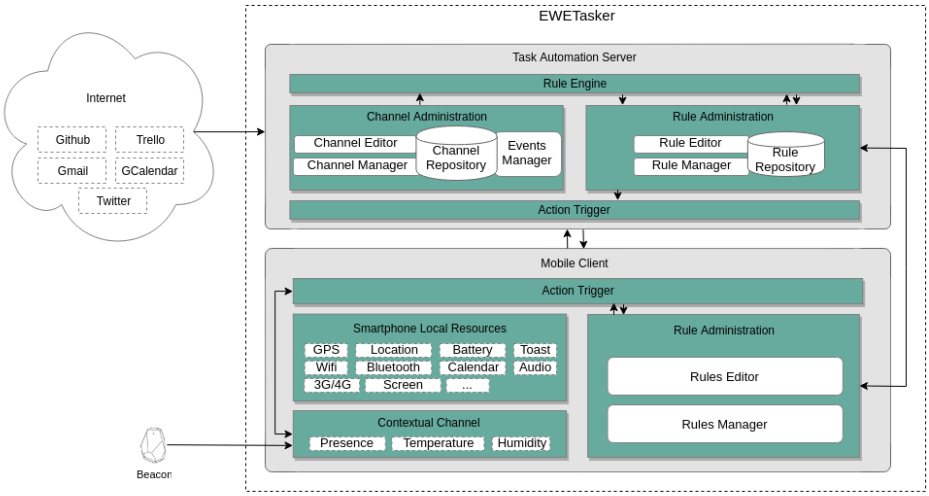

Figure 2. EWETasker architecture

users in the smart office scenario are able to manage their own automation rules, using for this purpose a graphic interface on a web client or a mobile client. Both modules will be described in detail below.

3.1. Task Automation Server

Users in a smart environment need a platform to manage their automation rules. Such a platform must provide functions for creating, updating, storing and erasing these rules. In addition, when programming these automations, users need to activate available channels. With the purpose of making easier the process of plugging new components, mentioned above, EWETasker provides functions for also managing these channels. Finally, the *task automation platform* has a main role: to automate tasks. It must be able to connect with several channels for receiving events, evaluating them together with stored rules and performing a corresponding action. Task Automation Server module has been designed to carry out all these functions, and it has been split into four submodules: Rule Administration, Channel Administration , Rule Engine, and Action Trigger.

The submodule involved in the rule creation is the **Rule Administration**. Its main purpose is to provide an automation *rule editor* in which users can configure and adapt their preferences about events coming from all sources. Users are able to create, remove, or edit rules in an easy way by a rule editor graphical interface, based on icons and “drag and drop" actions. As an example, the rule *“If I’m near the door, then open it."*, written in Notation3 is shown in Listing 1. This example rule has two conditions that must be fulfilled: the id of the sensors which detects the presence has to be “ABC123", and the distance from the sensor to the detected presence has to be lower than 2 meters. Nevertheless, we do not need to define specific rules for every sensor. One of the advantages of using semantic technologies is that these automation rules are generalized and can be based on event properties or classes, in contrast with other TASs such as Ifttt or Zapier.

Once created, rules are stored by the *rule manager* in the *rule repository*. Furthermore, users can import rules created by other users into their own repository. The *rule*

repository is implemented with the MongoDB database. *Rule repository* has one document collection for rule documents.

```
{
?event rdf:type ewe-presence:PresenceDetectedAtDistance.
?event ewe:sensorID ?sensorID.
?sensorID string:equalIgnoringCase 'ABC123'.
?event!ewe:distance math:lessThan 2.
}
=>
{
ewe-door:DoorLock rdf:type ewe-door:OpenDoor.
}
```

Listing 1: Rule example

Channel Administration provides users facilities for creating channels. Actions and events are defined using Notation3 [6], as shown in Listing 1. Once the required events and actions have been created, channels can be defined properly. *Channel manager* provides users access to all channels stored in the *channel repository*. This information can be managed through the *Channel editor*. *Channel repository* is also implemented as a MongoDB database, with three collections: *channels*, *events* and *actions*. Several vocabularies have been defined for our smart office scenario as shown in Listing 2.

```
ewe-door:DoorLock a owl:Class ;
rdfs:label "Connected door-lock"@en ;
rdfs:subClassOf ewe:Channel .

ewe-door:DoorOpened a owl:Class ;
rdfs:label "Door opened"@en ;
rdfs:subClassOf ewe:Event ;
rdfs:domain ewe-door:DoorLock .

ewe-door:LockDoor a owl:Class ;
rdfs:label "Lock door"@en ;
rdfs:subClassOf ewe:Action ;
rdfs:domain ewe-door:DoorLock .
```

Listing 2: Example of definition of the vocabulary for the channel *Connected Door*

The role of handling Internet and contextual events is carried out by the *events manager* in Channel Administration. Finally, the **Rule Engine** submodule is where events will be evaluated together with stored rules generating the corresponding actions.

Rule Engine is one of the most important modules in this system, and is based on an ontology model, which uses the EWE ontology [5]. It is divided into two parts: *EYE Helper* and *EYE Server*. The former is responsible for the reception of events from the Channel Administration and the load of rules that are stored in the repository. When a new event is received, *EYE Helper* captures it and loads the available rules. Then, events and rules are sent to the *EYE Server*, an Euler Yap Engine [7] reasoner that runs the ontology model inferences. The rule engine draws the actions based on the incoming events and the automation rules. EYE output is a string written in Notation3 whose processing

is not trivial, so we have developed a N3 parser that is provided by the *events manager* in the Channel Administration.

There are two **Action Trigger** submodules that will receive this JSON, one in the Task Automation Server and the other in the Mobile Client. The role of Action Trigger is to trigger the actions generated by the Rule Engine. The Action Trigger of the TAS is able to run actions that come from several channels like: Trello, Connected Door or Google Calendar. For this purpose, the Action Trigger module must be connected with those channels. So some examples of actions triggered by this module may be: opening a door, pushing a commit, sending an email or switching on a TV.

To sum up, automation rules are created in the Rule Administration using channels created in the Channels Administration. This submodule also receives events that are evaluated together with stored rules in the Rule Engine. Once the consequently actions have been generated, they are triggered in the Action Trigger module. As we have just described, the Task Automation Server is able to handle automation rules from their creation to their execution. However, in the smart office environment the user should be able to be connected via mobile client, which presented in the next section.

3.2. Mobile Client

Mobile Client module provides functions for managing rules from the smartphone in the **Rule Administration** submodule. In order to create rules from the *smartphone*, users should have access to the available channel list which is centralized in the web TAS. For this purpose, Rule Administration connects with the Channel Administration submodule of the web TAS, which provides a channel list that is used by Rule Administration to get the available channels with their events and actions. Once available channels are received, rules can be defined via the graphic interface that provides this module. Users need their local repository to be synchronized with the rule repository in the TAS, so when a rule is defined, it is sent to the TAS via POST request. In this way, users are able to manage rules in an easy way from their smartphones.

```
ewe-presence:PresenceSensor rdf:type ewe-
    presence:PresenceDetectedAtDistance.
ewe-presence:PresenceSensor ewe:sensorID 'A1B2C3'.
ewe-presence:PresenceSensor ewe:distance 1.
```

Listing 3: Input template

Users also need to automate rules where several sensors such as presence, temperature and humidity are involved. Therefore, the mobile client is also an event source. It generates events coming from the device itself or from other sensors such as beacons, and sends them to the TAS. Events coming from beacons are received by the **Contextual Channel**, whose main goal is to handle the data provided by them, and to send accordingly events to the server. The mobile receives from each beacon via Bluetooth the ambient temperature, the humidity and the distance. This submodule is responsible for converting this data to N3 events, that will be evaluated by the Rule Engine in the TAS. This process is straightforward, thanks to input templates. Input templates are structures written in N3 that can be used as base to create events. An example of these templates for the generation of a presence-detected event is shown in Listing 3.

As commented above, the device itself is also an event source. These events (i.e. battery level is low) are received by the **Smart Local Resources** submodule. It works in the same manner as the Contextual Channel, receiving events from channels such as Bluetooh, Wifi or Screen bright and converting them to N3 format before being passed to the TAS. However, the user's smartphone is not only an event source, but there are actions that are triggered by it (i.e. *When I arrive into the office, turn off data network and turn Wifi on*). For this reason, once the TAS has evaluated the passed events, the JSON containing the generated actions are received by the **Action Trigger** submodule.

The Action Trigger submodule has the same goal that its counterpart in TAS: to trigger actions. Some examples of actions that may be triggered by this module are: to show a notification, to mute the mobile or to call someone.

To sum up the whole architecture, this system counts with a TAS module that provides rules and channels management functions, evaluates events and rules and triggers the consequently actions; and a Mobile Client module that connects with beacons and other devices channels for generating events and triggering consequently actions.

4. Case study

To clarify some aspects explained on the architecture section, we will detail one scenario in order to show in depth the value of the project inside a smart office environment. The main scenario consists of an employee working in a company which uses EWETasker to establish task automation rules for avoiding repetition of daily activities. Some channels provided by the smart office environment are the employees' smartphones, with all their local resources (WiFi, Bluetooth, Calendar...) included, the beacons installed inside the office, the connected door and Internet channels related to social networks and project management applications. In Fig. 3 the beacon geographical distribution inside the smart office proposed in this case is depicted.

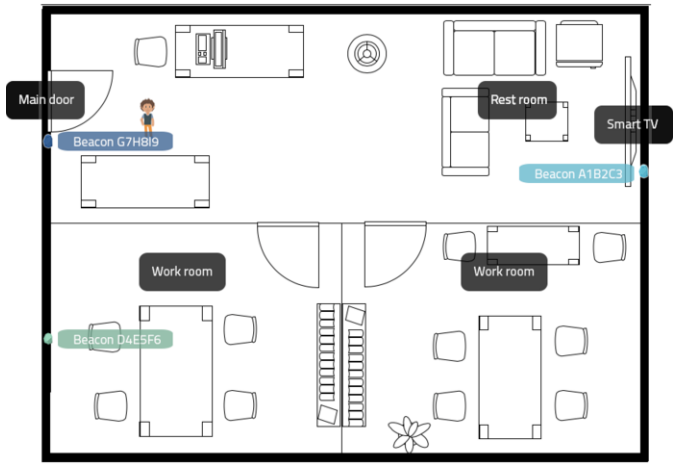

Figure 3. Map of the smart office scenario deployed at the research lab

The employee has the EWETasker application installed in his device and has declared the needed automation rules. The first activity the employee wants to automate is

the door opening when he arrives in the morning. For expressing this rule, he needs to open the EWETasker application and to define the rule, by selecting the *Presence Detector* as the event channel, and to be located at a distance lower than 2 meters from it as the event. Afterwards he must set up the action of the rule, selecting the *Connected Door* as channel and the door opening as the action. Finally, the rule created can be expressed in a more formal way like "*When I arrive work, then open the door.*". According to the Notation3 structure, and using the EWE ontology for channel modeling, some example rules will look in the mobile application and the web platform like showed in Fig. 4.

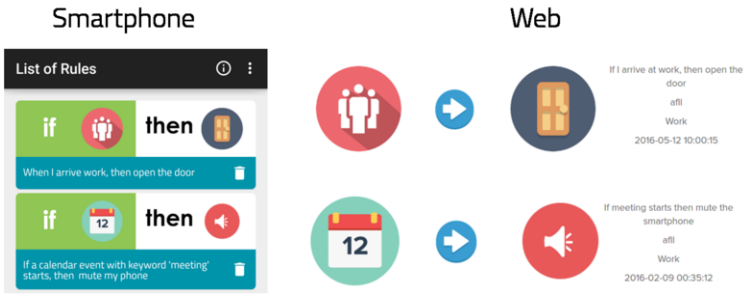

Figure 4. Interface of the rules in the Mobile Application and the Web interface

Once the rule is created, we are going to explain how the smartphone behaves when the event is triggered, and which elements described on the architecture participate in this process. First of all, the smartphone starts listening any change related to the rules created on the definition process. Every second, the application makes a sweep inside the beacon coverage area looking for any sensor transmitting. If success, then the device calculates the distance between each other, and generates an input for the inference engine, similar to the represented in Listing 3. This input is sent to the server, where will be processed, generating the consequent action.

All this automation process is invisible to the employee, who just need to preset the rule before he arrives to work and to authenticate with right credentials when the door is attempting to be opened. This avoids the use of external authentication devices like fingerprint readers or traditional keys, preventing its loss or undue forgery by people outside the company.

The second activity the employee wants to automate uses his smartphone local resources as channels, to define a rule. In this case, the user wants to mute his smartphone when a scheduled meeting starts. The channels involved in this process are the smartphone calendar and the notification module given by the phone. The calendar is where the user must create an event with a specific keyword as title and set the time range where it will take place. This keyword works as filter and is needed to listen only the events related to this category, so in this case it will be *meeting*. Once scheduled the appointment, the mobile application will be listening an event triggering from calendar. When the event starts, the input is sent to the rule engine responding with the phone audio manager as performer channel. Inside the smart office environment, this is useful to remind meetings or project milestones, and also to link the calendar events with third-party applications which focus on project version control or management.

Finally, beacons can also work as company information emitters. As shown in Fig. 3, several beacons have been deployed in the whole perimeter, so that they can provide con-

textual information to the networked enterprise. This contextual information is related to the enterprise, such as project status updates to employees, offers or new marketing campaigns to new clients that enter the office. In addition, the company and employees can define contextual automations for enabling energy saving or personalisation of the information shown in devices, such as the Smart TV, among others.

This scenario illustrates how the proposed ontology and architecture can realized our notion of smart office. Employees could define automation rules from their phone based on their context (e.g. position or working time based on calendar). With regards to companies, they could define good practices or corporate procedures as automation rules to maximize the company performance.

5. Related work

To determine which of the features discussed are supported by state of the art TASs, we have analyzed[4] web platforms for general audience (Ifttt), web platforms for business and enterprises (Zapier, Cloudwork, elastic.io, itduzzit), a web platform for cloud storage synchronisation (Wappwolf), mobile apps (Tasker, Atooma, Automateit, onx), and smart home systems (WigWag, Webee)[5].

Several authors have also proposed the use of ECA rules in smart environments, but usually these works have not reached the combination of enterprise and Internet services. Paraimpu [8] allows user to interconnect Http-enabled smartobjects and web services. However, final users need programming skills to configure them. As Karger [9] points out, one of the main disadvantages of these systems is that it is impossible to integrate new data suppliers and consumers unless the TAS company chooses to do so. To overcome this problem, Opasjumruskit et al. designed Mercury [10], a powerful TAS that features service discovery. This service is able to find appropriate sensors, services, actuators, etc; to perform certain functionality. Mercury relies on semantic annotation of web of device data sources so it can reason about them.

6. Conclusions and Future Work

This article has proposed a task automation architecture as an enabler of the envisioned notion of smart office. Task automation based on event-based ECA rules provides a unifying metaphor for users to configure and combine Internet services, IoT devices and mobile phones. The main characteristic of our proposal, which makes a step ahead compared to similar architectures [3,11] is the formalization of an ontology, EWE, which provides a number of benefits. Its formalization benefits from the advantages of semantic model, and should put impact on data interoperability and portability of automations, allowing external resource linking to the instances such as those from the LOD cloud.

[4]These systems are available online: Ifttt (http://ifttt.com), Zapier (http://zapier.com), Cloudwork (http://cloudwork.com), elastic.io (http://www.elastic.io), itduzzit (http://cloud.itduzzit.com), Wapwolf (http://wapwolf.com), Tasker (http://tasker.dinglisch.net), Atooma (http://www.atooma.com), AutomateIt (http://www.automateitapp.com), onx (http://www.onx.ms), WigWag (http://www.wigwag.com), Webee (http://www.webeeuniverse.com)

[5]A summary of the results is presented is available online at http://www.gsi.dit.upm.es/ontologies/ewe/study/full-results.html

The implementation of a smart office prototype, based on the proposed architecture, has shown some benefits of bringing automation into a workplace. The current implementation of the architecture performs a central execution of the rules, but it would be desirable to integrate a rule engine in the mobile component and research on the orchestration of automation among the central and mobile TASss. The main limitation for this is the availability of N3 rule engines optimized for mobile devices.

Nevertheless, some aspects of the proposal deserve more attention. Managing failures in IoT environments while executing automations [12]. An initial work on learning channel ontologies has been addressed in [5]. Other aspects, such as mining automation rules from user activity, or tackling social choice [13] will be addressed in future works.

Acknowledgements

This work has been partially supported by the Autonomous Region of Madrid through programme MOSI-AGIL-CM (grant P2013/ICE-3019, co-funded by EU Structural Funds FSE and FEDER) and by the Spanish Ministry of Industry and Competitiveness through the project SEMOLA (TEC2015-68284-R).

References

[1] J. C. Augusto, *Intelligent Computing Everywhere*. Springer London, 2007, ch. Ambient Intelligence: The Confluence of Ubiquitous/Pervasive Computing and Artificial Intelligence, pp. 213–234.

[2] K. Furdik, G. Lukac, T. Sabol, and P. Kostelnik, "The network architecture designed for an adaptable iot-based smart office solution," *International Journal of Computer Networks and Communications Security*, vol. 1, no. 6, pp. 216–224, 2013.

[3] S. Mennicken, J. Vermeulen, and E. M. Huang, "From today's augmented houses to tomorrow's smart homes: New directions for home automation research," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '14. New York, NY, USA: ACM, 2014, pp. 105–115.

[4] M. Coronado, C. A. Iglesias, and E. Serrano, "Modelling rules for automating the evented web by semantic technologies." *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7979–7990, 2015.

[5] M. Coronado, "A Personal Agent Architecture for Task Automation in the Web of Data. Bringing intelligence to everyday tasks," Ph.D. dissertation, ETSI Telecomunicación, feb 2016.

[6] T. Berners-Lee and D. Connolly, "Notation3 (n3): A readable rdf syntax," W3C Team Submission, Tech. Rep., 1998.

[7] J. D. Roo, "Euler yet another proof engine," http://eulersharp.sourceforge.net, 2013.

[8] A. Pintus, D. Carboni, and A. Piras, "Paraimpu: a platform for a social web of things," in *Proceedings of the 21st international conference companion on World Wide Web*. ACM, 2012, pp. 401–404.

[9] D. R. Karger, "The semantic web and end users: What's wrong and how to fix it," *Internet Computing, IEEE*, vol. 18, no. 6, pp. 64–70, 2014.

[10] K. Opasjumruskit, J. Expósito, B. König-Ries, A. Nauerz, and M. Welsch, "Mercury: User centric device and service processing–demo paper," in *19th Intl. workshop on Personalization and Recommendation on the Web and Beyond, Mensch & Computer, Konstanz, Germany*, 2012.

[11] F. Cabitza, D. Fogli, R. Lanzilotti, and A. Piccinno, "Rule-based tools for the configuration of ambient intelligence systems: a comparative user study," *Multimedia Tools and Applications*, pp. 1–21, 2016.

[12] B. Ur, M. P. Y. Ho, S. Brawner, J. Lee, S. Mennicken, N. Picard, D. Schulze, and M. L. Littman, "Trigger-action programming in the wild: An analysis of 200,000 ifttt recipes," in *Proc. CHI'16*, 2012.

[13] E. Serrano, P. Moncada, M. Garijo, and C. A. Iglesias, "Evaluating social choice techniques into intelligent environments by agent based social simulation," *Information Sciences*, vol. 286, pp. 102.–124, December 2014.