Semantic Web 0 (0) 1
IOS Press

FoT-Rules: A Semantic Rule-based Approach for Smart Spaces Through Fog of Things

Cleber Santana [a,b,*], Brenno Alencar [a], Ernando Batista [a,b] and Cássio Prazeres [a]
[a] *WISER: Web, Internet and Intelligent Systems Research Group. Computer Science Department, Federal University of Bahia (UFBA), Bahia, Brazil*
E-mails: brenno.mello@ufba.br, prazeres@ufba.br
[b] *Federal Institute of Bahia (IFBA), Bahia, Brazil*
E-mails: cleberlira@ifba.edu.br, ernando.passos@ifba.edu.br

Abstract. Smart Spaces or Smart Environments are related with ubiquitous computing in the sense that sensors, actuators, and others computational elements should be embedded seamlessly in the everyday objects. In the Internet of Things (IoT), Smart Spaces will enable environments to adapt according to people (users) needs by using smart and connected objects. However, to turn the IoT view into a reality, the users should know about technical details of such objects, which is not a trivial task for most ordinary users. Therefore, this article presents FoT-Rules, a proposal for the construction of semantic rules aiming to create Smart Spaces through Fog of Things, which is a paradigm for Fog Computing in the Internet of Things. FoT-Rules is designed to enable ordinary users to create semantic rules in the Event-Condition-Action standard (ECA) and to take actions according to the environment at the edge (Fog) of the network. In this work, we present a scenario, where the user can create semantic rules in the ECA standard, and then FoT-Rules perform the following functionalities: (i) obtain the semantic model that contains information related to an IoT device; (ii) execute a semantic reasoner over the semantic model according to the rule created by the user; (iii) provide a semantic observer that is responsible for observing changes in IoT devices; and (iv) in case the rule created by the user is activated, an action is taken for an IoT device. Finally, we performed three types of evaluation on our FoT-Rules proposal: reliability, efficiency and usability.

Keywords: Internet of Things, Smart Space, Semantic Web, Fog Computing, Fog of Things, Semantic Rule

1. Introduction

The proliferation of smart objects/devices (e.g. sensors and actuators) with communication and performance capabilities is increasingly turning the Internet of Things (IoT) view into a reality. According to Jayavardhana et al. [1], it is necessary to go beyond computational scenarios of IoT, which only use mobile devices, sensors and actuators so as to incorporate intelligence into everyday environments by making everyday objects smart and connected. Considering the interaction between humans and the IoT smart objects, Bikakis and Antoniou [2] claim that it is fundamental to transform everyday environments [3] (work, home, and others) into Smart Spaces that are able to adapt to changing contexts as well as people's needs and wishes.

According to Cabitza et al. [4], Smart Space is a scenario of the IoT that will enable environments to adapt according to people (users) needs, and will be able to feel and act according to requests made by them. As a consequence, Smart Spaces will operate based on requests for IoT devices made by users [4]. However, setting up and using IoT devices require knowledge about the details of such devices, which is not a trivial task for most ordinary users [5]. Mainetti et al. [6] argue that high-level access by users to Smart Spaces is still an open issue. The logic to manage sensors and actuators is not simple as it involves the implementation of complex algorithms. The trend is to provide end users with the ability to properly actuate and deactivate actuators when sensors detect changes in the environment.

*Corresponding author. E-mail: cleberlira@ifba.edu.br.

In addition, users are only interested in the final result of the interaction with IoT devices and, as a result, it is ideal that they come up with high-level rules that allow them to manage the behavior of IoT devices [7]. Therefore, some works [4, 6–11] have proposed the use of Event-Condition-Action (ECA) pattern as a solution for defining simple rules in which one or more actions can be performed when certain events occur under certain conditions. The Event-Condition-Action (ECA) pattern is inherited from the field of expert systems and it is largely used to provide reactive functionality in database systems [12].

In order to make information understood by devices and to render the execution of rules autonomous and efficient, it is important to make use of semantic technologies, since from the definition of a semantic model, a semantic reasoner can infer new information based on what is already present in the system [6]. Al-Fuqaha et al. [13] expound that semantic technologies in the Internet of Things refers to the capacity to extract knowledge by means of different devices which includes recognizing and analyzing data. In this context, some works [6, 10, 14–16] have proposed the use of semantic rules in the Internet of Things to allow the integration between devices so as to make them autonomous and efficient.

The aforementioned works present solutions in which rules are defined and executed mostly in the cloud environment. However, according to Bonomi et al. [17], IoT solutions that are based only on a centralized cloud paradigm have certain limitations such as limited support for mobility, high demand for bandwidth, high delay, and high latency [18, 19]. These limitations are undesirable in Smart Spaces scenarios in which users are at the edge of the network. In order to avoid and/or reduce these limitations, Bonomi et al. [17, 20] presents the Fog Computing paradigm, which aims to provide compute, storage, and networking services, between devices and traditional clouds, at the edge of network. In addition, Fog Computing can improve scalability and provide high levels of reliability and fault tolerance [18, 21].

In this context, this work proposes FoT-Rules, an approach based on semantic rules built into the ECA format to support end-users on the creation of Smart Spaces at the edge of the network. FoT-Rules extends the Fog of Things (FoT) paradigm proposed by Prazeres and Serrano [22], which aims to design and implement Fog Computing platforms for the Internet of Things. In addition, to illustrate the benefits of using a semantic approach to the creation of rules in the Fog of Things to provide Smart Spaces, we have designed an architecture, and then implemented and deployed it at a smart building.

According to Bikakis and Antoniou [2], a rule-based reasoning model has already been successfully applied in several domains such as the Web. By adopting a semantic-based representational model for creating Smart Spaces at the edge of the Internet of Things, we can point out the following benefits:

- Integration with ontology models: FoT-Rules has, in its architecture, a semantic model that allows the integration of rules with the semantic model itself. The use of a semantic model allows heterogeneous agents to communicate, share and combine their knowledge, thereby avoiding ambiguities and incorrect transactions. In FoT-Rules, formal constructions (e.g. SPARQL CONSTRUCT and Framework Jena) are used to create rules.
- Expressivity: When using formal constructions to create Smart Spaces, we obtain the following concepts. (i) Modularity – it allows for detecting causes or effects of specific changes in a context, and for managing the rules in a single rule repository as well as several rule repositories that can be distributed in other scenarios. And (ii) reactivity – when using the ECA format.
- Information Hiding: In the context of Smart Spaces, information is obtained from sensors. FoT-Rules obtains sensors' data that have been semantically enriched. From the semantics it is possible to infer knowledge of context and prescribe the behavior of the system from the rules created by the end user.

The remainder of this article is organized as follows. Section 2 describes some state-of-the-art works related to our proposal. Section 3 presents FoT-Rules, our proposal for semantic rules in the FoT. Then, in Section 4, we present the architecture and implementation of our proposal. In Section 5, we present the results of the evaluations. Finally, we provide some discussions, recommendations for future work and final remarks in Sections 6 and 7.

2. Related Works

In recent years, some works have been developed with the objective of constructing Smart Spaces by employing rules. Some of these works, as our proposal, uses the rules in the ECA pattern and others works do

not use ECA pattern. These both kind of works are presented in Section 2.1 and compared in Section 2.2, in order to highlight our proposal in contrast with the state-of-the-art works.

2.1. State-of-the-Art

One of the first works to use the ECA format in Smart Space was proposed by Leong, Ramli and Perumal [8]. In that work, the authors propose a framework based on rules in the ECA format for the management of heterogeneous systems in a Smart Home environment. The goal is to provide interoperability between sensors and actuators systems with the implementation of Web Services that exchange messages using the SOAP protocol. Our work is also based on the ECA format used by Leong, Ramli and Perumal [8] because we propose to implement and to run ECA rules at the network edge of IoT systems.

Huang and Jevad [14] provide an architecture for describing and processing information from sensors. The proposed architecture relates to the following: (i) collecting data from heterogeneous sensor nodes and sending them through gateways; (ii) semantic annotation of the data collected; (iii) semantic data processing through a rule mechanism to perform the inference on the semantically annotated sensor data; (iv) application to perform the interaction with the sensor.

Tuomisto et al. [7] present a rule editor for the construction of rules in the Internet of Things. This is aimed at people with little knowledge of technology so they can connect and configure sensors and their appliances. This editor provides an end user-focused environment. To facilitate its use, a graphical representation of IoT devices is included.

Choi et al. [15] propose a visual environment to assist users in creating context-sensitive services. With this environment, users can define the service environment, context and behavior of actuators (e.g. activate, deactivate). In addition, a semantic sensor data processing, based on ontologies, is proposed, which provides: (i) abstraction of the device and service to remove heterogeneity over syntax and structure, (ii) a semantic reasoner to determine certain phenomena, and (iii) service execution condition to control relevant actuators.

Mainetti et al. [6] propose an architecture based on Semantic Web technologies to build systems in the context of the Internet of Things. In their work, sensor data are semantically enriched, while interactions with actuators are conducted by high-level actions configured by users. Applications are in the form of a rule Event-Condition-Action, while the layered architecture separates aspects of high semantic reasoning with low-level details (e.g. execution details). A comparison with leading-edge solutions is made, and appropriate technologies for their implementation are suggested.

Valtolina et al. [9] propose an architecture to perform the collection and diffusion of data originated and transmitted by sensors. The authors also propose a new rule-based language that enables end users to configure complex events that include temporal, repeatable, and non-exact attributes (e.g. fuzzy logic). While supporting end users in rule making, the proposed architecture depends on the analysis of information that is arranged in a data warehouse.

Cabitza et al. [4] present a study that compares two systems (i.e. IFTTT and Atooma) which provide support for end-users when creating rules for the Internet of Things. This allows them to define Smart Spaces with a conceptual framework in three layers: (i) physical, (ii) inference, and (iii) user. Their work provide indicators for the implementation of the user layer, however, they do not implement it. The results suggest that the most appropriate interaction style for rule design by end users should follow the ECA standard. In addition, the tools must be multiplatform, that is, they must offer a wide variety of services and devices to be used and combined, while the user interface must be easy to use, attractive, among other features.

Gyrard et al. [16] propose a rule-based mechanism for analyzing data from devices of the Internet of Things called S-LOR (Sensor-based Linked Open Rules) and its use in smart cities. The proposal uses Web technologies (e.g. Web Services and Semantic Web), and deduces meaningful information from devices (e.g. sensors) that send data to be processed on the Web. The goal is to share, reuse, and execute rules from the data interpreted from devices. S-LOR is based on the principles of Linked Open Data, and was designed to select the best approach for data analysis according to the needs of data scientists. This mechanism can guide developers in choosing a specific approach to semantic-based reasoning (e.g. machine learning, distributed approaches, real-time processing, and recommendation systems).

Kaed et al. [10] present a mechanism for creating semantic rules in IoT environments (industrial gateways) focused on energy consumption control. They propose a tool for strategies of industrial environment controls based on dynamic and flexible rules. In addition, it is possible to manipulate semantic queries and provide

results according to previously defined concepts in ontologies. The proposed tool contains the layers: rules engine – responsible for providing gateways with decentralized intelligence and the ability to manage connected devices; Semantic Engine – responsible for providing the rules with the ability to integrate semantic queries; Rule Design – designed to support ECA rules format, publish/subscribe standard and orchestration between rules.

Wang et al. [11] propose a specific recommendation algorithm for IoT environments that considers user feedback on rules, user/device/rule interactions, and additional information such as matching rules and types of IoT devices. Traditionally in recommendation systems, users are defined by ID, however for IoT environments, not only the ID but also the location of the user was taken into account. The rules implemented through the recommendation algorithm improve IFTTT-like rules as a means to allow a greater range of actions (e.g. multiple triggers, multiple actions).

Takatsuka et al. [23] present a framework for M2M environments called RuCAS (Rule-based management framework for Context-Aware Services). This framework enables the systematic and context-aware management of M2M services with ECA rules. RuCAS enables users to define their own contexts with Web Services (e.g., sensor services, information service, networked appliances). Based on these contexts, users can define ECA rules (facilitated through a GUI) that bind contexts to custom actions.

Kumar et al. [24] propose a technique for adapting semantic rules and their applications in a scenario of Smart Buildings. In this context, the proposal is applied in cases of use of Smart Buildings for energy efficiency. The main idea of the work is to adapt and reuse semantic rules between users for the activation of alerts and monitoring of idle devices, taking into consideration possible contextual changes of the environment that the user is inserted. Although the user has to directly manipulate the rules, the environment information (triggered by the rules) can be viewed through the alert monitoring app (similar to a dashboard).

Taherizadeh, Stankovski and Grobelnik [25] describe and implement a distributed computing architecture that uses the concepts indicated by the Cloud Computing, Fog Computing and Microservices for Internet of Things. The proposal aims to support intelligent IoT applications according to a varying number of workloads focusing on IoT devices that dynamically change among geographic locations. The architecture is composed of three layers: (i) Edge Infrastructure, which can be cars, robots, smartphones or similar; (ii) Fog Infrastructure, instances with computational resources able to service Edge Infrastructure most of the time; (iii) Cloud Infrastructure, responsible for managing the demand of Edge Infrastructure in scenarios of high demand for computational resources. Because it is a solution focused on devices with location variations, all layers implement IoT microservices in dockers containers, facilitating the change and provision of microservices among the nodes of the different layers.

Froiz et al. [26] present a resindential automation system for Fog Computing environments called (ZiWi - Zigbee and WiFi).The main objective of the work is to provide IoT environments (focused on residential environments) in which ZigBee and WiFi technologies can be used. In the work it is proposed that actuator devices use connectivity via WiFi, since they need to be continuously connected waiting for asynchronous commands and sensor devices implement the ZigBee to send their data (ideal for energy savings). In this context, the main components of the proposed solution are: Sensor nodes (sensors), Actuators nodes, IoT Gateway and Home controller. Specifically the Home controller is a layer implemented by the OpenHAB residential control application [27]. This layer is responsible for communicating components (sensors and actuators) and user and also for creating IFTTT rules via the Web or Mobile interface.

2.2. Comparison

The rule-based semantic approach for Smart Spaces proposed in this work has the following characteristics:

- ECA rules: use of the ECA Rules format for rule construction;
- Rules editor: use of a graphical user interface (GUI) for rule construction;
- Device heterogeneity: elaboration of rules for different devices from different vendors;
- Semantic modeling: context definition, rules construction, device and service descriptions by using Semantic Web technologies and standards;
- User-centered design: application focuses on the end user;
- High-level abstraction for hiding low-level details: the user can easily set the environment and the rules without previous knowledgment about rules, IoT device configuration, Semantic Web, and other related technologies and/or standards;

– Solution implementation: implementation and evaluation of a concrete architecture;
– Execution environment at the network edge: Cloud Computing plays an important role in the Internet of Things, once it enables numerous connected devices to work together. However, our work focuses on devices that are closest to the edge of the network, because it is the appropriate platform for several services and critical applications of the Internet of Things such as Smart Spaces (Smart Homes, Smart Buildings, Smart Cities, among others) [20].

Based on the characteristics listed above, Table 1 summarizes a comparison among the various proposals presented in Section 2.1. These proposals are identified in Table 1 by the name of the first author of the work.

As can be viewed in Table 1, some works [4, 6–11, 23] use the ECA format to build the rules for Smart Spaces. In FoT-Rules, the use of rules in the ECA format helps end users when using the application, as explained in Section 4.2.2. A visual editor (GUI) can help the user in the creation of those rules. In this direction, some works [7, 9, 14, 15] have proposed user-centered approaches, while others works [4, 6, 16, 23, 26] provide, in their solutions, a layer that represents the interface between the end user and the proposed architecture. In our proposal, the GUI layer plays a fundamental role, once it allows the end user to create applications in the ECA format and to abstract to the semantic aspects and devices used in the model.

Except for Wang [11], who do not specify the use (or not) of several devices, the heterogeneity of devices is considered by all aforementioned works. However, Huang and Javed [14] focuses on devices from wireless sensor networks instead of IoT, as in our proposal. In the works of Kaed et al. [10] and Taherizadeh, Stankovski and Grobelnik [25], the execution of microservices can be done in different levels (Fog and Cloud). However, these solutions do not allow common users to manipulate information about the rules (Rules visual editor and User centered aspects of FoT-Rules).

As in our proposal, Semantic web technologies are applied by Huang and Jevad [14], Choi et al. [15], Mainetti et al. [6], Gyrard et al. [16] and Kumar et al. [24] – see Table 1. However, these works does not implement the semantic rules at the network edge, as we propose in FoT-Rules. Although Kaed et al. [10] propose semantic rules at the edge of the network, they are not user-centered and, for that reason, they do not present a rules visual editor, as we propose in FoT-Rules.

Except for Kaed et al. [10], high-level abstraction is treated by all the works analyzed in Table 1. Most solutions abstract raw sensor data as a means to collect useful information for end users. In FoT-Rules, the abstraction is given at the application level through which the user can configure the actions from the interface.In our work, actions involve devices that are available in the Fog of Things.

In Table 1, concrete implementations of the proposed architectures were performed by Leong, Ramli and Perumal [8], Huang and Jevad [14], Choi et al. [15], Gyrard et al. [16], Kaed et al. [10], and Wang [11]. Then, in these works, as in FoT-Rules, several kinds of evaluation can be performed, as those we discuss in Section 5.

While making this comparison, we observed that several works dealt with some of the research questions addressed by our approach (summarized in Table 1). However, none of the proposals addressed the delivery of services, to the user and in the Internet of Things, close to where the data is collected (network edge), with the objective of overcoming the current limitations of infrastructure and data processing, thus alleviating the demands of heavy computing resources that can occur on remote servers at the cloud.

3. FoT-Rules: the Rule Approach for the Fog of Things

In this work, we propose FoT-Rules, which aims to enable the creation of ECA rules described with Semantic Web standards in Smart Spaces scenarios for the Fog of Things. In FoT-Rules, when an event occurs at the network edge, and certain conditions are met, one or more actions may be taken in response to the detected event.

3.1. Fog of Things

This work extends the SOTF-IoT platform [22, 28, 29], which uses the Fog of Things paradigm with the following components:(i) FoT-Devices; (ii) FoT-Gateways; (iii) FoT-Servers; (iv) applications; (v) and, a message-service oriented middleware to allow for the exchange of messages in the FoT. This extension is done by adding a feature for the execution of rules over the SOTF-IoT platform. In addition, some SOFT-IoT

Table 1
Comparison related work.

Authors	ECA Rules	Rules Visual Editor	Device heterogeneity	Semantic Modeling	User Centered	High-Level abstraction	Solution implementation	Execution environment
Leong [8]	✓		✓			✓	✓	Edge
Huang [14]		✓	✓	✓	✓	✓	✓	Edge
Tuomisto [7]	✓	✓	✓		✓	✓		Cloud
Choi [15]		✓	✓	✓	✓	✓	✓	Edge
Mainetti [6]	✓	✓	✓	✓		✓		Cloud
Valtolina [9]	✓	✓	✓		✓	✓		Cloud
Cabitza[4]	✓	✓	✓			✓		Cloud
Gyrard [16]		✓	✓	✓		✓	✓	Cloud
Kaed [10]	✓		✓	✓			✓	Fog and Cloud
Wang [11]	✓		Not specified			✓	✓	Not specified
Takatsuka [23]	✓	✓	✓		Not specified	✓	✓	Cloud
Kumar [24]			✓	✓			✓	Cloud
Taherizadeh [25]			✓			✓	✓	Fog and Cloud
Froiz [26]		✓	✓		Not specified	✓	✓	Fog
FoT-Rules	✓	✓	✓	✓	✓	✓	✓	**Fog of Things**

components are extended and/or re-implemented in order to enable the implementation and execution of semantic rules, as described:

- FoT-Device (i.e. sensors and actuators discussed in Section 4) allows for the integration between the real world and the digital world.
- FoT-Gateway is a low-cost device with constrained processing capabilities and memory resources that allows RESTful Web services (i.e. Rules Execution, Semantic Observer, and Semantic Reasoner, as discussed in Section 4) to access IoT services at the network edge.
- FoT-Server has the capability to store: semantic enriched information collected by FoT-Devices (e.g. Temperature Sensor), and semantic rules, as described in Section 4.
- Application, in this work, is a mobile one (e.g. Rules Editor presented in Section 4) for the creation of rules by the user. The Rules Editor accesses the services offered by the FoT-Gateway and generates rules to be stored at a FoT-Server.
- The Message-Service Oriented Middleware allows for the communication not only between the application (in this article, the Rules Editor) and IoT services deployed in FoT-Gateways, but also between FoT-Servers, FoT-Gateways and FoT-Devices.

3.2. Problem Statement: an IoT Scenario

In order to understand this FoT-Rules proposal, a usage scenario for Smart Building is illustrated in Figure 1. In this scenario, the user can create rules (Figure 1 - part 1) that make it possible to change air temperature and open windows so as to keep humidity at a healthy level. The user can also create rules so that the lights are turned off whenever there is no one in the room, or so that the data collected by sensors can be analyzed to see if actions are required, such as activating an alarm or lighting, or a semaphore in case of fire, or even to detect an increase in temperature levels. Each floor may contain its own FoT node (FoT-Gateways or FoT-Servers). On the one hand, FoT-Gateways (Figure 1 - part 2) may be responsible for performing functions related to humidity control, emergency monitoring and response, temperature, and lighting. On the other hand, FoT-Servers (Figure 1 - part 3) may be responsible for providing a storage infrastructure in buildings so as to complement the limited capabilities of smartphones, tablets and/or computers located at the network edge.

The elaboration of these rules can be made efficient and effective (e.g. supporting reduction in energy consumption) by means of data analysis. Therefore, the data collected by sensors can be sent to the cloud for a

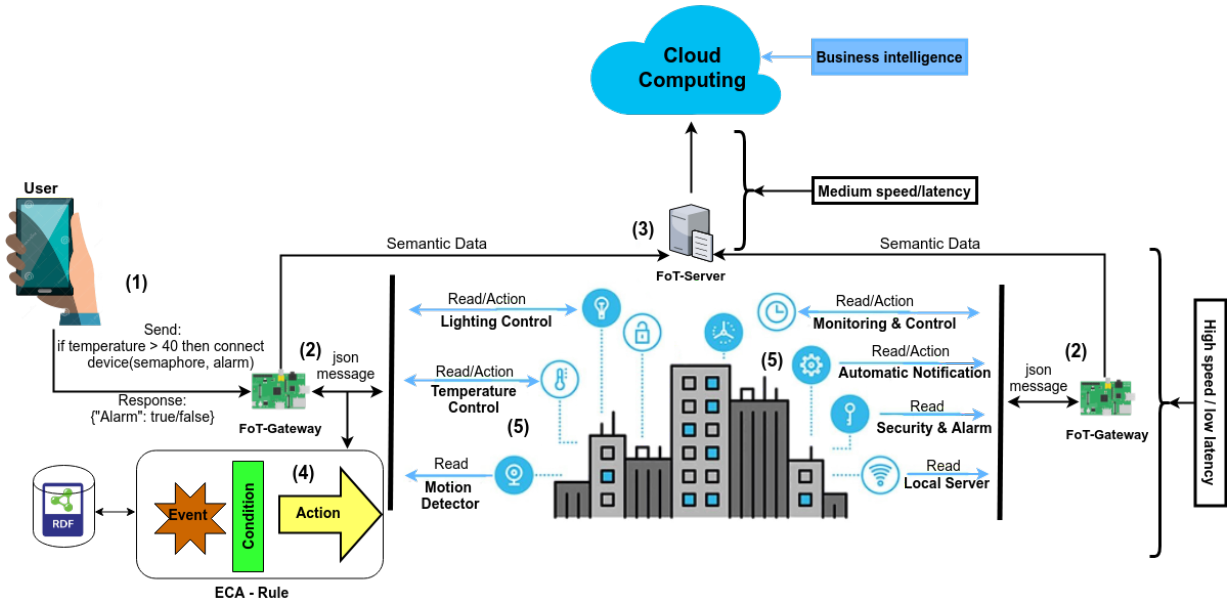

Fig. 1. FoT-Rules usage scenario.

large-scale analysis, thereby enabling improvements in the rules previously created by the user. In this context, the user is now in the cloud which, based on the analysis performed, should improve user rules and reduce resource usage.

It is important to note that in a scenario that illustrates the need for cutting-edge intelligence and localized processing, the use of a solution that is fully deployed in the cloud is impractical given the latency and delay that can be expected on the Internet. Thus, the solution based on Fog of Things proposed in this work proves to be feasible in scenarios such as the one shown in Figure 1.

In that scenario (Figure 1), the user creates the rule in the ECA format (Figure 1 - part 4) with the use of a mobile application as a means to avoid that air temperature and humidity in a building exceed their predetermined value, thus maintaining quality at a healthy level and without risks to the environment. If the temperature exceeds the allowed maximum value predetermined by the user, the devices (i.e. fire alarm, semaphore, window) are automatically activated by actuators. Listing 1 presents pseudo-code with rules in the ECA format which means: (A) if temperature is above 40°C, then some devices (e.g. alarm, semaphore) are activated; (B) if the humidity sensor detects results below 40% or above 60%, then other devices (windows, air conditioner) are activated.

```
(A)
on sensor temperature
if (temperature > 40) (hot) then
        connect device(semaphore, alarm)
(B)
on sensor humidity
if ((humidity) < 40 || (humidity > 60)) then
        connect device(window, air conditioner)
```

Listing 1: A rule described in pseudo-code.

Still in the scenario of Figure 1, an FoT-Gateway is the central node, responsible for coordinating the events among the various sensors and actuators available in this scenario. FoT-Gateway collects the data from the sensors and stores them in a local database (in the FoT-Gateway itself). The collected data is then semantically enriched and stored in an FoT-Server. At this point, FoT-Rules starts acting (Figure 1 - part 5) in the scenario by invoking its functions, which are implemented by modules distributed in FoT-Gateways and FoT-Servers (all modules are explained in Section 4).

The semantic enrichment of the data is performed by the Semantic Model module. Then, the Semantic Reasoner module identifies whether a rule has been activated. For instance, the temperature has reached the predetermined limit (as described in Listing 1-

A) which changes the data model available in the server, thereby activating the appropriate property. Afterwards, the Semantic Observer module detects that such a property has been enabled and then sends a command to the FoT-Gateway to turn on the air conditioner, for example. Finally, the Rules Execution module activates the actuator of the air conditioner so as to maintain the temperature below the limit.

Information regarding changes can be performed by the semantic reasoner. For example, when detecting that temperature is above 40°C, the semantic reasoner can change the isSettingFor property to true, and thus notify the semantic observer in order to activate those devices.

3.3. ECA Rules in the Fog of Things

Figure 2 illustrates the FoT-Rules process for the construction and execution of ECA rules: (1) in the first step, the user defines the rules by means of a mobile application; (2) in the second step, the semantic reasoner, which has the capabilities to make inferences about the user-defined rules and to make changes in the model, is activated; (3) in the third step, an observing agent identifies changes in the semantic model that can satisfy a condition; (4) finally, in the fourth step, if the condition exists, the process is completed with the action being sent to the device.

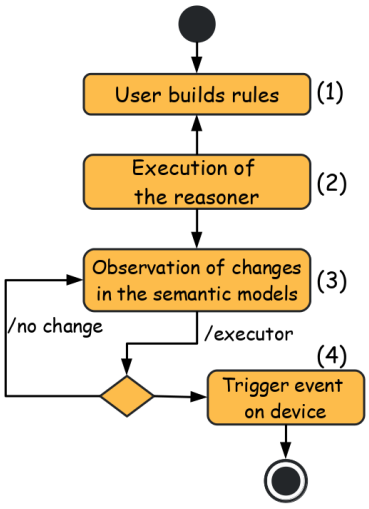

Fig. 2. Process to run rules in FoT-Rules.

Listing 2 illustrates the example of semantically enriched information obtained from the sensors. In this case, the hasDataValue property is responsible for storing the temperature, while the isSettingFor property holds information regarding changes that can be performed by the semantic reasoner. For example, when detecting that temperature is above 40°C, the semantic reasoner can change the isSettingFor property to true and thus notify the semantic observer as a means to activate those devices.

```
1 <http://wiser.dcc.ufba.br/smartUFBA/devices>
2 dul:hasDataValue 36;
3 dul:isSettingFor false .)
```

Listing 2: Semantically enriched information for an IoT device.

4. FoT-Rules: Architecture and Implementation

FoT-Rules extends some previous results [22, 28] in order to provide a rule-based approach for Smart Spaces through a Fog of Things platform: the SOFT-IoT (Self-Organizing Fog of Things for the Internet of Things) platform. In this section, we describe: how FoT-Rules modules is implemented in the SOFT-IoT platform (Section 4.1); and the FoT-Rules modules implementation (Section 4.2).

4.1. FoT-Rules Architecture and Technologies

As presented in Figure 3, the FoT-Rules was implemented over the SOFT-IoT platform, an infrastructure based on OSGi[1], which is a specification for middlewares that combines the functionality of a service-based architecture, flexibility, and modularity. This infrastructure is distributed along FoT-Gateways and FoT-Servers in order to implement the FoT paradigm presented in Section 3. Therefore, in the SOFT-IoT platform, FoT-Rules implements certain modules as microservices, as illustrated in the application layer of Figure 3, whose purpose is to enable the execution of ECA rules on the semantic level and trigger actions in IoT devices.

In the FoT-Rules proposal, the FoT-Gateway is deployed in a Raspberry Pi device, which implements the infrastructure as illustrated in Figure 3 with the following components: (i) service-oriented middleware based on the OSGi; (ii) message-oriented middleware.

[1]https://www.osgi.org/

The service-oriented middleware provides interfaces for applications to access devices through the RESTful Web Services (CXF in Figure 3). The message-oriented middleware (MQTT[2] in Figure 3) support the message exchange between the FoT-Gateway and IoT devices.

The FoT-Rules approach uses an OSGi specification implementation as the basis for its infrastructure. OSGi technology is a set of specifications that defines a system of dynamic components for Java applications. These specifications enable the development of applications consisting of several components that are involved in bundles[3]. As OSGI container we used ServiceMix[4], which allows the integration of functionalities such as Apache CXF, Karaf, MQTT, among others, thus creating a powerful runtime platform used to build the components of the FoT-Rules. ServiceMix is deployed in FoT-Gateways and FoT-Servers in order to support the deployment of four of our five FoT-Rules modules (as can be viewed in Figure 3): Semantic Reasoner, Semantic Observer, Rules Execution, and Semantic Model. However, the first three aforementioned modules are implemented on a ServiceMix instance deployed in the FoT-Gateway, as well as the last one (Semantic Model is deployed in the FoT-Server or Fuseki Server [5]). Such server is an endpoint for RDF triples and enables queries using SPARQL language over the HTTP protocol. The fifth module (Rules Editor, see Section 4.2.2) is not in Figure 3, because it is a mobile application and do not work in an OSGI Container.

4.2. FoT-Rules Implementation

After having presented in Section 4.1 a description of technologies used in the implementation of FoT-Rules modules, the following sections describe the modules themselves: Semantic Model (Section 4.2.1), Rules Editor (Section 4.2.2), Semantic Reasoner (Section 4.2.3), Semantic Observer (Section 4.2.4), and Rules Execution (Section 4.2.5).

4.2.1. Semantic Model

The Semantic Model module obtains information semantically described by RDF triples from the ServiceMix instance, deployed in the FoT-Server. This model is based on the work proposed by Andrade et al. [30], which presents the data interplay between Fog and Cloud by using semantic enriched data.

Listing 3 shows some RDF triples stored on the FoT-Server and obtained by the Semantic Model module. In these triples, line 1 identifies the sensor, lines 3 and 6 identify the time at which the information was obtained, and line 9 identifies the temperature collected by the sensor

4.2.2. Rules Editor

By using the FoT-Rules, the user can create rules with any external entity, which we call the Rules Editor module. In this article, we implemented the Rules Editor module as a mobile application[6], as presented in Figure 4. This mobile application implemented on Android that allows the user to visually create rules in the **EVENT**, **CONDITION**, and **ACTION** format. In Figure 4, the user chooses from a list the sensor for which s/he wishes to create rules and informs the appropriate values. In addition, in Figure 4, the user can obtain an up-to-date information regarding the environment (e.g. humidity, temperature).

Listing 4 shows an example of a rule that can be created by the mobile application of Figure 4, and executed by the FoT-Rules. In Rule A, it is inferred that temperatures higher than 40^{o}C are considered high (highTemperature true). In rule B, it is inferred that the air humidity is considered inappropriate when is lower than 40% or higher than 60%.

4.2.3. Semantic Reasoner

The Semantic Reasoner[7] module receives, as input data, the model (for instance, Listing 3) obtained by the Semantic Model module, as described in Section 4.2.1. This module is responsible for making inferences about the user-generated rules presented in Section 4.2.2. The Semantic Reasoner module uses the Jena framework to perform semantic reasoning over the RDF data, as shown in Listing 3. When the Semantic Reasoner executes the rule and infers that a condition was satisfied, it performs an update on the model. This update is performed only on the RDF triples that satisfy the rule by using the SPARQL UPDATE according to Listing 5. In line 4 of Listing 5, the information contained in the model is deleted, and then in line 5 the same property is inserted with a new value. In line 6, a constraint is applied to correctly identify the value to be changed.

[2] http://mqtt.org/
[3] https://www.osgi.org/developer/architecture/
[4] http://servicemix.apache.org/
[5] https://jena.apache.org/documentation/fuseki2/index.html
[6] https://github.com/WiserUFBA/Rules-Editor
[7] https://github.com/WiserUFBA/Rules-Semantic-for-IoT

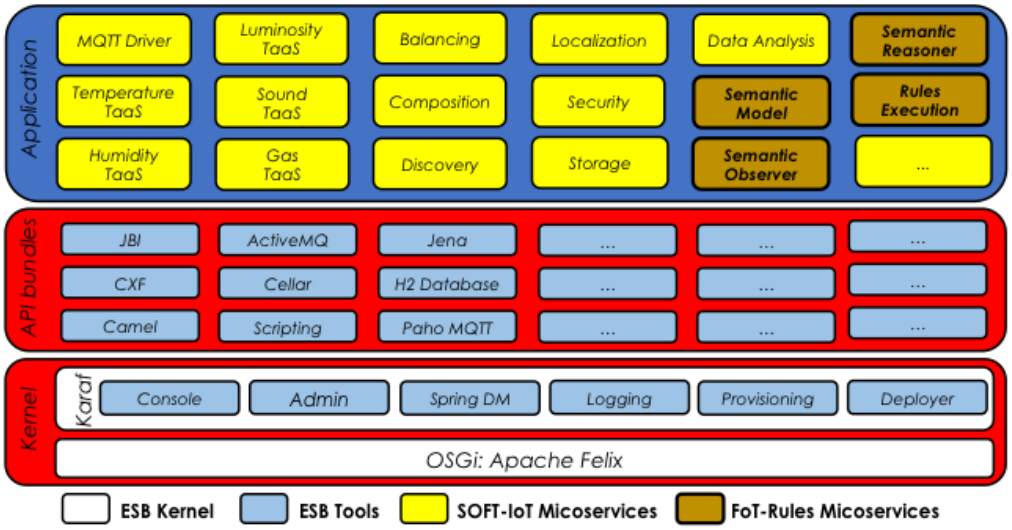

Fig. 3. FoT-Rules modules deployed as microservices in the SOFT-IoT platform.

```
1 <http://wiser.dcc.ufba.br/smartUFBA/devices/ufbaino/#startTimeInterval1491250481152149125054120 7>
2 dul:hasIntervalDate "2017-04-03T20:14:11.054Z" .
3 <http://wiser.dcc.ufba.br/smartUFBA/devices/ufbaino#startTimeInterval1491251112085149125117222 1>
4 dul:TimeInterval ;
5 dul:hasIntervalDate "2017-04-03T20:16:11.207Z" .
6 <http://wiser.dcc.ufba.br/smartUFBA/devices/ufbaino#obsValue_1491251021965149125108208 5>
7 ssn:ObservationValue ;
8 dul:hasDataValue "29"^^xsd:double .
9 dul:isSettingFor false.
```

Listing 3: Excerpt of the RDF triples stored on Server Fuseki.

```
1  Rule (A)
2  [rule1:
3  (?a dul:hasDataValue ?b) greaterThan(?b, 40)
4    -> (?a highTemperature true)];
5
6  Rule (B)
7  [rule2:
8  (?a dul:hasDataValue ?b) greaterThan(?b, 60)
9    -> (?a highHumidity true)
10   LessThan(?b, 40) -> (?a lowHumidity true)];
```

Listing 4: Example of a structural rule in FoT-Rule

4.2.4. *Semantic Observer*

The Semantic Observer[8] module is responsible for observing changes that are made in the model. In order to do that, the Semantic Observer module uses SPARQL queries, as shown in Listing 6. As shown in Listing 3, in line 9, initially, the isSettingFor property is false, and after executing a rule, this property is changed to true by the Semantic Reasoner module (Listing 5). Then, the Semantic Observer module, when identifying changes made in the isSettingFor property, notifies the Rules Execution module (Section 4.2.5), which is responsible for informing the actuator and activating the device (for instance, turn on an air conditioner).

[8]https://github.com/WiserUFBA/Semantic-Observer

```
1 PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn>
2 PREFIX dul:<http://www.loa-cnr.it/.../DUL.owl>
3 PREFIX xsd:<http://www.w3.org/2001/XMLSchema>
4 DELETE { <%s> dul:isSettingFor false .}
5 INSERT { <%s> dul:isSettingFor true .}
6 WHERE { <%s> dul:isSettingFor false .}
```

Listing 5: SPARQL UPDATE for updating the semantic model.

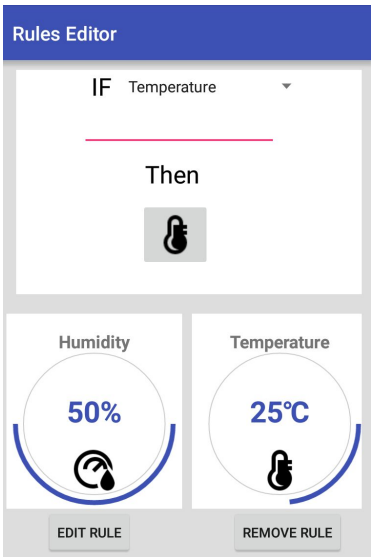

Fig. 4. Mobile application for rules creation.

```
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn>
PREFIX dul: <http://www.loa-cnr.it/.../DUL.owl>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema>
SELECT ?hasDataValue
WHERE{ <http://wiser.dcc.ufba.br/smartUFBA/...>
ssn:ObservationValue ;
dul:hasDataValue '37'^^xsd:double ;
dul:isSettingFor true . };
```

Listing 6: SPARQL query to observe changes made in the semantic model.

4.2.5. *Rules Execution*

The Rules Execution[9] module executes RESTful Web Services as a means to activate actuators (e.g. air conditioner, window, alarm, semaphore, etc.) by sending messages via the MQTT protocol [31].

Listing 7 illustrates the implementation of a RESTful Web Service for the activation of an air conditioner. This Web Service Listing 7 receives, via a POST request in line 4 of Listing 7, a "status" parameter (line 6 of Listing 7) which, in this case, should be on or off. Lines 12 and 13 instantiate a DriverMQTT to enable the communication between the FoT-Gateway and IoT device, i.e., in order to send the "status" (on or off) to the air conditioner. In addition to the POST request presented in Listing 7, we can also implement a POST request that changes other air conditioner functionalities such as the air conditioner temperature.

```
1  @Path("/devices/actuator/airconditioner")
2  public class AirConditioner {
3    public AirConditioner() {}
4    @POST
5    @Produces("application/json")
6    public Response setStatus (@FormParam("status")
7    boolean status) {
8      ResponseBuilder rb;
9      XmlErrorClass x = new XmlErrorClass();
10     AirConditioner ac = new AirConditioner();
11     ac.setStatus(status);
12     try {
13       DriverMQTT airconditioner;
14       airconditioner = new DriverMQTT("relay");
15       airconditioner.setValue("air-conditioner",
16         (ac.getStatus() ? 1 : 0));
17       rb = Response.ok(ac);
18     }catch (Exception e) {
19       x.setStatus(true);
20       rb = Response.ok(x);
21     }
22     return rb.header("Access-Control", "*")
23       .header("Access-Control-Allow-Methods",
24       "GET, POST, DELETE, PUT")
25       .header("Access-Control-Allow-Headers",
26       "Content-Type");
27 }
```

Listing 7: RESTful Web Service implementation for air conditioner.

5. Evaluation

Budgen [32] proposes four quality factors that should be considered when assessing software design quality: reliability, efficiency, maintainability, and usability. FoT-Rules, which has no focus on software maintainability, is based on: i) constrained network and devices in a Fog of Things scenario (reliability); ii) semantic reasoner over semantic rules, which can be lazy (efficiency); iii) generation of rules by the end user (usability). Therefore, we performed three types of evaluation on our FoT-Rules proposal: a reliability evaluation (Section 5.1) - to evaluate FoT-Rules over the FoT constrained scenarios; an efficiency evalua-

[9] https://github.com/WiserUFBA/Rules-Execution

tion (Section 5.2) - to determine if FoT-Rules can execute within an expected time; and a usability evaluation (Section 5.3) - to ascertain if FoT-Rules is useful for end users.

5.1. Reliability Evaluation

In order to prove that our proposal fits in with constrained scenarios as proposed in the Fog of Things paradigm, we execute a reliability evaluation with the objective of checking how FoT-Rules behave from the three perspectives proposed by Budgen [32]:

- **Completeness**: Does FoT-Rules do everything as it is supposed to do?
- **Consistency**: Does FoT-Rules always behave as expected?
- **Robustness**: Does FoT-Rules behave well under abnormal conditions?

Table 2 presents the results of our reliability evaluation in five-different-scenario configuration. For each configuration, we executed the rules thirty times and we varied the value of Humidity and Temperature sensors. In configurations 2 to 5, we simulated some kind of abnormal condition as a means to verify the FoT-Rules behavior under constrained situations.

5.2. Efficiency Evaluation

We present an efficiency evaluation of the communication between gateways and devices. The evaluation involved the FoT-Gateway, FoT-Server, and FoT-Device. In order to perform the evaluation of the efficiency of FoT-Rules, we consider two scenarios:

- **Scenario 1**: We evaluate whether the test time result for two samples will remain constant from different entries and a set of 5 users accessing the infrastructure, as illustrated in Section 4.
- **Scenario 2**: For this evaluation, we used the sending of bulk-rules to be analyzed by the semantic reasoner and semantic observer, simulating the rules built by many users with the objective of analyzing the infrastructure of FoT-Rules in a scenario of stress.

The efficiency evaluation of the architecture presented in Section 4 over the Fog of Things is important to ensure proper functioning during concurrent requests from distinct users.

In Scenario 1, we used two sample sets according to the configuration shown in Tables 3 and 4. We performed 10 test replicates for each sample, and in each replicate we simulated the simultaneous access of 5 users to the proposed architecture using the input parameters, as shown in Tables 3 and 4.Then, for each sample, we collected the time duration of each request and averaged each repetition, as shown in the graph of Figure 5. In this context, time collected comprises the reasoning time of the rule created by the user through the Reasoner module until activating the actuator through the Rules Execution module. Figure 5 shows that the time averages of the two samples are similar. To further justify the efficiency analysis, a Student's t-test was conducted according to the following steps[10]:

- **Hypotheses definition**: We define H_0 (null hypotheses) and H_a (alternative hypothesis). For H_0 the means are the same or the means are significantly different.
- **Homogeneity of variances**: We evaluated the variance of the sample of the two groups and for this we used Fisher's F-test. We obtained p-value (0.9381) greater than 0.05, then we can assume that the two variances are homogeneous.
- **T-Test for homogeneous variances**: We obtained a p-value (0.7929) greater than 0.05, then we concluded that the averages of the two groups demonstrated no significant difference (cannot reject H_0).

In Scenario 2, the testing was conducted from a 100-thread simulation executed concurrently. The results were computed in box plot (Figure 6) to evaluate the performance of the middleware in response time requirement. The testing was conducted with 10 steps.

In each step, we changed the input values and registered the time for execution. Figure 6 shows the empiric distribution of time of execution. The efficiency analysis shows that the median for performance with 100 tasks executed concurrently is 31 seconds. The minimum execution time has a value of approximately 20 seconds, while the maximum execution time has a value of approximately 37 seconds. In this scenario, we verified that the requests were met without any errors in the infrastructure.

We calculated the confidence interval for the results (i.e. average = 30.8, population standard deviation = sample size = 10 and 5% significance, that is, with 95% in a confidence interval) obtained for the second

[10] http://rpubs.com/cleberlira/EfficiencyEvaluationTtest

Table 2
Results for Reliability evaluation.

Configurations	Inputs	Repetitions	Abnormal Conditions	Results
1	Humidity = 50 Temperature = 36	30	No	Completeness=ok Consistency=100% Robustness=100%
2	Humidity = 55 Temperature = 25	30	Yes, observer module stopped 2 internal errors on server	Completeness=ok Consistency=100% Robustness=93.34%
3	Humidity = 60 Temperature = 27	30	Yes, reasoner module stopped	Completeness=ok Consistency=100% Robustness=100%
4	Humidity = 61 Temperature = 32	30	Yes, module rules execution stopped	Completeness=ok Consistency=100% Robustness=100%
5	Humidity = 38 Temperature = 24	30	Yes, Device off	Completeness=ok Consistency=100% Robustness=100%

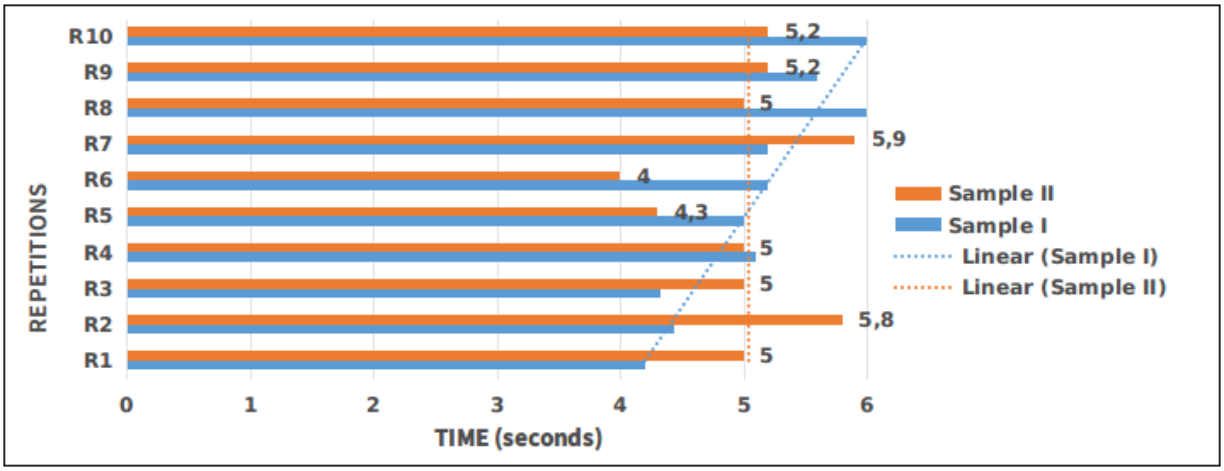

Fig. 5. Time for Sample I and Sample II.

scenario. It was then possible to significantly estimate that in a possible simultaneous access of FoT-Rules by 100 users, the response time would be between 26.97 seconds and 34.62 seconds.

5.3. Usability Evaluation

We performed a usability assessment, instantiating the scenario described in Section 3.2, based on the USE questionnaire technique developed by Lund [33]. Table 5 presents a description of all the questions used in the usability evaluation performed for this work. The USE technique suggests that users evaluate applications by means of four dimensions: (i) Usefulness; (ii) Ease of use; (iii) Ease of learning; and (iv) Satisfaction. For this reason, the evaluation performed in this study was divided into these four dimensions.

In order for users to perform the evaluation, we developed an HTML form containing the questionnaire presented in Table 5 and made it available on the Web. In addition, we provided an executable file of Rules

Table 3
Configuration for Scene 1 - Sample 1.

Repetition	Inputs
1	Humidity = 30-32-34-76-54 Temperature = 35-32-31-30-32
2	Humidity = 45-46-47-54-65 Temperature = 23-43-44-43-32
3	Humidity = 21-65-56-57-65 Temperature = 34-38-39-45-46
4	Humidity = 61-65-45-63-46 Temperature = 37-38-21-20-43
5	Humidity = 58-65-63-62-56 Temperature = 24-26-32-30-31
6	Humidity = 32-34-36-45-54 Temperature = 45-48-45-56-32
7	Humidity = 48-49-53-52-56 Temperature = 44-35-33-32-21
58	Humidity = 67-47-56-45-56 Temperature = 24-32-31-30-29
9	Humidity = 50-47-48-56-46 Temperature = 35
10	Humidity = 45-46-48-56-57 Temperature = 28-32-33-44-23

Table 4
Configuration for Scene 1 - Sample 2.

Repetition	Inputs
1	Humidity = 65-56-47-48-57 Temperature = 26-28-34-22-40
2	Humidity = 67-44-55-46-48 Temperature = 35-39-43-21-19
3	Humidity = 30-33-44-32-58 Temperature = 43-42-25-27-17
4	Humidity = 61-59-58-42-44 Temperature = 33-39-32-21-22
5	Humidity = 68-58-47-55-54 Temperature = 34-35-32-17-45
6	Humidity = 58-56-32-44-46 Temperature = 44-20-30-17-19
7	Humidity = 48-45-43-23-45 Temperature = 24-23-32-34-21
8	Humidity = 54-56-54-53-60 Temperature = 22-27-28-29-39
9	Humidity = 56-59-60-54-55 Temperature = 44-32-44-14-20
10	Humidity = 54-55-56-57-56 Temperature = 32-34-38-39-40

Table 5
USE Questionnaire: Usefulness, Satisfaction, Ease of Learning, and Ease of Use.

DIMENSION	**STATEMENT**
USEFULNESS	The Rules Editor application is useful.
	The Rules Editor application helps me be more productive when building smart services in my building.
	The Rules Editor application saves me time when I use it.
	The Rules Editor application meets the control needs of temperature and humidity of the building.
	The Rules Editor application does everything I would expect it to do about control of temperature and humidity of the building
EASE OF USE	The Rules Editor application is easy to use.
	The Rules Editor application is simple to use.
	The Rules Editor application requires the least number of possible steps to accomplish what I want to do with it.
	The Rules Editor application is flexible.
	The Rules Editor application is user-friendly.
	Using the Rules Editor application does not require much effort.
	I can use the Rules Editor application without written instructions.
	I have not noticed any inconsistencies when using the Rules Editor application.
	In my opinion, both regular and occasional users would enjoy using the Rules Editor application.
	By using the Rules Editor application, I can recover from errors quickly and easily.
	I was able to use the Rules Editor application successfully every time.
EASE OF LEARNING	I learned how to use the Rules Editor application Editor
	It is easy to learn how to use the Rules-Editor application. Editor
	I became able to use the Rules Editor application quickly.
	I easily remember how to use Rules Editor.
SATISFACTION	I am satisfied with the Rules Editor application.
	I would recommend the Rules Editor application to a friend.
	It is nice to use the Rules Editor application.
	Rules Editor works the way I want it to work.
	Rules Editor is wonderful.
	I feel I need to have Rules Editor
	Rules Editor is pleasant to use.

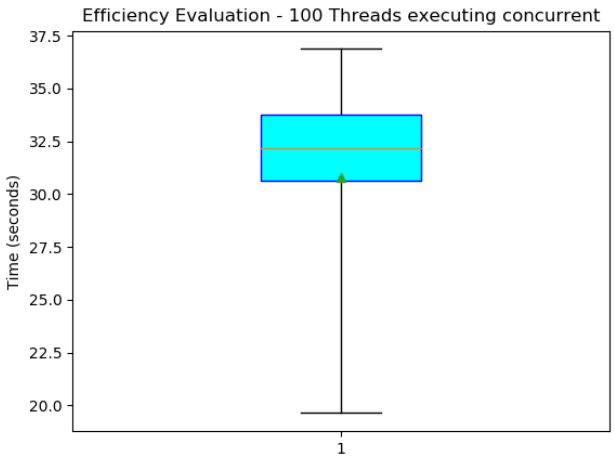

Fig. 6. Efficiency Evaluation.

Editor [11] for users to carry out the installation of the application.

Usability researchers discuss the number of participants required to perform system usability analyses. In our proposal, we consider the studies discussed by Six and Macefield [34] and Nielsen [35], in which the authors consider between fifteen and twenty people as the ideal number for identifying usability problems (i.e. discover over 80% of the problems), since by adding other participants to perform the tests, most of the time the same problems would be identified, thus little new relevant information regarding usability tests would result.

Fifteen users with different backgrounds (e.g. social assistant, language teacher, Information Technology student, Gastronomy student, and nursing technician) used the Rules Editor and then ranked each question presented in Table 5 in a 7-point Likert Rating scale [36] (ranging from “totally disagree” to “totally agree”).

We extracted the answers and generated the graphs presented in Figures 7, 8, 9, 10, which summarize the usefulness, ease of use, ease of learning and satisfaction, respectively, of the rules editor of FoT-Rules as informed by users. In these dimensions, we consider ratings between 5 and 7 as “good acceptance”, ratings between 1 and 3 as “rejection” and ratings of 4 as “neutral acceptance”.

The result for the first dimension (usefulness) of the evaluation is shown in the graph of Figure 7 and it shows that users rated the FoT-Rules proposal with the use of Rules Editor as useful.

On average, approximately 98% of users reported good acceptance. If we consider only the first four statements of the usefulness dimension, the rate of good acceptance increases to 100%, which means that the best ratings for usefulness were given for statements related to efficiency and productivity aspects.

The result of the second dimension (ease of use), as shown in the graph on Figure 8 shows that most users rated the Rules Editor as easy to use. On average, approximately 90% of the users reported good acceptance and they rated statements such as (i) "The Rules Editor application is simple to use" and (ii) "In my opinion, both regular and occasional users would enjoy using the Rules Editor application" with an even better rating.

The third dimension (ease of learning) of the assessment is presented in Figure 9. The graph there shows that only three users did not learn how to use the Rules Editor application in order to create ECA rules in FoT-Rules.

The dimension (satisfaction) of the remote evaluation is presented in Figure 10. There, the graph shows that on average, approximately 87% of users were satisfied with the Rules Editor and would recommend it to a friend.

6. Discussion and Future Works

By conducting this work, we have noticed some gaps that are opportunities to be studied as a menas to improve the creation of Smart Spaces at the edge of the network. Thus, we have identified these areas for improvement based on the lessons learned:

- **Instantiating FoT-Rules in other domains**: In a future work, we will instantiate FoT-Rules in other scenarios and domains in order to demonstrate and evaluate how extensible FoT-Rules can be.
- **Discovering Rules**: To create a module that can provide a discovery mechanism in order to retrieve specific rules according to sensor types. For example, a user selects a temperature sensor in her/his application, the discovery mechanism then allows her/him to retrieve all rules related to temperature sensors at the edge of the network. This module can further support the automatic construction of semantic rules from the

[11] https://github.com/WiserUFBA/Rules-Editor

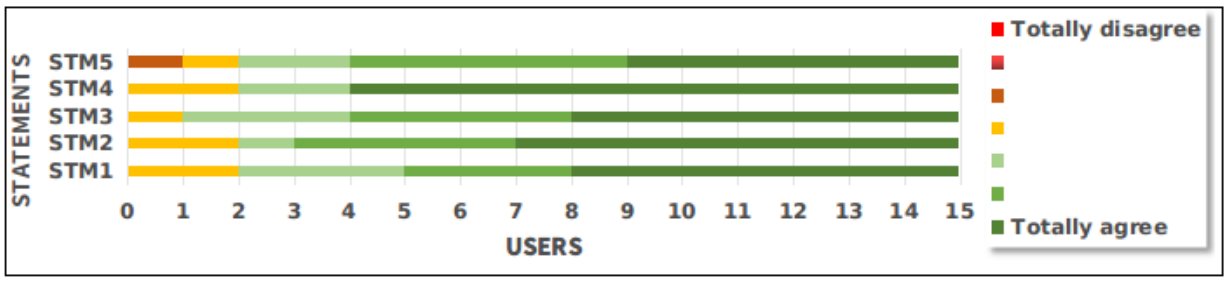

Fig. 7. Results of usefulness questionnaire.

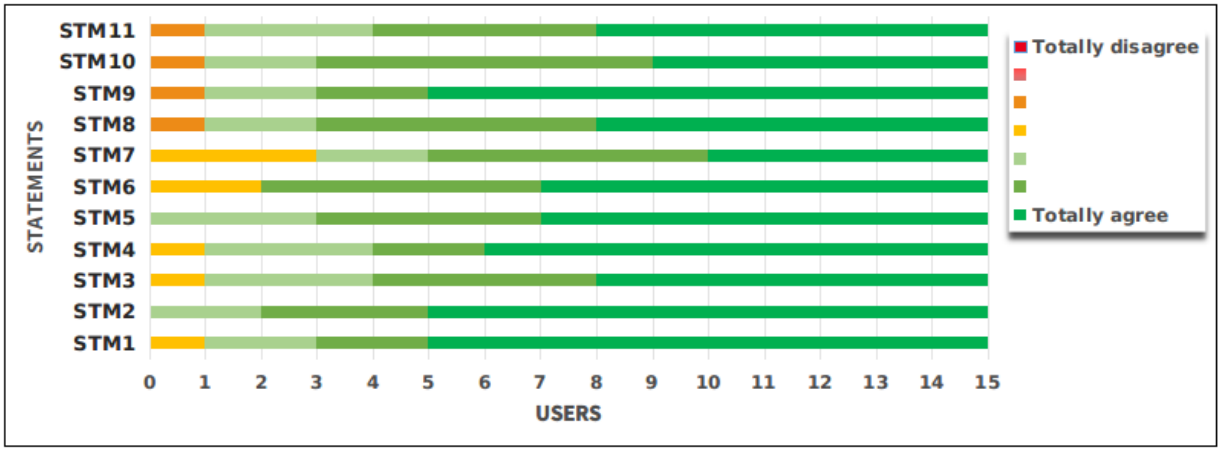

Fig. 8. Results of ease of use questionnaire.

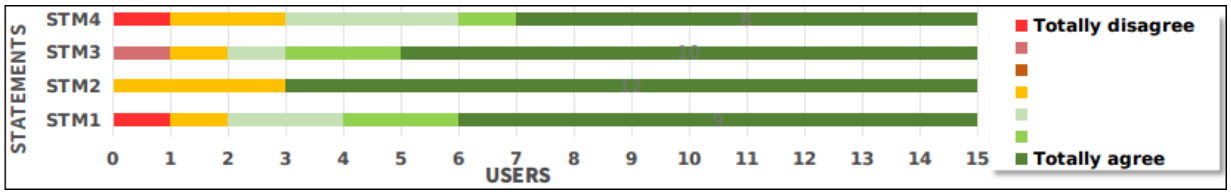

Fig. 9. Results of ease of learning questionnaire.

analysis of data generated in the scenarios. Therefore, that FoT-Rule can deduce the information and perform actions that promote end-user satisfaction.

– **Privacy and Security**: Maintaining end-user privacy and establishing a secure environment are fundamental issues in the Internet of Things and, as a consequence, also in our discussion of building intelligent scenarios at the edge of the network. In FoT-Rules, as sensors gain access to information, users personal information may be exposed. Therefore, a solution for the authentication and authorization of information should be considered in a future work.
– **Microservices**: In a recent study, Santana, Alencar and Prazeres [37, 38] stated that the Microservices architectural style is being applied in several domains such as the IoT. Microservices enables the creation of a system from a collection of small and isolated services capable of managing their own data [39]. When compared to traditional architectures for designing applications

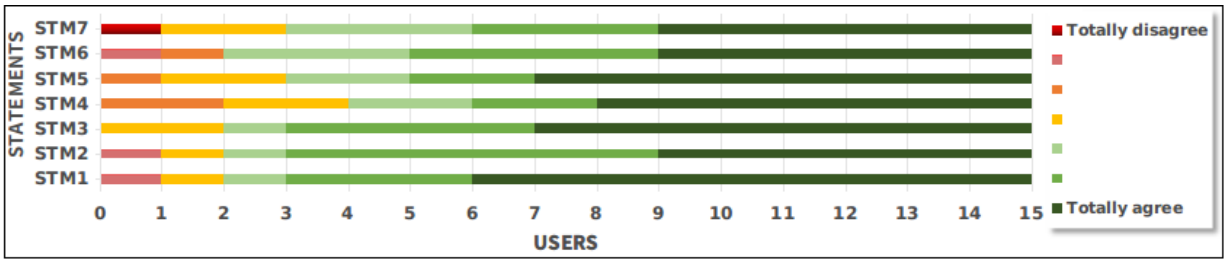

Fig. 10. Results of satisfaction questionnaire.

in IoT, Microservices architecture offers advantages such as those discussed by Newman [40]: i) Technology Heterogeneity; ii) Resilience; iii) Scalability; iv) Ease of Deployment; v) Organizational Alignment; vi) Composability. These advantages enable the development of large-scale IoT applications. In addition, Santana, Alencar and Prazeres [37, 38] emphasis that other fields of research associated with Microservices will be explored as Reactive Systems. Reactive Systems are built according to the following characteristics: if possible, the system will be (i) **responsive** in a timely and (ii) **resilient** manner, and will continue to respond in case of failure; and (iii) **elastic**, the system will respond according to the variation of demand. To have these characteristics, the system must use asynchronous messages in order to guarantee low coupling, isolation and location transparency [41]. According to Escoffier and Clement [42], Microservices that have these characteristics are called Reactive Microservices. In FoT-Rules, Reactive Microservices can be applied to improve the elasticity and resilience at the application level.

7. Conclusion

In Smart Spaces where the amount and diversity of devices connected to the Internet are constantly increasing, it is fundamental that semantic data be used to ease integration with the already existing devices as well as devices that will be created in the foreseeable future. Furthermore, the participation of the end user in the construction of Smart Spaces brings innumerable benefits as well as the possibility of personalizing the final solution. However, the solutions identified in the literature for the construction of rules in Smart Spaces are based on cloud solutions. These solutions might have problems related to latency and delay experienced over the Internet. The evaluations have made it possible to illustrate, from different aspects and perspectives (i.e. Reliability, Efficiency, Usability), the reliability of the architecture proposed in this work. In this context, this work has presented the FoT-Rules, an approach based on semantic rules for the construction of Smart Spaces in the Fog of Things, which considers the usage of all processing and capacity of devices located at the network edge and, as a result, allows for the development of Smart Spaces nearer to the end user.

References

[1] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, Internet of Things (IoT): A vision, architectural elements, and future directions, *Future generation computer systems* **29**(7) (2013), 1645–1660.

[2] A. Bikakis and G. Antoniou, Rule-Based Contextual Reasoning in Ambient Intelligence, in: *Semantic Web Rules: International Symposium, RuleML 2010, Washington, DC, USA, October 21-23, 2010. Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 74–88. ISBN 978-3-642-16289-3. doi:10.1007/978-3-642-16289-3_8.

[3] M. Weiser, The computer for the 21st century, *Scientific American* **265**(3) (1991), 94–104.

[4] 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* **76**(4) (2017), 5221–5241, ISSN 1573-7721. doi:10.1007/s11042-016-3511-2.

[5] M. Pahl, G. Carle and G. Klinker, Distributed smart space orchestration, in: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 979–984, ISSN 2374-9709. doi:10.1109/NOMS.2016.7502936.

[6] L. Mainetti, V. Mighali, L. Patrono and P. Rametta, A novel rule-based semantic architecture for IoT building automation systems, in: *2015 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2015, pp. 124–131. doi:10.1109/SOFTCOM.2015.7314063.

[7] T. Tuomisto, T. Kymäläinen, J. Plomp, A. Haapasalo and K. Hakala, Simple Rule Editor for the Internet of Things, in: *2014 International Conference on Intelligent Environments*, 2014, pp. 384–387. doi:10.1109/IE.2014.72.

[8] C.Y. Leong, A.R. Ramli and T. Perumal, A rule-based framework for heterogeneous subsystems management in smart home environment, *IEEE Transactions on Consumer Electronics* **55**(3) (2009), 1208–1213, ISSN 0098-3063. doi:10.1109/TCE.2009.5277977.

[9] S. Valtolina, B.R. Barricelli and M. Mesiti, End-User Centered Events Detection and Management in the Internet of Things, in: *Current Trends in Web Engineering*, F. Daniel and O. Diaz, eds, Springer International Publishing, Cham, 2015, pp. 77–90. ISBN 978-3-319-24800-4. doi:10.1007/978-3-319-24800-4_7.

[10] C.E. Kaed, I. Khan, A.V.D. Berg, H. Hossayni and C. Saint-Marcel, SRE: Semantic Rules Engine for the Industrial Internet-Of-Things Gateways, *IEEE Transactions on Industrial Informatics* **14**(2) (2018), 715–724, ISSN 1551-3203. doi:10.1109/TII.2017.2769001.

[11] B. Wang, X. Guo, M. Ester, Z. Guan, B. Singh, Y. Zhu, J. Bu and D. Cai, Device-Aware Rule Recommendation for the Internet of Things, in: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, ACM, New York, NY, USA, 2018, pp. 2037–2045. ISBN 978-1-4503-6014-2. doi:10.1145/3269206.3272009.

[12] N.W. Paton, F. Schneider and D. Gries (eds), *Active Rules in Database Systems*, 1st edn, Springer-Verlag, Berlin, Heidelberg, 1998. ISBN 0387985298.

[13] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications, *IEEE Communications Surveys Tutorials* **17**(4) (2015), 2347–2376, ISSN 1553-877X. doi:10.1109/COMST.2015.2444095.

[14] V. Huang and M.K. Javed, Semantic Sensor Information Description and Processing, in: *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*, 2008, pp. 456–461. doi:10.1109/SENSORCOMM.2008.23.

[15] H.S. Choi, J.Y. Lee, N.R. Yang and W.S. Rhee, User-centric service environment for context aware service mash-up, in: *2014 IEEE World Forum on Internet of Things (WF-IoT)*, 2014, pp. 388–393. doi:10.1109/WF-IoT.2014.6803197.

[16] A. Gyrard, M. Serrano, J.B. Jares, S.K. Datta and M.I. Ali, Sensor-based Linked Open Rules (S-LOR): An Automated Rule Discovery Approach for IoT Applications and Its Use in Smart Cities, in: *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, Republic and Canton of Geneva, Switzerland, 2017, pp. 1153–1159. ISBN 978-1-4503-4914-7. doi:10.1145/3041021.3054716.

[17] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, Fog Computing and Its Role in the Internet of Things, in: *Proceedings of the First Workshop on Mobile Cloud Computing*, ACM, 2012, pp. 13–16. ISBN 978-1-4503-1519-7. doi:10.1145/2342509.2342513.

[18] M. Abdelshkour, IoT, from Cloud to Fog Computing, 2015, [Online; accessed 26-November-2018].

[19] D. Amaxilatis, O. Akrivopoulos, I. Chatzigiannakis and C. Tselios, Enabling stream processing for people-centric IoT based on the fog computing paradigm, in: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1–8, ISSN 1946-0759. doi:10.1109/ETFA.2017.8247674.

[20] F. Bonomi, R. Milito, P. Natarajan and J. Zhu, Fog Computing: A Platform for Internet of Things and Analytics, in: *Big Data and Internet of Things: A Roadmap for Smart Environments*, N. Bessis and C. Dobre, eds, Studies in Computational Intelligence, Vol. 546, Springer International Publishing, 2014, pp. 169–186. ISBN 978-3-319-05028-7. doi:10.1007/978-3-319-05029-4_7. http://dx.doi.org/10.1007/978-3-319-05029-4_7.

[21] E. Batista, G. Figueiredo, M. Peixoto, M. Serrano and C. Prazeres, Load Balancing in the Fog of Things Platforms Through Software-Defined Networking, in: *2018 IEEE International Conference on Computer and Information Technology (CIT)*, 2018, pp. 1785–1791.

[22] C. Prazeres and M. Serrano, SOFT-IoT: Self-Organizing FOG of Things, in: *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2016, pp. 803–808. doi:10.1109/WAINA.2016.153.

[23] H. Takatsuka, S. Saiki, S. Matsumoto and M. Nakamura, Design and Implementation of Rule-Based Framework for Context-Aware Services with Web Services, in: *Proceedings of the 16th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '14, ACM, New York, NY, USA, 2014, pp. 233–242. ISBN 978-1-4503-3001-5. doi:10.1145/2684200.2684310.

[24] V. Kumar, A. Fensel and P. Fröhlich, Context Based Adaptation of Semantic Rules in Smart Buildings, in: *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, IIWAS '13, ACM, New York, NY, USA, 2013, pp. 719–719719728. ISBN 978-1-4503-2113-6. doi:10.1145/2539150.2539174.

[25] S. Taherizadeh, V. Stankovski and M. Grobelnik, A Capillary Computing Architecture for Dynamic Internet of Things: Orchestration of Microservices from Edge Devices to Fog and Cloud Providers, *Sensors* **18**(9) (2018), ISSN 1424-8220. doi:10.3390/s18092938. http://www.mdpi.com/1424-8220/18/9/2938.

[26] I. Froiz-Míguez, T.M. Fernández-Caramés, P. Fraga-Lamas and L. Castedo, Design, Implementation and Practical Evaluation of an IoT Home Automation System for Fog Computing Applications Based on MQTT and ZigBee-WiFi Sensor Nodes, *Sensors* **18**(8) (2018), ISSN 1424-8220. doi:10.3390/s18082660. http://www.mdpi.com/1424-8220/18/8/2660.

[27] OpenHAB, OpenHAB, 2018, [Online; accessed 18-November-2018].

[28] C. Prazeres, J. Barbosa, L. Andrade and M. Serrano, Design and Implementation of a Message-service Oriented Middleware for Fog of Things Platforms, in: *Proceedings of the Symposium on Applied Computing*, ACM, 2017, pp. 1814–1819. ISBN 978-1-4503-4486-9. doi:10.1145/3019612.3019820.

[29] N.R. Sousa and C. Prazeres, M2-FoT: a Proposal for Monitoring and Management of Fog of Things Platforms, in: *2018 IEEE Symposium on Computers and Communications (ISCC)*, 2018, pp. 01038–01043, ISSN 1530-1346. doi:10.1109/ISCC.2018.8538560.

[30] L. Andrade, M. Serrano and C. Prazeres, The Data Interplay for the Fog of Things: A Transition to Edge Computing with IoT, in: *2018 IEEE International Conference*

on Communications (ICC), 2018, pp. 1–7, ISSN 1938-1883. doi:10.1109/ICC.2018.8423006.

[31] E. Batista, L. Andrade, R. Dias, A. Andrade, G. Figueiredo and C. Prazeres, Characterization and Modeling of IoT Data Traffic in the Fog of Things Paradigm, in: *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, 2018, pp. 1–8. doi:10.1109/NCA.2018.8548340.

[32] D. Budgen, *Software Design*, 2nd edn, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003. ISBN 0201722194.

[33] A.M. Lund, Measuring usability with the use questionnaire, *Usability Interface* **8**(2) (2001), 3–6.

[34] J.M. Six and R. Macefield, How to Determine the Right Number of Participants for Usability Studies, UXmatters, 2016, [Online; accessed 26-November-2018].

[35] J. Nielsen, How Many Test Users in a Usability Study?, Nielsen Norman Group, 2012, [Online; accessed 26-November-2018].

[36] R. Likert, A technique for the measurement of attitudes., *Archives of Psychology* **22**(140) (1932), 1–55.

[37] C. Santana, B. Alencar and C. Prazeres, Microservices: A Mapping Study for Internet of Things Solutions, in: *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, 2018, pp. 1–4. doi:10.1109/NCA.2018.8548331.

[38] C. Santana, B. Alencar and C. Prazeres, Reactive Microservices for the Internet of Things: A case study in Fog Computing (TO APPEAR), in: *Proceedings of the Symposium on Applied Computing*, ACM, 2019, pp. 1–9.

[39] M. Fowler and J. Lewis, Microservices, 2014, [Online; accessed 26-November-2018].

[40] S. Newman, *Building microservices: designing fine-grained systems*, 1st edn, O'Reilly Media, 2015.

[41] J. Bonér, D. Farley, R. Kuhn and M. Thompson, The Reactive Manifesto, 2014, [Online; accessed 26-November-2018].

[42] C. Escoffier, *Building Reactive Microservices in Java: Asynchronous and Event-Based Application Design*, 1st edn, O'Reilly Media, 2017. ISBN 978-1-491-98626-4.