The Journal of Systems and Software 161 (2020) 110494

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

In Practice

LAURA architecture: Towards a simpler way of building situation-aware and business-aware IoT applications

Sergio Teixeira[a,c,*], Bruno Alves Agrizzi[a], José Gonçalves Pereira Filho[a], Silvana Rossetto[b], Isaac Simões Araújo Pereira[a], Patrícia Dockhorn Costa[a], Adriano Francisco Branco[d], Ruan Rocha Martinelli[a]

[a] Departamento de Informática, Universidade Federal do Espírito Santo (UFES), Vitória, ES, Brasil
[b] Departamento de Ciência da Computação, Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, RJ, Brasil
[c] Centro Universitário Católico de Vitória (UCV), Vitória, ES, Brasil
[d] Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, RJ, Brasil

ARTICLE INFO

Article history:
Received 11 February 2019
Revised 1 December 2019
Accepted 8 December 2019
Available online 9 December 2019

Keywords:
Internet of things
IoT Architecture
Wireless sensor networks
Situation-Awareness
Business Process

ABSTRACT

The explosion of smart objects made companies rethink their Business Model (BM) using Wireless Sensor Networks (WSN) and the Internet of Things (IoT) aiming to improve their Business Processes (BP) to achieve competitiveness. Business environments are complex due to the wide variety of technologies, hardware and software solutions that compose heterogeneous enterprise environments. On the other hand, putting real-world IoT scenarios into practice is still a challenge for even experienced developers, because it requires low-level programming skills and, at the same time, specific domain knowledge of a company's BM. This research paper proposes LAURA – **L**ean **AU**tomatic code generation for situation-aware and business-awa**R**e **A**pplications, a flexible, service-oriented and general open-source conceptual architecture, designed to support the deployment of decoupled IoT applications. Empirical evaluation has shown that LAURA simplifies the development of final Situation-Aware or Business-Aware applications, reducing the need for specialized IoT low-level knowledge, while showing an acceptable performance. LAURA also provides the freedom and independence to modify, adapt or integrate its architecture according to specific needs of the stakeholders.

1. Introduction

The influence of the *Internet of Things* (IoT) is, for many, already an everyday reality and it is expected to increase, contributing ubiquitously with applications that can improve the quality of the environment and the automation of corporative processes in a wide range of knowledge areas. The explosion of smart objects has already been highlighted by many companies and research institutions. The International Data Corporation (IDC) report, for example, predicts that the IoT market will be worth around $1.7 trillion in 2020 (IDC, 2017, 2015). According to the Gartner Group, by 2020 more than half of the major new companies will have some elements of the IoT incorporated into their Business Processes (BP) and systems (Plummer et al., 2015; Stamford, 2016). A panorama of 500 billion connected devices across the world, by 2030, is expected (Oxford Economics and Cisco, 2017).

Weiser (1999) envisioned a world full of small intelligent objects that work in a transparent and omnipresent way with the aim of improving people's lives by making them more aware of events surrounding them. One of the key elements in putting this vision into practice is the rise of Wireless Sensor Networks (WSN) of tiny devices with processing, storage, communication and sensing capacities, which provide a way to monitor physical world magnitudes at a low cost (Akyildiz et al., 2002; Yick et al., 2008).

WSN has evolved in recent years and it is now possible to easily connect their nodes with the Internet, enabling the concept of IoT in practice. The precision capillarity, low cost, small size, Internet-based communication means that today there is a great potential for many types of IoT applications that were unimaginable at the time Weiser made his prediction (Su et al., 2011). Fig. 1 presents some examples of IoT real-world application scenarios.

The IoT paradigm has made companies change or rethink their Business Model (BM), making IoT solutions an integral part of this process. In our previous work (Teixeira et al., 2017b), we have verified that since 2012 there has been an increase in the demand from companies to integrate their IoT solutions into Business

* Corresponding author at: Departamento de Informática, Universidade Federal do Espírito Santo (UFES), Vitória, ES, Brasil.
E-mail address: sergio@multicast.com.br (S. Teixeira).

https://doi.org/10.1016/j.jss.2019.110494
0164-1212/

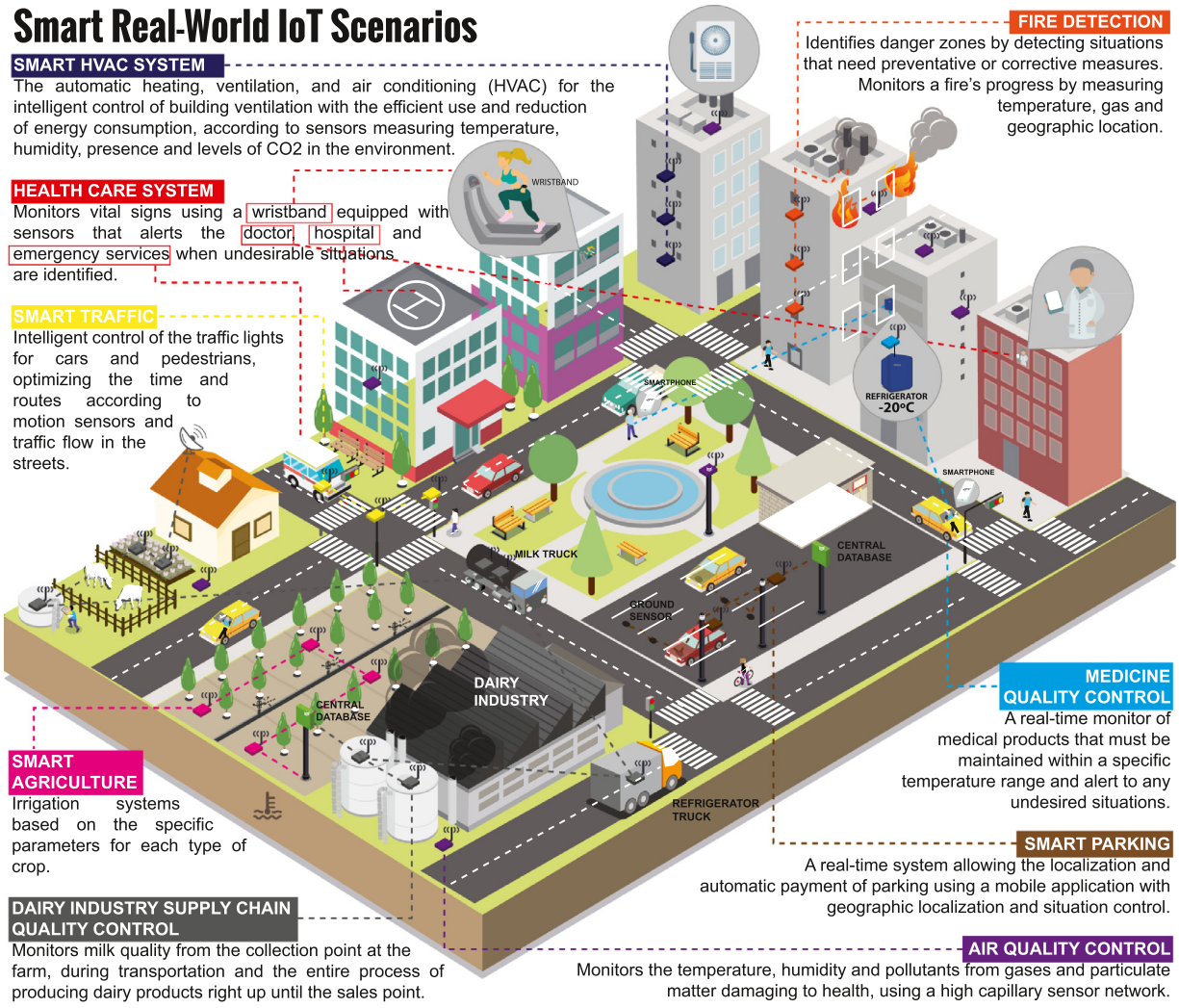

Fig. 1. Smart real-world IoT scenario.

Process Management (BPM) systems. By including sensing capabilities in the BP, companies are able to improve their ability to monitor and respond to changes in real-time, thus giving their business a competitive advantage. In other words, companies are taking advantage of the benefits that smart devices offer to improve and add value to their BP.

Meanwhile, putting real-world IoT scenarios into practice is still a challenge even for experienced developers. Developing and implementing IoT applications require low-level programming skills and, at the same time, specific domain knowledge of a company's BM. In addition to the specific aspects related to IoT devices, it is necessary to integrate and balance a solution with the heterogeneous computational environment within a company. This heterogeneity increases the complexity of deciding which solution is best for a company to adopt.

Usually, the development and deployment of IoT solutions demand a multidisciplinary team due to the complexity and heterogeneity of certain aspects related to the IoT itself, the domain knowledge of the application, and the specific characteristics of the company's BP (Baldam et al., 2014; Rodrigues et al., 2015; Tranquillini et al., 2012).

Thus, one of the greatest challenges in this area is proposing solutions that are able to simplify the entire IoT application development and deployment process, taking into account the need to involve professionals from different knowledge domains. For instance, Domain experts, System experts, BPM experts, Situation/Rule experts, and IoT experts. The Domain expert, for example, who has specific knowledge in the area that the solution will be applied to, is not usually involved in the development of the IoT solution. It would be beneficial, therefore, if the Domain expert contributes to various aspects during the development process, which would ultimately improve the quality of solution.

In this research paper, the term 'final application' (Ayala et al., 2015; Gyrard et al., 2015; Xi and Yuzhi, 2014) refers to the applications that will be used by final consumers, such as the Domain experts. A type of 'final application' addressed in this study is a Business-Aware application that is based on Business Process Management Suites (BPMS) (Gartner, 2017). BPMS are systems that support automated process management. BizAgi software (Bizagi, 2018) and jBPM (Redhat, 2015) are examples of BPMS.

From the software design standpoint, one commonly used way to address the issues raised above is to adopt "separation of concerns" methods (Almeida, 2006; Doddapaneni et al., 2012; Mellor et al., 2004). These methods simplify modeling activity when one has to deal with many technical issues that must be addressed at different levels of abstraction. The separation of concerns facilitates the interaction among distinct stakeholders during the entire process of development and deployment of complex applications. It ensures that the decoupling of the layers is possible, preventing

the higher-level stakeholders from needing to concern themselves with the lower-level technical details.

From the deployment standpoint, one widely used way to allow flexibility in the final applications and to provide standardized interfaces and full decoupling of the low-level aspects is through Service-Oriented Architecture (SOA) (Papazoglou et al., 2008), an architectural style for software development. SOA is currently one of the preferred options in major companies to develop and integrate their final application with the IoT solution.

Another trend that has been on the rise in recent years is the use of Rule-based systems (Murray et al., 2016) in the corporate environment as a strategy to respond faster to changes that could occur in the business model or the application requirements. A rule-based (RB) system is easier to understand and maintain if compared with imperative programming and may increase the involvement of professionals that work within the definition of the business rules (Flasiński, 2016).

In addition to the increasing adoption of RB approaches, IoT solutions can benefit from Complex Event Processing (CEP) and Stream Processing (SP) techniques (Aliverti et al., 2016; I. Gartner, 2017). The CEP paradigm provides a solid model and toolset to reason over a near-uninterrupted flow of pieces of information coming from decoupled sources (sensors, mobile devices, web services, etc). It allows us to build upon primitive events towards complex event patterns by means of event data extraction and composition through explicit causal, temporal or spatial relations (Flouris et al., 2017).

One of the key capabilities of a reactive and proactive IoT application is the bridging of the gap between events that occur in the environment and the particular state-of-affairs of interest also known as situations (Pereira et al., 2013). It is upon these that the application is required to act on or react to. Within the scope of application development, this capability has often been referred to as context-awareness (Dey, 2001) or situation-awareness. This terminology has been taken from human factors research (Endsley, 1995). The aim of situation-aware applications is to promote effective interaction with users by autonomously adapting application behavior according to the user's current and projected situation. As discussed in (Costa et al., 2016), in order to leverage the benefits of the situation abstraction concept, proper support is required at design-time to specify situation types, and run-time support that can detect and maintain information regarding situations.

Results obtained from our previous work (Teixeira et al., 2017b), reveal that there are few solutions or architectures that deal with the entire process from lower-level to higher-level aspects of supporting the development of decoupled real-world IoT applications, particularly providing integrated mechanisms for: (i) situation-awareness handling; (ii) automatic code generation from high-level business process specifications (e.g., BPMN descriptions); (iii) detection of changes in resource state through different event-driven methods; (iv) bytecode dissemination with a view to rapid and low-cost application reprogramming; (v) straightforward domain reference model to promote a common ground of understanding and a representation of meaningful IoT real-world concepts which need to be modeled in the application software.

This study presents LAURA – **L**ean **AU**tomatic code generation for situation-awa**R**e **A**pplication, a service-oriented architecture to support the development, deployment, and execution of situation-aware or business-aware IoT applications, which realize the set of integrated facilities listed above. LAURA components operate in coordinated ways to provide high-level stakeholders with an easier process for developing dynamic, real-world IoT applications in scenarios such as those shown in Fig. 1.

The remainder of this paper is organized in the following manner: Section 2 presents the domain challenges; Section 3 presents the related work and system requirements; Section 4 presents LAURA domain reference model, the conceptual proposed architecture, and the applied architecture view; Section 5 evaluates LAURA by presenting and discussing an application scenario, two modeling applications, a performance evaluation, and an empirical evaluation of LAURA; Section 6 concludes the paper and highlights where future studies can be developed.

2. Domain challenges

Fig. 1. highlights relevant IoT scenarios with their respective domain challenges. These IoT scenarios were selected based on their relevance. Smart HVAC systems, for example, are highlighted due to energy concerns. The use of IoT solutions in such systems can reduce investment costs by up to 40% and allow energy efficiency by between 7% and 35% (Acciona, 2013). The demand for IoT and smart technologies in the HVAC industry is growing due to their potential to increase energy efficiency, reduce costs and improve comfort (Kingatua, 2017). HVAC systems equipped with IoT technologies offer numerous benefits to building owners, occupants and facility managers by allowing users to automatically monitor and control a wide range of variables, such as temperature, water usage, lighting, occupancy, and CO^2 levels, as well as to provide real-time status of almost all components.

Healthcare systems are also a matter of interest. The world health spending is forecast to exceed $18 trillion by 2040 (IHME, 2016; murabet et al., 2018; Townsend, 2019). The use of IoT technologies in healthcare and medical sectors has brought significant improvement to medical services, including: reducing emergency room waiting times, ensuring the availability and accessibility of critical hardware, addressing chronic diseases, tracking staff, patients and inventory, and enhancing drug management, among other benefits (Matthews, 2018). IoT technologies can also contribute to the implementation of remote monitoring solutions such as Ambient Assisted Living (AAL) systems through patient remote monitoring and follow-up in harmony with the treatment of patients who prefer or need to live at home.

Smart agriculture, another relevant IoT application scenario (highlighted in Fig. 1) is a worldwide issue, a popular research topic in the farming industry and academia, and also a more recent subject of concern. The global population is set to reach 9.7 billion by 2050 (Nations, 2019). To feed this increase in population, the farming industry must therefore fully embrace new technologies such as IoT, which have the potential to improve and transform agriculture in many aspects. Data collected by smart agriculture sensors such as weather conditions, soil quality, crop growth progress, and cattle health, can enable growers and farmers to reduce waste and enhance productivity. IoT and smart technologies can assist farmers and producers in finding new approaches to driving efficiency and improving operations through the use of connected devices, such as RFID tags and wearables, and the application of predictive analytics on real-time data.

The following paragraphs discuss Domain Challenges (DC) that are common to the development of smart real-world applications in these three examples of relevant IoT scenarios as well as those scenarios depicted in Fig. 1.

Domain challenge 1 (DC1): The need to change business rules or application configuration parameters at any time due to competitive forces (variables or facts related to prices of a product or service) or specific domain aspects related to the nature of the problem. For example, in HVAC systems, there may be situations where energy savings are more important than pleasant temperature sensations. Such rules may either change, or differ according to the geographic location, climate or type of enterprise. This requires resources and flexibility for everyone involved in the development and deployment of the solution to change the behavior of the HVAC system. Thus, when a rule changes, it should be

relatively easy to make changes through high-level languages without requiring neither knowledge of low-level aspects related to the operation of the IoT application nor hardware platforms, and without the need for manual reprogramming.

Domain challenge 2 (DC2): The need to deal with scenarios composed by numerous contextual sources, in a simpler manner. Consider, for example, a healthcare domain that deals with various and heterogeneous data sources with strong response time demands. Thus, it may be necessary to model rules for decision-making in a medical emergency system that is dependent on the (efficient) processing of various events, e.g. data from sensors coupled to the patient, availability and other data informed by the doctor, data from smart traffic systems to indicate the best routes, data from emergency services with available ambulances near the patient, data on hospital availability, including the stock of medicines indicated for the patient's condition. In such scenarios, involving many variables in complex decision-making, it is necessary to have user-friendly languages to facilitate the modeling of the solution that should include the possibility to configure timely responses according to the criticality of the applications.

Domain challenge 3 (DC3): The need to engage a multidisciplinary team to work in a simple and more intuitive manner over different abstraction levels. For example, the BPM expert should be able to make business rule changes at any time, based on their expertise, with minimal additional effort. On the other hand, the Situation/Rule expert should also be able to change the situations/rules in user-friendly language and closer to the way the problem is perceived in the real world by the Domain expert.

Domain challenge 4 (DC4): The need to deal with heterogeneous, dynamic and flexible real-world environments as simple and structured as possible. This need is related to the integration of technologies, standards, languages and devices with the existing IT solutions. For example, depending on the country or region, there may be advantages to adopt certain types of technology. In this way, the solution should offer a fully decoupled approach that allows one to choose different types of hardware platforms, without the need to change higher-level aspects of the application.

LAURA architecture seeks to tackle these four (4) challenges by promoting the flexibility required to meet the demands of adaptive real-world IoT applications/domains. Indeed, the increase of interest in applications in the aforementioned IoT scenarios will demand solutions capable of dealing with requirements that may change at any time. These solutions should provide features such as portability, scalability, and full decoupling capacity in order to support a complex and heterogeneous IT business environment which is composed of a wide variety of technologies integrated with enterprise systems. This makes support solutions such as LAURA a strategic component for the deployment of IoT applications.

3. Related work and system requirements

The papers discussed here have been selected and analyzed based on a rigorous systematic review process described in our previous work (Teixeira et al., 2017b). The same systematic process has been performed again in order to include relevant papers published subsequently. The spreadsheet containing the entire systematic review process (planning, searching and analyzing) is available in the 'Related-works-Systematic-Mapping' repository of the LAURA project at GitHub (Teixeira et al., 2017a).

3.1. Related work discussion

We have analyzed related work on the light of the Domain Challenges (DC) identified in Section 2. This analysis is summarized in Table 1, as described in the next following section. Papers will be identified with a '#' and an ID number to facilitate visualization in Table 1.

The work presented in Bai et al. (2007), identified as #01 in Table 1, presents a 5-layered architecture as follows: abstract hardware, service registry, context model, reasoning and application. The architecture also includes two cross-layer modules: one on energy management and another on supporting service-oriented, programmable context-aware applications in various scenarios, using heterogeneous devices and a synchronization method to ensure sensitivity to context changes. The dynamics of the solution follows an agent-oriented approach, i.e. an agent is created for each registered device. Agents become pro-active and capable of perceiving context event changes, which are inserted into a Rule-Based System for further inferencing. Another interesting functionality of this work is its energy efficiency capabilities, implemented by means of the Energy Optimization Module. This component implements techniques to save energy and unnecessary network traffic based on system's situations (context). With respect to the DC, this work does not address DC1, DC2 and DC3. It only partially tackles DC4 by providing support to different hardware platforms through the "abstract hardware" layer. It also provides support for data storage by means of the Working Memory component as part of the Rule-Based System.

Paper (Choi and Rhee, 2012), identified as #02 in Table 1, presents a distributed Semantic Sensor Web Platform (SSWP) aiming at simplifying information processing through three levels: (i) aggregating raw data via Smart Gateway (SG) components and multiple interfaces; (ii) context virtual sensor descriptions, which are generated by SGs in order to make smart decisions and (iii) inferred context information (also known as situations) based on events received from multiple context providers and SGs. The paper applies the solution in an irrigation system scenario in which irrigation is activated based on data collected from sensors spread over the crop. The system is also capable of making smart decisions based on events provided by external context providers such as changing truckersśroutes due to poor weather conditions and/or accident reports. Another interesting aspect of this work is the separation of semantic services from domain services by means of multiples SGs using (i) management functions such as aggregation, discovery and history management; (ii) brokering functions such as service discovery, registry and abstraction; and (iii) storage and inferencing functions with ontology and rule-based systems. With respect to the DC, this work does not address DC1 and DC2. DC3 is partially addressed by means of the proposed architecture and its components, but it does not define which stakeholders are involved and their role in the development of applications. DC4 is almost fully addressed by means of SGs.

The work discussed in Jara et al. (2014), identified as #03, presents the "Digcovery" architecture, a global resource discovery and service-oriented framework aimed at facilitating the deployment of context-aware applications. This architecture deploys mechanisms to allow users to register sensors to discover various kinds of resources based on geo-location, resource types and mobile platforms. The architecture allows these external resources to interact with a scalable and elastic search engine in order to allow high complex queries (serialized as JSON messages) to be efficiently executed (with low response time). To achieve these objectives, the following strategies are proposed: (i) the integration of different kinds of systems, technologies and standards such as IPv6, 6lowPAN, EPC and legacy devices to transform any kind of service, device or resource into a semantic interoperable format via a JSON API; (ii) the use of a Digcovery component, which is a centralized point to manage and discover resources and services. This component is based on a common ontology and profiles from the IP-enabled smart objects alliance (IPSO) compatible with DNS-SD types to interoperate with final applications through standard API

Table 1
Domain challenges and system requirements.

Domain challenges	Requirements	#01	#02	#03	#04	#05	#06	#07	#08	#09	#10	#11
DC1	R1	No	No	No	No	No	No	Yes	No	Yes	No	No
	R2	No	No	No	Yes	No	No	No	No	No	No	No
DC2	R3	No	No	No	No	No	No	No	No	No	Yes	No
	R4	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DC3	R5	No	No	No	No	No	No	No	No	No	No	No
	R6	No	Yes	No	Yes	No	No	No	No	No	Yes	Yes
DC4	R7	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	R8	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
	R9	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	R10	No	Yes, but only REST API	Yes, but only REST API	No	Yes, but only REST API	Yes, but only REST API	Yes, but only REST API	No	Yes, but only REST API	Yes, but only Pub/Sub API	Yes, but only REST API

such as DNS, CoAP and REST APIs; (iii) the use of an ElasticSearch component to collect unorganized data from different directories in order to filter services and to integrate these services with different types of resources. This aims to provide methods to form a global organized lookup service capable of efficiently managing a distributed and heterogeneous set of repositories, filtered by resource type (e.g., temperature). This paper does not address DC1, DC2 nor DC3. DC4 is partially addressed by means of the Digcorevy Core APIs and Discovery Protocols components.

The work discussed in Kuo et al. (2008), identified as #04, proposes a heterogeneous middleware architecture and an open platform to develop context-aware applications based on: (i) a SensorInfo gateway component capable of hiding hardware discrepancy and deploying web-based API to facilitate the development of context and location-based applications; (ii) Location-based services capable of providing location estimation for the developers from different positioning systems; and (iii) Remote update features in order to support the update of sensor mote images from a web interface according to each platform. This work does not address DC2 and partially addresses DC1, DC3, and DC4. DC1 has been addressed by means of the remote update features provided by the platform. DC3 has been addressed by means of the fully decoupled nature of the components and services of the architecture. Finally, DC4 has been addressed by the SensorInfo Gateway, but it fails to present information about how they offer APIs to final applications.

The work presented in Paphitou et al. (2015), identified as #05, introduces a generic context-aware architecture featuring: (i) a Sensor set component capable of connecting different sensor platforms with a variety of context sensors; (ii) management application facilities, to offer services to develop context-aware via an exposed API; (iii) RESTful services, to support Web Services operations such as addition or deactivation of specific sensors; (iv) flexibility with respect to sensor configuration to change measurement frequency according to users' requirements; and (v) information provisioning according to the sensors connecting to corresponding boards. This work does not address DC1, DC2 nor DC3. It tackles DC4 by supporting various types of platforms by means of exposed WSs. In addition, since RESTful services are supported, it facilitates the integration of external sensors.

The work presented in Cubo et al. (2014), identified as #06, proposes a Cloud-based service-oriented platform to manage and orchestrate heterogeneous devices according to their behaviors based on the following strategies: (i) a lightweight model to characterize device behaviors, organize communication messages, and to compose devices using the concept of devices as services. In order to promote interoperability among devices, message exchange is realized via gateway in a generic and standardized manner; (ii) a web-based approach by means of an external cloud-based API; (iii) combination of devices via gateway according to their complexity or specificities; (iv) a discovery service, to discover, manage and orchestrate the interaction among devices based on specific scenario requirements; (v) a behavior-aware orchestration method to deploy a cloud-based (Google Cloud Platform) approach in order to use the concept of devices as services. Service subscriptions are based on standard languages such as WSDL, SOAP, WS-Discovery and SOAP-over-UDP and are used to manage event channels, which run on top of a protocol stack from IPv4, IPv6 and IP multicast network layer until SOA applications. This paper fails to address challenges DC1, DC2, and DC3. It contemplates DC4 given it provides a strong framework to deal with device heterogeneity.

The work discussed in Corredor et al. (2011), identified as #07, presents a Knowledge-Aware and Service-Oriented (KASO) ontology-based middleware to enable the development and deployment of real-world applications via common API services in the Cloud. By means of these services, sensors and actuators can be registered with a network hierarchy to minimize the complexity of underlying mechanisms, such as resource allocation, orchestration, and service discovery. This work uses protocols, services, and other mechanisms to enable the deployment of business process applications according to the following strategies: (i) minimizing service overhead via lightweight protocols and optimized service-orientation; (ii) reducing costs with 'plug and play' discovery mechanisms, which minimize manual node service configuration work; (iii) orchestrating resource-aware services to allow dynamic resource allocation to optimize resource allocation according to current situations; (iv) simple programming in order to simplify the development of complex scenarios for business process applications. This work does not address DC2 and DC3. It approaches DC1 given it provides ways for the development of final IoT applications. However, this work's approach does not take into account the possibility of remotely changing the node code applications behaviors during system runtime, when considering the dynamic and constant changes in business rules. DC4 is addressed through the semantic-based framework, as previously discussed.

The work discussed in Avilés-López and García-Macías (2009), identified as #08, proposes TinySOA, a service-oriented architecture capable of integrating external applications and facilitating the use of sensor networks to create abstractions at the level of Web Services (WS). The approach is based on a centralized server that incorporates the gateway, database, and communication

functions, which define web services for accessing external applications and/or gathering data directly to/from the sensor network nodes. The architecture defines the following components: (i) node, to encapsulate sensing node functions, using well-known approaches to deal with node topologies such as service discovery to promote hardware identification of sensing capabilities, actuator control and abstraction of hardware low-level aspects; (ii) gateway, to aggregate sensor node capabilities; (iii) registry, to store information about the infrastructure, user events and task management information to define how node data should be read; (iv) server, which acts as a web service provider, offering interfaces for each supported network and sending commands to network without the use of task management. This work does not address DC1, DC2, and DC3. It tackles DC4, but it fails to present information about the available APIs to final applications.

The adaptive Web of Things (WoT) convergence platform proposed by Yu et al. (2016), identified as #09, presents a global mechanism to help user communication through the Web or external applications based on a common semantic or non-semantic data in order to facilitate device interoperation. IoT devices in this platform define HTTP Rest interfaces. The platform is composed of: (i) registry, to support data conversion among non-semantic data and support for semantic translation based on technologies such as RDF and SPARQL queries; (ii) retrieval and recommendation, to maintain virtual nodes and their data, including data synchronization services. This component also implements recommendation capabilities; (iii) ubiquitous process management (UPM) and engine, responsible for resource identification, services mashup with smart devices, thing-to-thing communication, and thing monitoring capabilities for collaboration. This work does not address DC2 and DC3. DC1 is partially addressed through the global mechanism to facilitate communication and DC4 is addressed by means of platform components previously presented.

The work presented in El-Mougy et al. (2019), identified as #10, presents features, concepts and aspects that address several of our DC. The approach is based on a conceptual architecture consisting of 4 layers (sensing, fog, Core and Application/services layers) that together fulfill requirements to develop personalized scalable IoT networks. The main goal is to offer optimized techniques to allocate the most appropriate application sensing resources considering dynamic requirements (such as unpredicted user needs). The following strategies are proposed: (i) virtualization of physical sensors networks (similar to VM-based approaches) in order to facilitate the dynamic allocation of resources through optimization algorithms, resource discovery, and application decomposition. Application decomposition is required when problems in the sensing layer occur (such as a physical defect or a security/privacy breach) or when resources need to be reallocated to meet changing requirements; (ii) migration of applications to the so-called “Fog layer” in order to meet specific requirements, or when there is a more rigid latency requirement. The approach is capable of using various meta-information (such as data precision and freshness) to decide the best application deployment environment (such as the Cloud or the fog layer) considering these dynamic data; (iii) using information-centric networks to tackle mobility and traffic congestion requirements through pub/sub event-based mechanisms to provide proactive caching; (iv) supporting artificial intelligence and context awareness approaches to leverage mobility prediction, cognitive networking and adaptive duty cycling. These mechanisms allow, for example, reprogramming sensors according to users’ requests to meet changing application requirements. This work does not address DC1 and partially addresses DC3. It is the only one of the related works to address DC2 by means of the Core Networking and Fog layers. DC4 is addressed by means of the optimized techniques and strategies previous presented.

The context-aware system platform for Industrial IoT (IIoT) proposed by Alexopoulos et al. (2018), identified as #11, presents an infrastructure to support decision making for mobile and static users organized as follows: (i) the context information layer based on an ontology model to dynamically support changing of applications and user contexts using standards such as UML, the Web Ontology language (OWL) and REST interfaces; (ii) the event-driven architecture (EDA) layer to support heterogeneous communication and decoupling aiming to facilitate the management of complex systems with independent features; (iii) the enterprise layer to provide business-specific services which are key to reducing assembly errors and production costs, such as services to help locate people on the factory floor to improve work task allocation and execution, and, to track available material and errors reports during the production process in order to support expert proactive actions. This work does not address DC1 and DC2 and partially addresses DC3. DC4 is addressed by means of the Event-driven approach and through the integration layer.

3.2. System requirements

The previous section allows us to refine our challenges in terms of essential requirements that should be fulfilled by our solution (the LAURA architecture). Papers #7 and #9 tackle DC1 by offering infrastructures that support the deployment of business-oriented IoT applications. The IoT solution should offer facilities for BP modeling in a way in which the BPM Expert does not have to know the lower-level aspects. Since this is a relevant feature for most of the scenarios we are interested in (Fig. 1) we derive our first requirement R1: the architecture should define functions to allow easy deployment of business-aware IoT applications.

DC1 has been (partially) addressed by papers #4, #7 and #9. Paper #4 introduces remote node update features which are interesting solutions to deal with dynamic business environments in order to avoid manual work. The use of VM-based WSN platforms, that have light-weight code-dissemination functions, make the maintenance of the solution easier, avoiding manual node code reprogramming. Based on this, we derive another system requirement R2: the architecture should include mechanisms to allow remote reprogramming to cope with ever changing business environments.

Paper #10 tackles DC2 by offering support for the deployment of fog-oriented IoT applications with facilities to (i) avoid unnecessary data being sent to final applications; and (ii) provide response time required by real-time or critical applications, such as smart traffic application to make real-time decisions about traffic routes. In addition, they must be able to deal with large quantities of data to process, infer, aggregate, compose and make temporal correlations. In this way, we define the system requirement R3: the architecture should provide mechanisms and features to deal with the fog computing paradigm.

All the papers partially tackle DC2 by offering support for the deployment of context-aware or situation-aware IoT applications, since this is a requirement for the search and selection of the related works. It is desirable that solutions offer facilities to support a user-friendly language, using the “situation” abstraction to represent situations as they occur in the real-world and also support the specification of situations decoupled from the low-level aspects. Thus, we define the system requirement R4: the architecture should provide mechanisms and features to allow the deployment of situation-aware applications.

DC3 has been (partially) addressed by papers #2, #4, #10 and #11. No paper introduces mechanisms to reduce node processing by moving some operations or processing that are commonly performed on the nodes to the upper layers. Based on this, we

define system requirement R5: the architecture should include components or mechanisms to reduce node processing.

It is desirable that IoT solutions promote engagement, adhesion and the definition of stakeholder roles compatible with those found in the market. In complex solutions that demand the participation of multiple stakeholders and multidisciplinary teams, this should be encouraged by the platform, however, each professional should have well-defined roles according to the profiles that can be commonly found in the labor market. Facilities to engagement and adhesion of stakeholders working on their specific abstraction levels have been addressed by papers #2, #4, #10 and #11. The solutions include the use of 'Separation of Concern' applied to the levels of abstraction compatible with professional profiles that are commonly found in the labor market or the use of a BPM-based approach to generate node codes in order to facilitate the work of the professionals working at the highest level layers of abstraction. Thus, we define system requirement R6: the architecture should provide facilities for the engagement of multidisciplinary stakeholders according to well-defined professional profiles.

DC4 has been (partially) addressed by all papers by means of flexibility and freedom to use different WSN hardware platforms or IoT devices. Usually, companies adopt the solution that best meets their needs. The adopting of VM-based devices or WSN facilitates the adoption of new hardware platforms due to the fact that the programming language and OS system remain the same. Incorporating a new platform is necessary to adapt the radio communication and sensor components according to the VM-based approach used. From these aspects, we define system requirement R7: the architecture should support different devices or hardware platforms. The use of decoupled and flexible gateways provides portability by offering support or facilities for using new platforms that are not yet used by the architecture. In this way, we define system architecture R8: the architecture should support portable data communication gateways to allow the use of new devices.

Paper #3 is the only one that doesn't support facilities to store sensed data, which is an important feature to allow data analysis over periods of time, as it may be necessary to analyze data behavior over time window intervals to identify situations. Thus, we define system architecture R9: the architecture should support data storage.

Papers #2, #3, #5, #6, #7, #9 and #11 provide support to the deployment of decoupled final IoT applications via REST API and paper #11 provides the same support via a Pub/Sub API. Business IT environments are usually heterogeneous, complex and prefer the use of cloud computing. The stakeholders, who are responsible for the development of final applications, must deal with many high-level aspects affected by those that work on the lower levels aspects. The use of user-friendly communication interfaces and well-established standards, with full decoupling of lower-level technical aspects, is a strategy that can be used to allow flexibility for the deployment and integration of final applications for IoT solutions, according to business needs. In this way, the stakeholders only need to know the communication methods, patterns and formats available to access the data through the APIs. From the above aspects, we define system requirement R10: the architecture should provide facilities and freedom for the deployment of a range of decoupling final real-world IoT applications.

Table 1 presents an analysis of the domain challenges (DC) and the corresponding system requirements, considering the related work. Some of the requirements marked with 'no' mean that they were not identified or there was not sufficient information available in the study to reach a conclusion. In these cases, when it was partially addressed in the paper, the cell contains a description of what was found. For example, Choi and Rhee (2012) did not refer to or specify if processing occurred. Therefore, it was considered as a 'no', as the proposal did not describe or identify if the data were previously processed or not.

Aspects related to the treatment of context-aware or situation-aware issues were not characterized in Table 1, as it was considered a prerequisite for the selection of related works that is well characterized in the systematic mapping performed and available in the spreadsheet. Therefore, all related work somehow deals with context-aware aspects.

The entire process of planning and analyzing the papers to obtain the selected papers together with the analysis and discussion of related works helped to reinforce and confirm the initial hypothesis that, in the scientific literature, there are few architectures with properties or characteristics that are similar to the LAURA conceptual architecture proposal. This process also helped to confirm the importance, effectiveness and relevance of the properties and elements proposed by LAURA conceptual architecture, indicating real possibilities for future modifications and improvements that would permit the addition of new layers, elements and concepts. For example, the inclusion of the 'Fog Layer' in the conceptual architecture was a direct result of the analysis process of the selected papers (Aazam and Huh, 2014; Al-Doghman et al., 2016; Bonomi et al., 2012; Chen, 2017; Chen et al., 2017; El-Mougy et al., 2019; Hu et al., 2017). This, in turn, demonstrated the flexibility, adaptability, decoupling, and freedom that LAURA conceptual architecture provides.

The analysis of each stage in the systematic mapping protocol, revealed that many studies present architectures that focus on a domain-specific problem (Hilal and Basir, 2015) or architectures that were designed for specific kinds of final applications (Jantunen et al., 2008; Zirpins, 2016). Some studies concentrate on the low-level (Sánchez López et al., 2012; Shu et al., 2016) or high-level aspects (Frömel and Kopetz, 2016; Happ et al., 2017) and others present similar focus at the context/rule layer to the one proposed in the LAURA architecture (Da et al., 2014; Fonseca et al., 2016; Forsström and Kanter, 2014) although do not offer the flexibility and the full control of complex situations present in the entire LAURA architecture. In summary, this systematic related work analysis has shown that few studies present an equally streamlined and integrated approach that is geared to meet full multi-layer facilities. LAURA design, on the other hand, aims to simplify the development of situation-aware and business-aware IoT applications. This is introduced in the following sections.

4. LAURA architecture

This section presents the design of LAURA. The section starts presenting the stakeholders that interact with LAURA. The proposed domain reference model and the main components of its conceptual architecture are then described. The section also elaborates on the technological issues and the choices made for the development of LAURA applied architecture.

4.1. Stakeholders

One key aspect that facilitates the deployment of an architecture is that the roles are compatible with the professional profiles that can be found easily in the labor market. By decreasing the cognitive effort needed in the deployment of a solution, less training is needed for stakeholders, making this a more attractive option for the decision-makers within a business context. When this is not taken into consideration, stakeholders may be required to take on roles that are not within their profile or demand additional knowledge. For example, Rodrigues et al. (2015) defined the Domain Expert as a professional with basic software development skills, mainly in modifying UML models. In theory, this approach is conceivable. However, considering the reality of the IT

business scenario, the training required to prepare employees to fit this role makes this approach less attractive in a business context. Another example, Tranquillini et al. (2012) works under the assumption that the Domain Expert will be a professional with basic knowledge in BPM and IoT. This professional profile, however, is not commonly found in the labor market.

The above examples show how a design strategy can influence or complicate the use of a software architecture with similar characteristics to LAURA architecture. Its acceptability and feasibility to be adopted as a solution on large scale domains will depend greatly on the ease with which professionals can be found in the real-world market to fill the defined roles. The following paragraphs describe the roles and profiles of stakeholders that were defined for the proposed architecture.

The Domain Expert normally has experience in the field that the IoT solution will be applied; and represents the group of people who will benefit most from the results that the IoT solution will provide. For example, domain experts could be doctors, engineers, nurses, pharmacists, biologists, oceanographers, etc. Thus, this professional is one of the main users or interested in the IoT solution and, at the same time, is a member of the development team in order to assist or contribute with technical aspects of the domain problem to help the other stakeholders. It is expected that these professionals do not have any knowledge about the IoT platform or systems.

The IoT Expert is the professional who has knowledge of the different components in a solution, such as hardware platforms and the software that will be used. They are expected to have low-level programming knowledge such as the ability to manually program node codes. Furthermore, when using an automatic node code generation approach, these professionals may have some background knowledge and they only need to make small adjustments to the node code sensor.

The BPM Expert is a specialist in BPM who has practical experience with BPMN modeling and business process automation with the use of BPM systems. He has a superficial knowledge of IoT, similar to what has been described above for the Domain Expert, and also hold basic programming skills.

The System Expert is a specialist in information systems. He is the Information System Development Manager who, in many cases, performs the role of Software Engineer. This professional is normally the project team leader, providing supervision and support for the other stakeholders. The System Expert is required to have knowledge of IoT that is equivalent to the description of the Domain Expert. Depending on the size of the company, this professional may also perform the role of the BPM expert.

The Situation/Rule Expert is the specialist in rule or situation design responsible for the creation of the rules and situations in the case of situation-aware applications. Finally, the System Operator usually works closely with the support of Domain Expert to get support for the use, management or operation of an IoT solution. This role is typically held by a technician or the person responsible for supervising and running the solution. He performs the required tasks to meet the needs and objectives of the solution, such as monitoring and reporting on the results obtained.

The LAURA architecture was designed to support the professional profiles described above. In this way, the architecture aims to promote the involvement of multidisciplinary teams in order to simplify the development of applications based on IoT technologies.

4.2. LAURA domain reference model

Fig 2. presents the LAURA domain reference model. The proposed model was inspired by previous initiatives such as IoT-A (Bauer et al., 2013), Butler and FI-WARE (Butler Consortium, 2013), IoT-lite (Bermudez-Edo et al., 2017), and SSN (Compton et al., 2012). This model offers stakeholders a lean and useful artifact that promotes a common ground of understanding of the abstract concepts, properties, attributes, relationships and semantics, necessary to develop their own version of LAURA applied final IoT solution.

The proposed domain reference model is divided into two parts or visions that complement each other. The blue highlighted elements represent the concepts addressed in the '*Physical*' and '*Core*' layers of LAURA architecture and the green highlighted elements represent the concepts addressed in the '*Fog*', '*Situation/Rule*' and '*Application/Business Process*' layers.

The blue highlighted elements represent the virtual view of things, devices, entities and other related virtual artifacts. The focus is to represent the general concepts used in the '*Core*' layer. The lower level concerns related to hardware aspects, such as the type of device, sensor, tag or actuator, are not addressed by the domain model.

The '*Sensing Device*' class defines the virtual representation of a physical sensing device. An instance of this class has a single identification '*key*' and a '*descriptor*' that helps the stakeholder to characterize its purpose or utility, e.g., '*Living-room controller*', '*Claire's mobile phone*', '*Store Beacon*'. A sensing device can add one or more observable properties (of type '*ObservableProperty*') that can be any property, quality or attribute that a sensing device is able to observe or collect, for example, temperature, humidity or location. An observable property represents a topic that can be measured over time with shared conditions such as source/sensor origin, measurement unit and accuracy. Examples are the temperature of a specific place or area, the location of a certain individual, etc. Observable properties and their stream of values can be available or "exposed" by means of '*endpoints*' via an API.

Class 'Observation' represents a value or physical world magnitude identified for a certain property at a specific time. An observation of geolocation, for example, has, at least, two pairs of key values (of type '*keyvalue*'), latitude and longitude, respectively. Frequently, the magnitudes captured by observations are transmitted periodically in temporally ordered series (streams of data) with their respective observable properties.

It is possible to associate characteristics or metadata, according to the necessity or preference of the stakeholder, in any instance of the blue highlighted entities. The 'metadata' have a 'name', 'value' and 'kind'. Following this, some examples of 'metadata' values are: 'name'="devicemodel", 'value'="MTM-CM5000-MSP", 'kind'="String" (used to identify a device model); 'name'="unit", 'value'="Celsius", 'kind'="String" (used to identify the magnitude unit of an 'ObservedProperty') and 'name'="precision", 'value'="0.3", 'kind'="Double" (used to define the precision of an observation).

The green highlighted elements are related to the concepts addressed in the 'Fog', 'Situation/Rule' and 'Application/Business Process' layers.

An instance of '*Entity*' is a virtual representation of a real-world object or individual, may be composed by attributes (of type 'Attribute') and may be associated with contexts (of type *'Context'*). Instances of '*Context*' represent the universe of discourse in which the stakeholder is interested in. An entity must have a '*key*' for unique identification, should have a descriptor for human-friendly characterization, and a kind to designate its type, class or scheme. The tuple (50303, "Sergio", "Person") is an example of entity with its values referring to attributes '*key*', 'descriptor' and 'kind', respectively. An entity may present a set of attributes (instances of 'Attribute'), which are values that belong to an individual or thing that remain unchangeable during the existence of its owner in the application or scenario. A valid example of an attribute associated with entity "Sergio" would be (label = "birthday", value = "1987-03-16", type = "datetime").

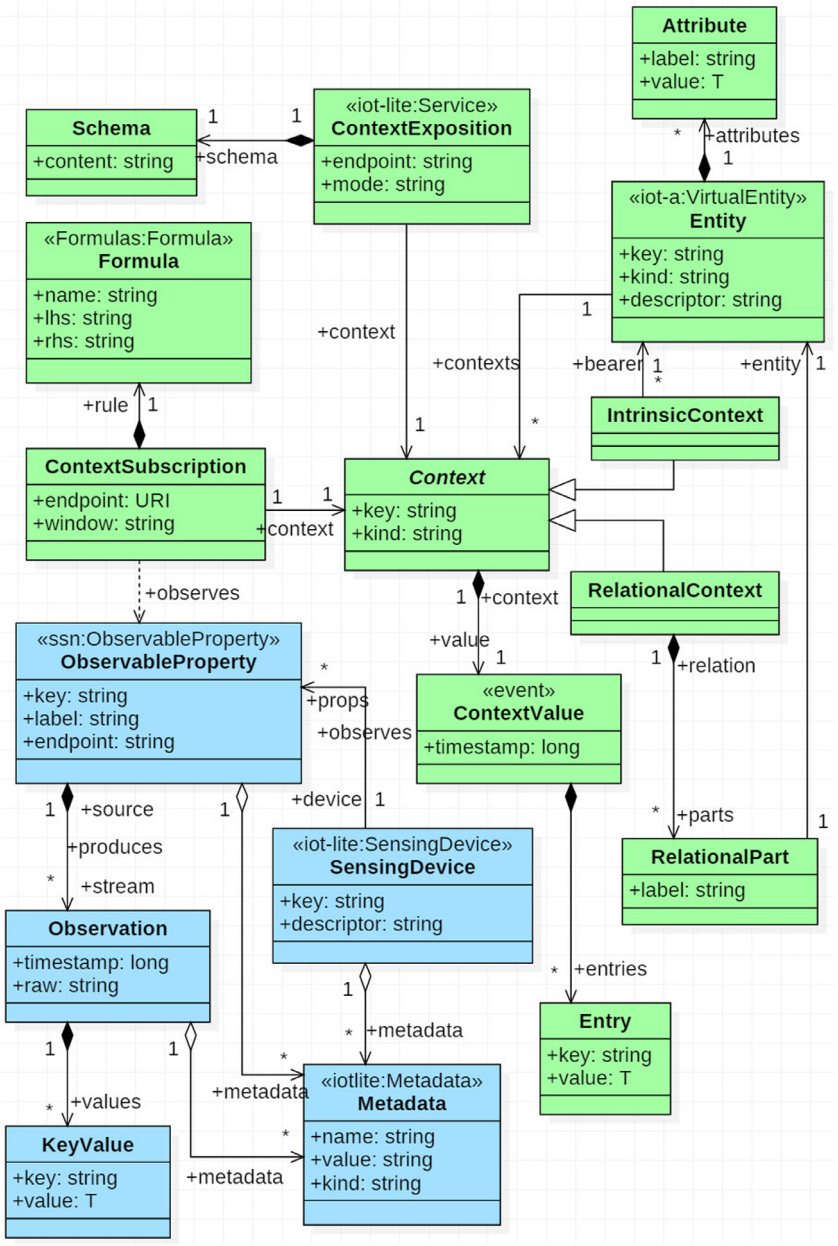

Fig. 2. LAURA domain reference model.

A context (of type 'Context') makes up the state-of-affairs of an entity, i.e., represent characteristics of an individual that are submitted to changes in the environment that it is inserted. The green highlighted elements are based on Dockhorn Costa (2007) and use a similar classification when distinguishing 'Context' in 'Intrinsic' and 'Relational'. 'IntrinsicContext' represents properties that relate only to your bearer, e.g., "temperature", "location", "mood" of an entity. 'RelationalContext' constitutes states shared between two or more entities. Good examples are "friendship", "proximity", "containment", "ownership", and so on. In both classifications, an instance of 'Context' must have a key and kind, e.g., (key = "temp50303", kind = "Temperature"), (key = "prox50303-45231", kind = "Proximity"), and in such 'Relational Context' it must be given a specific role or part within the relation to entities, like in an "ownership" context, one is expected to be labeled as the owner and the other one labeled as the owned. A 'RelationalPart' class helps in qualifying those participations.

Besides its intrinsic or relational characteristics, every instance of 'Context' has a relation with a particular '*ContextValue*' which represents its current state, quite similar to an '*Observation*' in the '*Core*' layer, in the sense that it's a temporal construct, a point-in-time event, and may present a composite of key-value pairs, a set of '*Entries*', as it can be seen in Fig. 2.

The link between the *'Core'* and *'Application/Business'* layers is made by exposing contexts and observing properties of the 'Core' layer. This can be seen in the connection between *'ContextSubscription'* and '*ObservableProperty*'. The proposed model allows an association of a certain 'Context' to the consumption of streams of 'Observation', produced directly by *'ObservableProperty'*. Once the 'Context' is defined, it is possible to associate the *'ContextSubscription'*, whose purpose is to maintain a subscription through an endpoint that refers to the URL or URI in which a particular *'ObservableProperty'* was exposed. The technical characteristics related to how this link, subscription or observation should be performed are not part

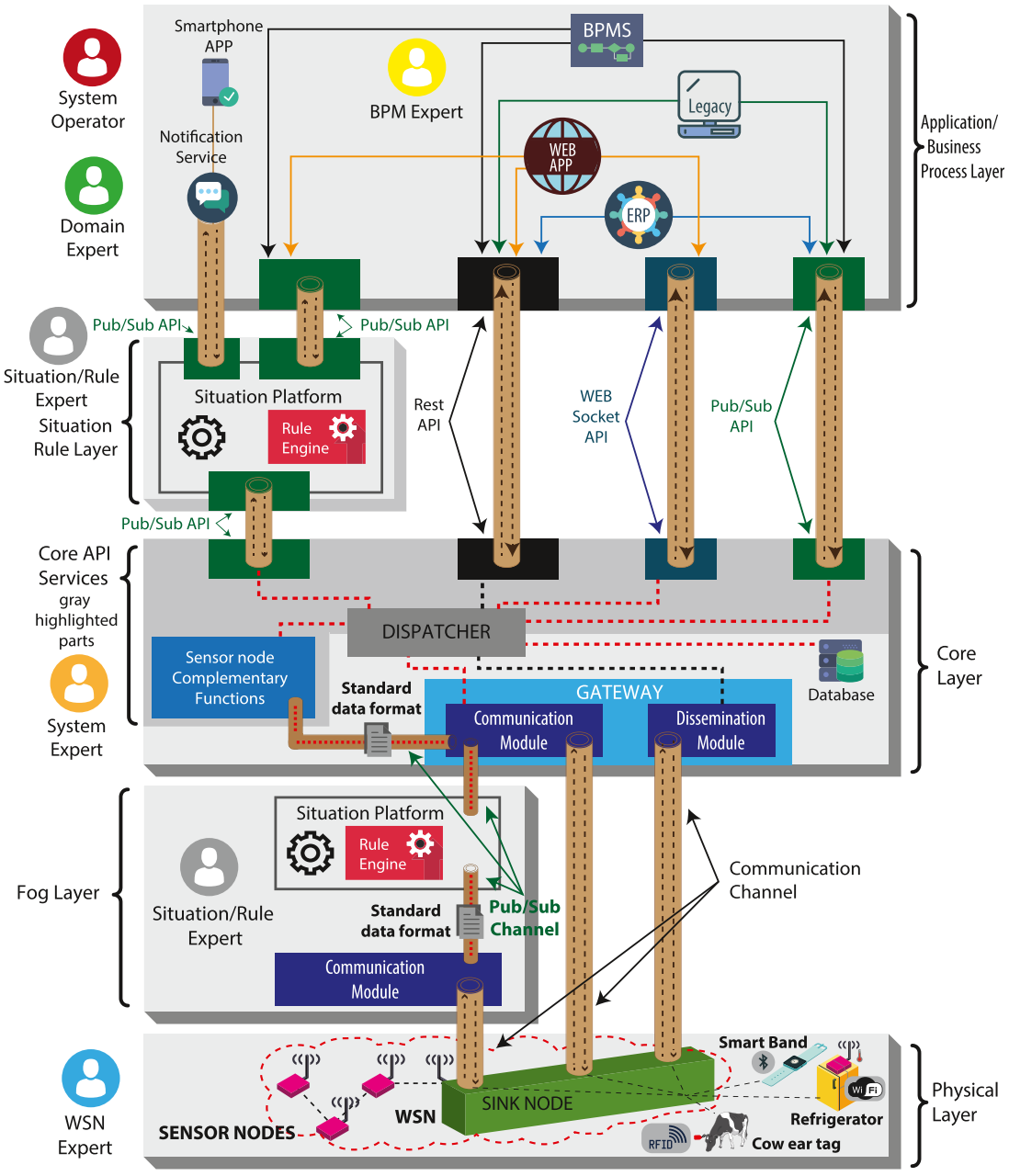

Fig. 3. LAURA Conceptual architecture overview.

of the scope of this reference model. Furthermore, in the '*Context-Subscription*', the stakeholder must be able to define a window rule in order to establish a time window to be used by the consumed part in order to interpret and send the *'Observation'* group of interest. For example, it is possible that the consuming part needs to perform some pre-processing, such as an aggregation operation over a value of the last N *'Observation'* (length window) or even *'Observation'* occurred in the last 10 min (time window). The applications of these operations are defined by means of a 'Formula'.

4.3. LAURA conceptual architecture

Fig. 3 shows an overview of the LAURA layers, and the stakeholders who are actively involved in each layer. For didactic purposes, the following sections will present each layer using a bottom-up view, describing their main components as well as highlighting the requirements met in each layer and different activities performed for each of the stakeholders in correspondence of each layer.

4.3.1. 'Physical layer'

The 'Physical Layer' includes all low-level aspects related to the running of the WSN or IoT devices, such as specific details concerning the hardware, OS system and communication protocols adopted. LAURA assumes that there is a preconfigured network of IoT devices. In this layer, the IoT Expert defines the physical devices, platform, OS systems and the most appropriate topology of sensing, taking into account the objectives of the final application. In addition to the operating environment, the IoT devices keep the code responsible for receiving data collected from the monitored real-world space or physical object, and for sending them to the Sink node.

Considering the dynamic nature of an IoT application and the corporate environment, a relevant aspect to be taken into account

when selecting the hardware platforms and systems to be used in a solution is the ability to make dynamic updates (or automatic reprogramming) of the code that runs on network nodes. Specially in the corporative environment, this feature of automatic reprogramming is highly relevant as the application requirements may change at any moment due to a change in the BM or competitive forces.

A dynamic code update is understood as how a specific solution is able to automatically spread sensor codes through the use of radio links without manual serial programming in a running network (Dong et al., 2014). A network becomes less scalable when it is necessary to recover a device, manually update the node code and reinstall it in the original location, particularly when this is in an inhospitable location.

A good reprogramming mechanism must be capable of updating the nodes in an efficient manner, with low interference in the running applications and avoiding, as much as possible, the increase of the processing and energy consumption of the nodes. There are several dissemination mechanisms. The Full-Image Replacement, for example, executes a complete replacement of the node code image. This approach is implemented by Deluge (Hui and Culler, 2004), the native TinyOS dissemination protocol. TinyOS is the first and currently the most commonly used embedded Operating System (OS) for low-power WSN devices (Levis, 2012). One concern with this technique is the high use of bandwidth and power consumption during the image update.

In Virtual Machine (VM) approaches (Balani et al., 2006; Branco et al., 2015), a high-level language is executed instead of a native platform code. After the compilation process is completed, the high-level code is transformed into bytecode and broken up into small parts that can be transmitted through the network with low bandwidth use.

LAURA follows a VM-approach, which means that the 'Physical Layer' is composed of one or more virtual machines based on a VM infrastructure, called Terra System (Branco et al., 2015), built on top of TinyOS (see section 4.4). This infrastructure combines a reactive programming language and a component-based virtual machine, which allows for different VM customizations using the same high-level language and compiler.

From a design point of view, the LAURA Physical Layer consists of a sink element (Sink node) and a set of connection elements (Interfaces). The Sink Node is responsible for exchanging data packets and node image updates between the IoT devices and the upper layers via communication channels. The data of the phenomena of interest are sensed by sensor nodes and IoT devices and they are periodically sent to the Sink node, which is responsible for packaging and sending the messages from the sensor nodes to the upper layers. Each message travels from the Sink node to the Core Layer Gateway or from the Sink node to the Fog Layer using a dedicated communication channel and a specific communication mechanism for sending and receiving messages. In addition, the Sink Node coordinates the reprogramming process with the IoT nodes and interoperates with the Core Layer '*Dissemination Module*' via a communication channel to receive and forward the node images to the WSN nodes.

In this layer, LAURA offers three communication interfaces. One communication channel is responsible for sending and receiving data packets between the IoT device and the 'Communication Module' of the '*Core Layer*'. This channel is used when it is not necessary or possible to use the features provided in the 'Fog Layer'.

For example, when the network nodes code process collected data locally, minimizing the number of packets sent.

Another example would be in situations where there are technical limitations that prevent the deployment of a device to act as a '*Fog Layer*', forcing the packets to go directly to the '*Core Layer*'.

The second communication channel is used exclusively for node code reprogramming. This channel exchanges packets containing only the updated node code data between the IoT devices and the '*Dissemination Module*' located at '*Core Layer*'.

The third communication channel is used for communication with 'Fog Layer' when it is intended to use the features of reducing, filtering or aggregating data. This aims to achieve requirement R3.

It is worth to mention that the VM approach and the communication solution provided by LAURA at Physical Layer provide the IoT Expert, regardless of the domain and type of application shown in Fig. 1, with a uniform way of programming and dealing with heterogeneous hardware platforms and devices. This seeks to meet requirement R7.

4.3.2. 'Fog layer'

The systematic review carried out in this paper and the particular analysis of application requirements and characteristics revealed that the traditional model of Cloud Computing is unable to give adequate support to some particular IoT applications when: (i) it is necessary to deal with applications that produce or consume large quantities of data from the edge of the network. According to a Cisco prediction, data produced by people, things or machines will reach approximately 500 zettabytes by 2019 and 45% of this data will be analyzed, processed and stored on edge devices (Hu et al., 2017). By introducing techniques and mechanisms to filter, aggregate or prevent unnecessary, invalid or redundant data from being transferred to the Cloud or uppers layers it is possible to considerably reduce the volume of sent data; (ii) the application requires a short response time of between milliseconds to seconds. The round-trip delay from the devices to the Cloud and vice versa can take minutes and this period of time may be impracticable in some cases (Cisco, 2015; Yi et al., 2015). For example, in a Healthcare IoT system (Rahmani et al., 2018) the response time of vital signs is a very critical aspect that requires rapid processing and action that can normally be done by a device located at the edge or closer to the sensors; (iii) the application requires mobility support in a widely geo-distribution environment or location awareness (Bonomi et al., 2012). By using mobility mechanisms, it is possible to decouple the device identification from the physical location via distributed directory systems (Chen, 2017). For example, a smart traffic light system that controls some sensors in a specific area in order to detect the presence of bikers and people, calculates the distance of an approaching vehicle and takes actions when necessary. In this case, the devices must interact quickly with the smart lights, sending warnings to prevent accidents; (iv) it is necessary to deal with a heterogeneous environment (Hu et al., 2017) by supporting different types of data that is sent from a wide variety of devices with different types of hardware, languages, operating systems and communications systems. In this way, it is necessary to dynamically deal with different requirements, especially low latency, which can change at any time according to the conditions of a specific moment of the communication links. For example, an application that aims to prevent manufacturing line shutdowns needs a response in milliseconds from the devices to restore the power. Low latency makes the difference between an incident that can be quickly controlled without causing major disruption to a disaster due to a cascading system failure (Cisco, 2015).

In response to the challenges presented above, Cisco first introduced the concept of Fog Computing (Aazam and Huh, 2014; Cisco, 2015; Marín-Tordera et al., 2017). Fog can be considered an extension of the Cloud, with widely spread devices (also called 'fog nodes') capable of storage (usually temporary), processing and communication with the final application in order to support new services and applications with specific QoS demands.

LAURA architecture proposes a '*Fog Layer*' to reduce, filter or aggregate, in the space or time dimensions, the sent data from 'Physical Layer', preventing, as much as possible, unnecessary or invalid data packets from being transmitted to the upper layers. For example, the imprecision of the measures can be offset by some kind of statistical processing in the Fog Layer. Therefore, in order to associate the sensed data, Quality of Context (QoC) parameters (e.g. Coverage, Up-to-dateness, Frequency, Accuracy, and Significance) (Buchholz et al., 2003; Gray and Salber, 2001; Hoffman, 2016; Toninelli et al., 2009) can be checked, allowing the final applications verify the usefulness or the temporal validity of this data, according to its purposes. In this way, it is possible to add QoC features in this layer to prevent unnecessary or invalid data from being sent to the upper layers.

The idea is to include a Fog node with a power supply, high processing, and memory capacity, between the 'Physical Layer' and 'Core Layer'. The Fog node is usually located at the edge of the network but may be placed anywhere in the network between applications and the network or sensor nodes. It can be co-located alongside smart gateways.

LAURA 'Fog layer' is responsible for: (i) managing the communication channel between the 'Sink node' and 'Physical Layer' to receive sensed data; (ii) the standardization of data packets before forward to the 'Fog node' module in order to facilitate the data processing and promote the 'separation of concerns'; (iii) managing of communication with the 'Fog node' within the layer itself through a Pub/Sub channel in order to promote the decoupling.

The 'Fog node' module consists of a 'Situation platform' with a rule-based engine that supports a CEP computing paradigm. The 'Situation platform' must be capable of encapsulating the features provided by the CEP engine and offers a user-friendly language and mechanisms that support modeling and processing of situation specification, detection, and lifecycle control.

The situation types and rules will be created by the Situation/Rule expert with the support of the other stakeholders, mainly the Domain expert. The objective is to establish particular situations of interest or complex contexts in which the data can be filtered or aggregated, without causing any type of difficulty or prejudice for the final application.

The 'Fog Layer' seeks to meet requirement R3. The layer is currently under development; thus, it is not fully available in this version of LAURA deployment architecture. The architecture design takes into account that the fog layer may not be used depending on the specificity of the design whose requirements may not require this layer.

4.3.3. 'Core layer'

The '*Core Layer*' plays a key role in abstracting and decoupling the low-level aspects that are commonly found in the '*Physical Layer*', providing communication interfaces for the application layer. The Core Layer minimizes the processing and the complexity of the '*Physical Layer*' or '*Fog Layer*', offering to the applications homogeneous access to low-level services executed by IoT devices. Additionally, the Core Layer is responsible for the dissemination of a new node code image received from the Application Layer. Thus, this layer has the purpose of providing the necessary support to meet the major requirement R10. The System Expert works at this layer registering the entities and contexts derived from the specific domain of the problem, configuring core API services and eventually defining situation rules that will trigger specific actions in the IoT application.

The main components of this layer are the Gateway, the Dispatcher, the Complementary Functions container, and the Database. The layer also holds the *'Core API services'*, which are communication facilities running at the Core Layer on behalf of the applications.

The Gateway is a central element for the operation of this layer. It is divided into two modules: Communication and Dissemination. The *'Communication Module'* is a software artifact that must be able to process and delivery packets quickly and that can be configured or adapted to maintain interoperability with different types of communication channels according to the technological choices made by lower layers. This module is responsible for sending and receiving data packets to and from Physical or Fog Layers, performing message conversion to a standard lightweight data-interchange format that is independent of language and easy for human manipulation or for machines to parse and generate it.

This feature seeks to meet R8 due to the capacity to offer portability and flexibility, by abstracting and transforming low-level IoT data packets to a common layout. Furthermore, it also enables other modules and layers to process information easily, regardless of the type or the data structure used in the Physical Layer.

Another gateway module is the '*Dissemination Module*'. It hosts a code dissemination service, which is needed for any final application. The dissemination starts when a new image is received from the API and through the '*Dispatcher*'. After the new node code image has been received, this module establishes a communication channel with any kind of device available in the '*Physical Layer*' (Sink Node or any IoT device), according to specific features of the IoT platform used. As aforementioned, this feature seeks to meet requirement R2 and avoids the need for manual node code reprogramming.

The '*Dispatcher*' is a key part of the Core API services that sends and receives data packets or updated node code data packets in different data flows, according to the type of API. This element seeks to meet R9 due to the capacity to interacts with the Database performing data persistence to maintain a record of the sensed data, according to the parameters or rules defined in the architecture. The persistence of data in a Database is justified by the fact that some applications need to access and maintain the data for a certain period. It is common that final users are interested in the statistics or history of the data for a certain period, according to the characteristics and objectives of the applications.

The Core API services are accomplished through widely adopted open standards. There are three types of APIs in this version of LAURA, as shown in Fig. 3: (i) the Pub/Sub API is used when the response time is a concern or there is a lot of traffic; (ii) the WebSocket API is used when the final application (commonly Web applications) requires this type of communication; and (iii) a REST API is used when the response time is not a concern.

The Pub/Sub communication interfaces allow final IoT applications to receive sensed data in real-time from the '*Physical layer*' devices. Applications interested in a specific data flow subscribe to any available channel provided by Core Layer to receive the data. In this case, the '*Core Layer*' acts as Publisher in the communication. The Pub/Sub channels are also permanently connected to the 'Situation / Rule Layer' in order to send the sensed data to the 'Situation Platform'.

The WebSocket API defines a second form of data access and exists to meet the demands of Web applications that are interested in the sensed data. The option to offer this specific API is due to the growing demand for real-time Web applications and because it is a W3C standard.

The Rest API supports final applications or any kind of communication that does not require real-time data. Thus, in order to allow more specific queries to previous data records, the Core Layer offers the RESTful API as a third alternative to Physical Layer data access. Through this API, the applications can perform on-demand queries, searching, for example, data of a specific sensor of the network or temperatures that reached a certain threshold It also enables queries to the Database in order to access data previously saved by the Dispatcher.

There are three data flows that are handled by the '*Dispatcher*'. The first one refers to the sensed data that come from the '*Physical layer*' through the '*Communication module*'.

The second flow refers to data coming from superior layers to the '*Physical layer*' or '*Fog layer*'. This data flow is usually related to actions to be performed on a particular sensing device or actuator. Once the data are received, the *'Communication Module'* performs the necessary data conversions before forwarding it to the physical layer. This is the inverse process of what is done when the data are received from the physical layer. In general, the operation of this component is simple: after identifying the need for measurement conversion of the received data, the respective conversion function is performed to transform it in a user-friendly format that is normally used by the final applications. The third data flow refers to new node code images coming from the upper layers and that is directed to the *'Physical layer'*. This flow is a specific kind of data, related to reprogramming functions, and it is handled by the *'Dissemination Module'*.

The '*Sensor Node Complementary Function*' is a Core Layer component that encompasses several auxiliary functions. This component seeks to meet R5. By moving certain types of the processing out of the sensor node, it becomes possible to reduce complexity, processing and energy consumption into the network.

4.3.4. 'Situation/Rule layer'

The requirements of a business system change with a certain frequency. The changes may occur due to competitive forces or with changes to the BM (Bali, 2009). Furthermore, business systems are becoming increasingly complex and there has been a growth in process automation in various types of BP, mainly from the influence of the IoT on the company BM. Considering this, there is a demand for a model or development paradigm that is more suitable for a context in which there are constant changes to the business requirements, directly affecting the rules of the BP. In this way, the rule-oriented development paradigm, or declarative programming, offers a series of advantages providing fast answers to the constant changes that may occur at any moment.

It is possible to highlight the following advantages in using Rule-oriented systems: (i) rules are easier for a Domain expert or any employee from the administration area to understand; (ii) changing or adding a new rule is easier when compared with imperative programming style based on if-else aligned instructions.

Despite the possibilities and advantages that Business Rule Management Systems (BRMS) can offer, e.g. JBoss/Drools (RedHat, 2017), there are groups of IoT real-world applications with specific features that traditional BRMS are unable to provide (Aliverti et al., 2016). This group of applications demands, for example, the analysis of facts over the time, the ability to deal with and make inferences of large numbers of events from multiple sources, infer relationships, and others in order to allow complex events to be detected (De Maio et al., 2014; Fülöp et al., 2012). These types of applications require a Complex Event Processing (CEP) computing paradigm (Gartner, 2017). For example, a fraud detection system needs to check, in real-time, a large number of events from different locations, including the transaction timestamp and duration and expiration date, in order to detect complex situations that cannot be identified in few events (Aliverti et al., 2016).

Ideally, in scenarios like these, the system should be capable of providing the Domain Expert with the means to specify how the system should respond, abstracting the low-level technical aspects, with a user-friendly language that is closer to the way in which the problem is expressed in the real world. These specific methods and ways to specify complex real-world situations lead to the concept of Situation Awareness (SA) (Endsley, 1995). SA refers to the perception of environmental contexts and events, taking into consideration its relationship in space and time, understanding its meaning and the projection of its general state in a 'Domain of Discourse'. In other words, a situation refers to how a stakeholder defines a state of affairs in reality (Adi et al., 2000; Ye et al., 2012).

Although conventional CEP-based systems, e.g., ESPER (EsperTech Inc., 2017), Drools, StreamInsightTM (Microsoft, 2017), are capable of aggregating, composing, reasoning over and deriving composite events, few are able to represent and manage situations from a specific domain in an expressive and user-friendly manner. Conventional CEP-based systems fall short in its ability to support the situation-awareness concept by itself since events in these systems are only detected when they occur, i.e., when they finish at the same time or before the present moment. A situation, on the other hand, can be detected in the first instant that it appears, without knowing when or if it will end. An event does not support the concepts of "happening now", however, it is an intrinsic part of the situation concept.

Real-world IoT domains like those depicted in Fig. 1 are composed of several context sources. In these domains, the potential situations that can result from the combination of environmental context and events are enormous, i.e., the states of affairs in reality that may be of interest of the stakeholders and applications are considerably large. Therefore, it is highly recommended that the IoT platform provide situation management support for dealing with those real-world IoT applications. In fact, one of the distinguishing features of LAURA is the situation/rule layer, which is managed by a rule-based platform based on Drools.

A situation refers to how a stakeholder defines a state of affairs in reality (Adi et al., 2000; Ye et al., 2012). Situation, as composite events, inherits event dimensions such as: composition, aggregation and temporal correlation with other situations or events. In addition, situations present an activeness dimension. A situation is said to be active while those properties satisfy predicates captured in the situation type logical expression. It ceases to exist when those properties no longer satisfy the defined predicates. In this case, the situation is said to be a past situation. The point, in time, in which a particular situation instance is detected, is called the 'situation activation instant' and the point, in time, in which the situation ceases existing, is called 'situation deactivation instant'.

SCENE (Costa et al., 2016) is the rule-based platform adopted by LAURA. It connects CEP-based and situation-aware approaches by leveraging the Drools general-purpose rule system and Drools Fusion CEP module as a mechanism for situation detection and lifecycle management. The Fusion module offers an event schema, stream processing, temporal correlation, and time-window operators, seamlessly integrated with its Drools Rule Language (DRL).

The main objective of the 'Situation/Rule Layer' is to allow an easy connection between situation-aware final applications and any kind of chosen situation platform or engine. In this way, the application is able to receive triggered situations according to the programming done previously by the Situation/Rule Expert. The central component of this layer is the Situation Platform which objective is to process situations and create a service-oriented access in an easy and practical way so that the applications only need to program the actions to be performed in response to the triggered situations. For example, a BPM-based application can initiate a process according to a triggered situation and program a flow of BPMN actions.

The 'Situation platform' is composed of a rule-based engine that supports CEP computing paradigm and a platform, system or tool that is capable of encapsulating the features provided by CEP engine, offering a user-friendly language and mechanisms that support modeling and processing of situation specification, detection, and lifecycle control. In addition to meet requirement R4, this platform also seeks to meet requirement R1 since the BPM Expert can model a BP from a triggered situation, minimizing the need

to know the lower-level aspects, as situations are modeled by the Situation/Rule or System Expert.

4.3.5. 'Application/Business process layer'

As we argued in the Introduction, enterprise infrastructure should promote the participation of high-level users in the development of IoT applications. BPM experts, for example, should continue using their preferred graphical modeling notation, leaving low-level details to a model compiler and a run-time system. Business Process Model and Notation (BPMN) (OMG, 2015), for example, graphic language, is the de-facto standard of modeling business processes, being suitable for non-expert IT professionals to model BPs without needing low-level knowledge or extensive training.

LAURA's Application/Business is the place where BPM Experts, Domain Experts, and other high-level users interact with LAURA. In this sense, this layer provides functionalities related to the development of final applications. It provides a place in which domain applications and business processes can be visually specified, i.e., the rules of business application and the situations of interest are modeled through a graphical interface. This layer is also responsible for starting the process of transforming the specification diagrams created by the developer into node/device processable instructions, so the machine can execute the model.

The '*Application/Business Layer*' also sets a place where the users can get information and monitor the state of the IoT devices. For example, they can identify available sensor nodes with their respective endpoints, including the APIs and methods available for use by final applications. They can also upload and automatically disseminate sensor node codes or associate the endpoints with the access keys of the stakeholders according to established profiles and permissions. The layer contains components that offer dashboard reports and graphs to aid the work of the IoT experts who is usually responsible for maintaining and monitoring the solution.

This layer provides a way for the integration of final applications with one or more IoT devices through standardized communication interfaces, freeing developers from any concerns about systems or technologies related to the lower layers or the operation of IoT devices. In this way, the developer only needs to be informed about the available APIs so that the final application can access the data collected by the sensors. This allows total decoupling from the low-level technical aspects. The stakeholder, responsible for the creation or integration of the final applications can gain easier access to the sensed data or triggered situations through a Web system. The only requirement is that this person knows the endpoint or the URI of the resource, methods and access keys needed to receive sensed data from the IoT devices.

Ultimately, the 'Application/Business Process Layer' provides a place where the stakeholders can create or modify their applications at any time according to new demands or needs of the company business.

4.4. LAURA applied architecture

This section presents LAURA applied architecture based on the conceptual architecture, and according to the technological choices that were discussed and presented in Section 4.3. The choices made are fundamental in simplifying the work of the developer of the final application. Some deployment considerations will also be presented, regarding the technologies, languages, standards, systems, libraries, and tools used to develop LAURA applied architecture, which is depicted in Fig. 4. The main differences between the applied and conceptual architecture are highlighted in this figure.

Concerning the '*Physical Layer*', we have mentioned that LAURA uses a VM approach based on Terra System. There are three basic elements in a network based on Terra (Fig. 5): (i) a Céu-T script language – that relies on the reactive Céu language proposed by SantAnna et al. (2013) – which is used for node (bytecode) programming; (ii) a set of customized components that are defined according to the node platform and application type; and (iii) the Terra virtual machine (VM-T), which controls the bytecode execution and handles the output and input events.

Among many interesting features found in Terra System, the following three were central in our decision to choose Terra as the VM-oriented implementation at the Physical Layer. The first feature refers to the availability of high-level programming resources and functions. This speed up and makes easier the programming of low-level tasks. The second important characteristic is its native support for dynamic code update by means of bytecode dissemination, which is an attractive solution regarding bandwidth consumption. The choice of a VM-based approach and Terra System in particular aims to achieve requirement R2.

The third feature is related to the possibility of using the same node code for different platforms. As the bytecode runs on the VM-T of the node, the code is the same regardless of the hardware platform, making it a portable solution that can be adapted for use with new types of hardware devices. Thus, it is possible to use or adapt the VM nodes for different types of hardware platforms, making such a solution more interesting to use in heterogeneous environments. This seeks to meet requirement R7.

It is also noticed that current Terra System implementation uses Active Message (AM) mechanism to exchange messages between IoT devices or WSN nodes (Eicken et al., 1992). Active Message is a message-driven asynchronous network communication mechanism that reduces processing and communication overheads between the devices in network communication (Levis et al., 2005). Besides, the communication channels between the Sink node and the 'Core Layer' make use of the TOSSAM (Silvestre, 2017), a library that supports different types of communications based on the AM protocol.

The realization of the 'Core Layer' encompasses the deployment of the following modules: (i) 'Terra Gateway', which acts as the 'Communication Module'; (ii) 'Terra Core', which provides the 'Core API services'; and (iii) 'Terra Dia', which realizes the 'Dissemination Module'. MySQL (MySQL, 2017) was the technological choice for the database. These modules, the other elements, and the communication channels are illustrated in Fig 4. To maintain compatibility and standardization, some of the technological choices, made in the lower layers, were also applied in 'Core layer'. For example, TOSSAM channels are used as a way to connect this layer with the 'Physical layer' or 'Fog layer'.

Terra Gateway (Martinelli, 2017) accomplishes the tasks designed to the 'Communication Module'. It is responsible for communicating the sensor network with the Core layer through a TOSSAM communication channel. In addition, it maintains a Pub/Sub communication channel with 'Sensor node Complementary Functions', allowing real-time interoperability with other components of 'Terra Core' module. In this way, it receives and forwards data originating from the Terra System network to the Terra Core component, enabling the full decoupling of lower-level aspects from communication interfaces that will be made available for the final applications. Furthermore, Terra Gateway is also responsible for the conversion of AM data packet to a JSON standard structured string in order to facilitate sensor data manipulation and processing, making the solution more portable and flexible. JSON – JavaScript Object Notation – is a lightweight data-interchange format based on JavaScript Programming Language (International, 2013).

'TerraDia' (Martinelli, 2017; Ribeiro, 2016) implements the Dissemination Module proposed at Core layer. It is responsible for transmitting new node code image packets to the Sink node, via a TOSSAM channel, providing remote reprogramming features to

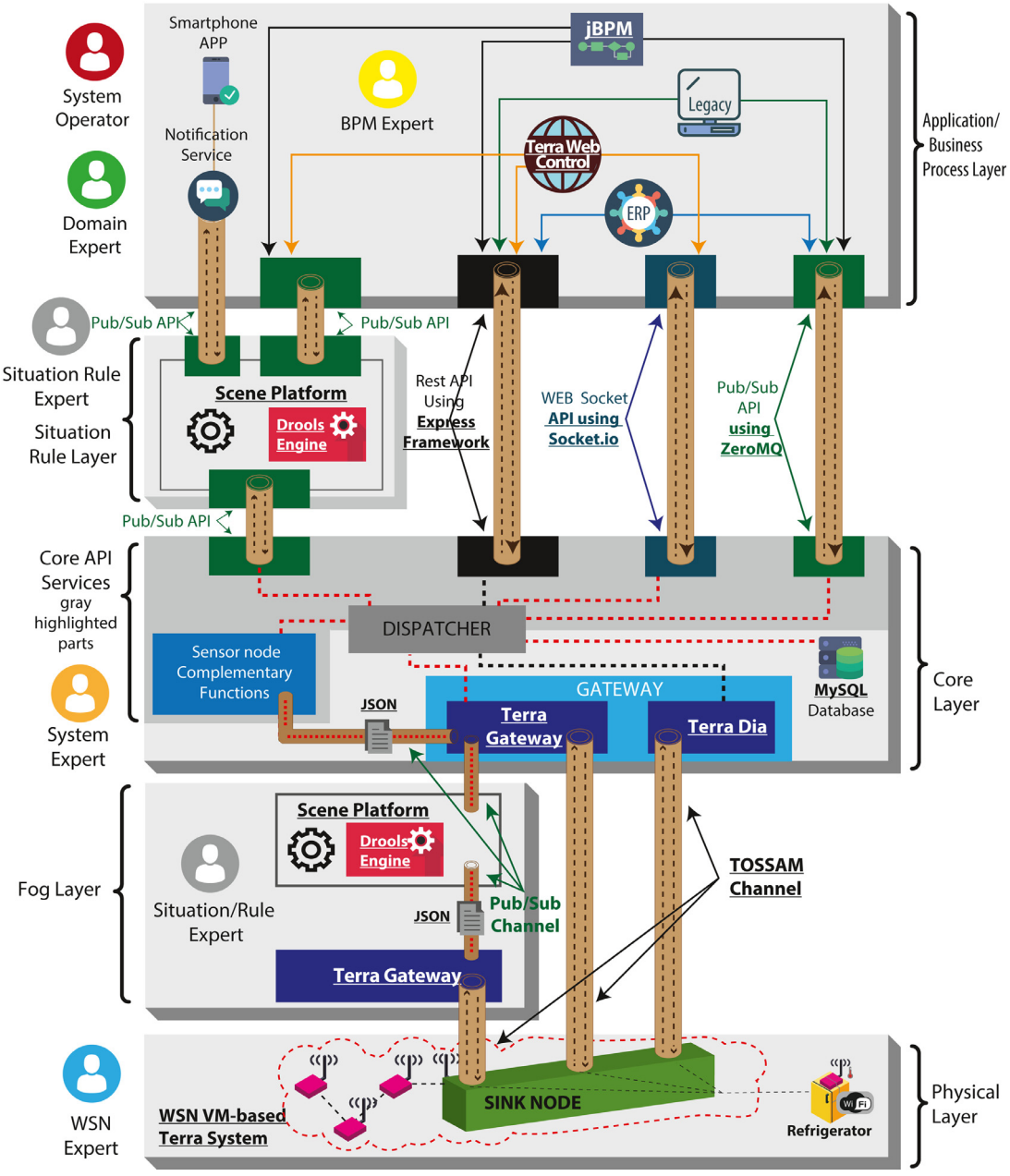

Fig. 4. LAURA applied architecture overview.

WSN nodes, contributing to achieving requirement R2. TerraDia interoperates with the Sink node to transmit the bytecode blocks using the native Terra System dissemination process.

The 'Terra Core' API services is illustrated in Fig. 4 by gray highlighted parts. It is a JavaScript Node.js (Expressjs, 2017) application module, responsible for providing the three standardized communication interfaces (API REST, Pub/Sub and WebSocket) defined in this layer. For the Pub/Sub API, ZeroMQ library (ZeroMQ, 2017) was the deployment choice. For the WebSocket API, the Socket.io (Socket.io, 2017) library was selected. Finally, for the REST API, the Express Framework (Expressjs, 2017) was employed.

In addition, there is a permanent Pub/Sub channel between 'Terra Gateway' and 'Sensor nodes complementary Functions' that allows decoupling and accelerates message forwarding. This process will only occur in the absence of a 'Fog Layer'. Otherwise, the conversion process of an AM packet to a JSON object will have already taken place when the packet is received by the Terra Gateway in the 'Fog Layer'.

The implementation of the 'Dispatcher' aimed at performing the functions, actions, and features previously described in Section 4.3.3 (Core layer). Thus, after converting the data to a useful measure and updating the JSON object, the Dispatcher is invoked with the purpose of forwarding the message containing the sensed data to the deployed 'Terra Core' APIs. In addition, the data are persistent with the recording in a MySQL (MySQL, 2017) database. As aforementioned, such persistence, in the database, is justified by the need for applications to access and maintain the data for a certain period of time.

The 'Situation/Rule Layer' basically takes advantage of SCENE situation platform (Costa et al., 2016; Pereira et al., 2013) to provide situation detection and lifecycle management. SCENE is a rule-based platform adopted by LAURA that connects CEP-based

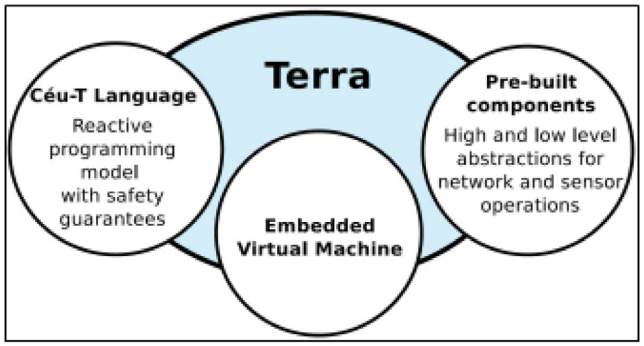

Fig. 5. Terra System basic elements (Branco et al., 2015).

and situation-aware approaches by leveraging the Drools general-purpose rule system and Drools Fusion CEP module as a mechanism for situation detection and lifecycle management. The Fusion module offers an event schema, stream processing, temporal correlation and time-window operators, seamlessly integrated with its Drools Rule Language (DRL).

From a deployment perspective, data are received by SCENE platform from lower layers via Pub/Sub API using ZeroMQ library and are readily processed in accordance with the situation rules previously defined. The detected situations are then made available as services to final Situation-Aware applications.

Situation types are specified in SCENE by means of structural and behavioral aspects, which are realized by Situation Classes and Situation Rules, respectively (Costa et al., 2016). Situation Class specializes the predefined class SituationType.

Situation specification in SCENE occurs as follows. Firstly, SCENE requires the definition of structural aspects. In this phase, a user-defined Situation Class should specify the specific roles played by domain entities in that situation type. For example, in a Fever Situation type, if the domain entity Person is playing a role febrile, it should be explicitly defined as such. In SCENE approach, situation properties are tagged as roles using the @SituationRole Java annotation.

When a situation is activated, a situation fact is inserted in the working memory representing that specific situation occurrence. From this point on, SCENE starts the lifecycle management of that particular situation.

As previously mentioned, the 'Application/Business Process Layer' encompasses a set of capabilities that aims at facilitating the communication between a wide variety of final applications and LAURA lower layers via available APIs. These facilities are made available through a 'Management Tool', which assumes the role of centralizing element of the layer, i.e., this tool provides a portal to a set of LAURA services. In addition, the Management Tool offers dashboard reports and graphs to aid the work of the IoT expert who is usually responsible for maintaining and monitoring the solution.

The 'Management Tool' is actually a 'WEB APP', which assists the users in the management of their solutions, offering centralized management facilities of all the endpoints with the respective permissions and access keys according to the stakeholder profile. With the use of this tool, it is possible to: (i) identify, test and evaluate the data received, identifying its origin and performance information; (ii) associate the endpoints with the access keys of the stakeholders, according to established profiles and permissions; (iii) use a user-friendly interface to upload new codes (bytecodes) that will be automatically disseminated through the Core Layer's 'Dissemination Module'; (iv) use a node code generation solution for WSN from BPMN specifications, allowing the BPM expert to generate codes that run on the LAURA VM nodes (Agrizzi, 2018).

One of LAURA's application-level facilities is the 'Terra Web Control' (TWC). TWC uses a Websocket API and a RESTful API to access the communication interfaces available on 'Terra Core'. TWC provides key features to the IoT expert by generating data traffic and other relevant information regarding the running of the IoT devices. In addition, if a problem or error in the application occurs, it can also be used as an alternative way to monitor the physical entities data via the API WebSocket and Rest. It is also able to interoperate by means of the API Pub/Sub, receiving triggered situations from the SCENE Platform.

Another application-level facility is the 'Terra Dia App', a Web application that offers the mean by which the IoT expert can remotely upload a new node code. This Web application establishes communication with *TerraDia*[1] module in Core layer, via WebSocket API, uploading a new node code that will then be transmitted remotely to the network nodes. The source code and additional documents about TWC and TerraDia are available in the repositories of LAURA project at GitHub (Teixeira et al., 2017a). A step-by-step guide explaining how to download and execute the ready-to-use LAURA VM is available in the repository 'how-to-use-laura'[2] of the same GitHub address.

Another example of LAURA's application-level facility is 'TerraGen' (Agrizzi, 2018), the LAURA's solution for node code generation from BPMN specifications. 'TerraGen' provides BPM experts, who have basic programming skills, a way to generate WSN node codes. 'TerraGen' implements an approach for transforming BPMN 2.0 language elements into Céu-T reactive language constructors, allowing the generation of codes or code blocks with equivalent and coherent semantics in Terra System, the virtual machine used in LAURA architecture. The source code and other '*TerraGen*'[3] documents are also available at the LAURA's project repository.

4.5. Security and privacy considerations

Security and privacy aspects were not deployed in the current version of the architecture. This section will outline some ways to address security and privacy aspects in LAURA so that they can be explored in future studies in order to continue or improve the proposed work.

Privacy or security failures can damage the operation of IoT applications, especially when they involve a heterogeneous environment using wireless communication. For example, in Healthcare Systems it is important that the patient's vital data are not exposed or accessed improperly to ensure their privacy and to avoid making the wrong decisions that can put on risk his/her health or life. Improper access or manipulation of data/rules on HVAC systems by intruders may affect equipment integrity or cause damage to persons in an environment where ventilation/heating parameters do not function properly. Interference or intrusion into Smart Traffic Systems can lead to accidents or even death. A damage Fire detection system can generate undue alerts or cause tragedy by failing to alert or trigger fire-fighting equipment in a proper time. The manipulation or any improper interference made by any hacker in any type of IoT System can cause inconvenience, financial loss or risk to people's lives.

Security issues involve requirements such as Availability, Integrity, Confidentiality, Privacy, Authentication, Non-repudiation, and Freshness (Pietro et al., 2014; Sha et al., 2018) as well as the use of key-based cryptographic mechanisms or techniques that will increase complexity and the communication processes from

[1] https://github.com/laura-architecture/terra-dia.
[2] https://github.com/laura-architecture/how-to-use-laura.
[3] https://github.com/laura-architecture/terra-generation.

lower-level aspects involving sensor devices and networks to the higher-level aspects related to business processes or applications (Sha et al., 2018). These techniques can prevent or minimize attacks that may damage the security or privacy of systems, such as Eavesdropping, Denial of service, Message tampering, Selective forwarding, Sinkhole attacks, Wormhole attacks, Sybil attacks (Baronti et al., 2007).

The use of encryption in WSN based on IEEE.802.15.4 standard has some limitations because it uses devices designed for low processing, energy concerns, and low bandwidth. In this case, we commonly use encryption based on symmetric keys. In Terra-based system devices as proposed by LAURA, secure communication can be done by the use of TinySec (Karlof et al., 2004), MiniSec (Luk et al., 2007), SNEP (Perrig et al., 2002) or TeenySec (Dener and Bay, 2016). TeenySec is a more recent work supporting requirements as data confidentiality, data integrity, data freshness, and data authentication. In the case of use more complex resources, it is possible to deploy new communication layers with new hardware platforms in Terra System. For example, there are Terra versions that support UDP/IP protocol with WiFi IEEE-802.11(Branco, 2017) networks.

One way to provide encryption communications from data reception in the Fog and Core layers of LAURA via TerraGateway module until the 'Application/Business Process' layer is through the use of https connections across the data flow and connection between the various modules and components of the architecture, including the APIs until the data reception and presentation in the final application (Rescorla, 2000). In addition, ZeroMQ (ZeroMQ, 2017) and socket.io (Socket.io, 2017) library and Express (Expressjs, 2017) Framework used in the Fog, Core and Situation/Rule layers provide security features. For example, ZeroMQ provides mechanisms that allow authentication and confidentiality in the communication between the client and the server.

According to (Bassi et al., 2013) a secure communication infrastructure can be obtained by using hop-to-hop or end-to-end encryptions. They proposed some tactics and the corresponding strategies to deploy privacy features: (i) pseudonymization, in order to replace the identification of the sensed data with an artificial identifier (ID) that makes it difficult to identify the origin or the recipient. In the specific case of LAURA, it's possible to use a universally unique identifier (UUID) (Leach et al., 2005) generated by the final application as the identifier and an arbitrary ID for each sensor or actuator. The ID will only be known by the end-user with specific permission to access the data stored in encrypted form; (ii) Avoid transmitting ID clear, using encryption in the entire information flow from the devices to the final application, including wireless communications. In this case, it is possible to use the encryption strategies as previously mentioned and the UUID to make it difficult to identify the destination of the data. The purpose is to allow identification and localization of the device only by the final application seeking to prevent unauthorized access.; (iii) Enable privacy setting in order to manage the access control.

These are straightforward ways to address security and privacy in LAURA for sensitive IoT applications. For further research in this direction we suggest a more rigorous study starting at (Bassi et al., 2013). Depending on the requirements or specifics of the applications, it is necessary to evaluate contradictory aspects, as performance may be compromised. In addition, there is a pressure between privacy and security, as authentication mechanisms must be used to establish trust, but, at the same time, privacy must be maintained without exposing device and user data. Following the LAURA architecture proposal as a simpler way of building the final IoT Applications, the security and privacy proposals presented seek to: (i) Use end-to-end encryption approaches for all applications in order to simplify the deployment of the entire solution; (ii) pseudonymization the identification of the sensed data in such a way that it is not possible to identify the source or recipient of the data; (iii) Use authentication and encryption mechanisms, establishing the relationship between source, destination and users' access permission only in final applications, preventing the identification of the user's data in case of an interception of packets during the data flow.

5. Application scenario, model applications, performance evaluation, and empirical evaluation

The application scenario proposed here is an IoT application called Medicine Quality Control (MQC). MQC is a situation-aware application for real-time monitoring of medical products such as Botulinum Toxin Type A (onabotulinumtoxinA) bottles that will be used for esthetic or medical treatments by dermatologists. The bottles must be maintained within a specific temperature range inside the refrigerator or freezer.

MQC is able to identify undesired situations, alerting the system users in real-time and allowing them to respond and potentially preventing product loss. For example, in the case of power loss, the application may stop receiving sensed data and be programmed to send an alert requiring a response from the user. According to this response, the 'Situation Platform' can even trigger new situations. This allows constant interaction and greater sensitivity to real-time context.

The same scenario was used to create a BPMS Situation-Aware application. In this scenario, processes are modeled in BPMN by the BPM expert, reacting to the situations previously defined by the Situation/Rule Expert. The starting event begins when a predefined situation is triggered. The number of modeled processes in BPMN will depend on the actions required by the Domain Expert and suggestions from the other stakeholders. A process can specify alert actions for the user, such as, email, SMS or a message sent to other systems.

The following sections present a proof of concept of the LAURA Applied Architecture based on the proposed application scenario. This proof of concept is composed by the following: two modeling application implementations and, a performance and an empirical evaluation experiment.

5.1. MQC situation-aware modeling application

This modeling application intends to show the main steps for building a final situation-aware application (called MQC application). We assume that the ready-to-use LAURA VM is configured according to the tutorial available in the repository 'how-to-use-laura' at GitHub.

In accordance with LAURA domain reference model, the '*1st Step*' encompasses the requirement analysis made by a System expert, who draws a UML class diagram that represents the domain application scenario in terms of its *entities, static attributes, intrinsic contexts* and relations to other entities through *relational contexts*. Fig. 6 presents the UML model developed for the proposed domain, using a simple stereotype entity-context terminology.

The entity with a kind *Person* (that comes to be the user of the application) wants to *observe* zero or more *Products* that are sensitive to variations in temperature. Thus, a relational context *Observation* exists between them. Furthermore, the *Product* must be kept in a pre-determined temperature range, according to the manufacturer's recommendations. A *Container* has a *Temperature* control system (which translates in an intrinsic context *Temperature)* and it can store zero or more products, therefore a relation context of containment or *Storage* exists between a *Product* entity and a *Container* entity. In this way, the temperature of the *Product* can be considered the same as that of the *Container* in which it is stored. Both, *Container* and *Person* present an intrinsic context *Geolocation,*

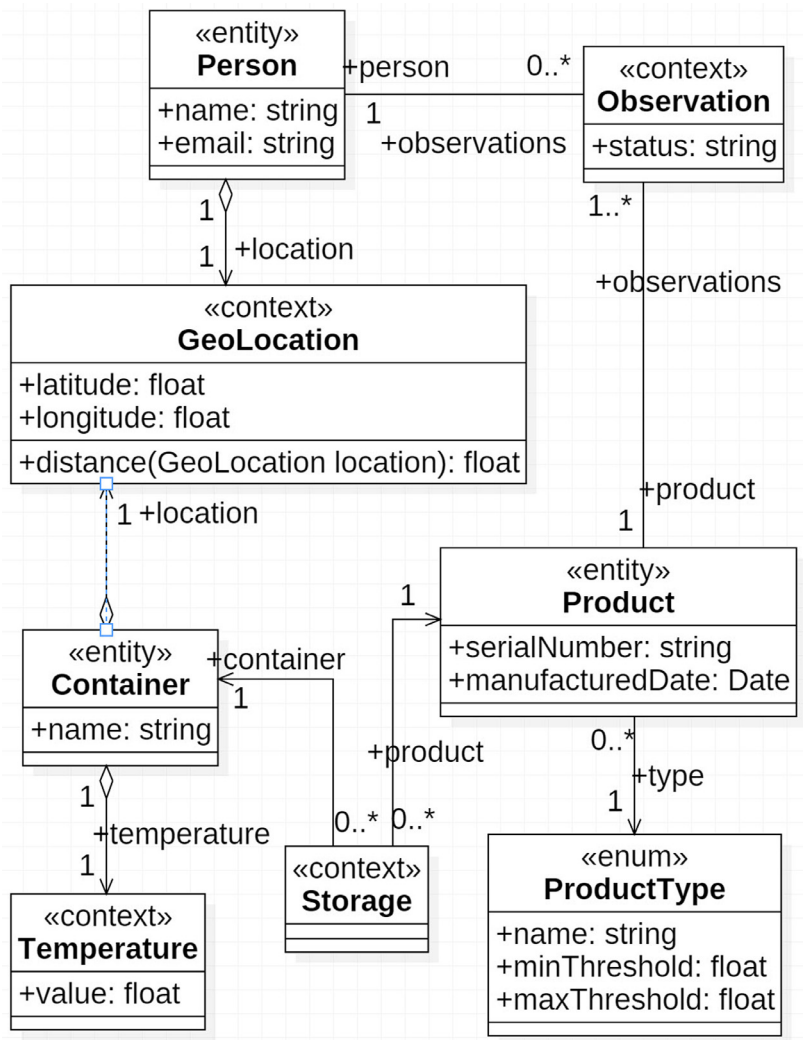

Fig. 6. The UML class diagram that represents the model of the MQC scenario.

in order to provide information like the distance between an *observer* and where the *observed* product is being stored.

According to the application scenario, the dermatologist needs to monitor vacuum-packed in sterile glass bottles of *onabotulinumtoxinA* that are stored in two freezers. One of the freezers is in the office and the other one is at home. Each freezer has a DS18B20 (Integrated, 2018) waterproof temperature sensor connected to an Arduino (Arduino, 2016) board.

The *MQC modeling application* serves as an interaction point with the doctor, secretary or whoever is in charge of watching and keeping the previously mentioned products safe, delivering alerts, notifications or even asking for user's immediate actions in order to prevent any loss.

To interoperate with this application, a *MQC service* is built on top of the *LAURA Context Broker*, a service that fits into the Situation/Rule layer, and that provides a RESTFul API to manage the context-aware scenarios revealed by the previously presented domain reference model. Working along with a situation inference engine (SCENE), the MQC *service* can go from subscribing to events from sensing device's properties (*ObservableProperties*) through feeding domain entities *contexts* and composing them into meaningful *situations*.

The *'2nd Step'* comprises the task performed by the System expert in order to register the entities and context derived from the domain. The model was realized in *LAURA Context Broker* through its RESTFul API methods, first inserting all entities instances (*Person, Product, Container)* and context instances (*Temperature, Geolocation, Observation and Storage)*.

The exchange format for all API operations is JSON. Table 2 presents a JSON representation for an entity instance of the *'Person'* kind. For the sake of objectivity, the payload schemas are left outside this work. More detailed schemas for all the payloads expected by the API can be found in the *'LAURA-Context-Broker'*[4] repository of the LAURA project at GitHub.

Once the entity-context model was set, the *'3rd Step'* encompasses the *binding* of contexts to appropriate virtual sensors (provided by *Terra Gateway*), i.e., associating a particular context to a particular observation provider (a sensor node with unique identifier) so its observed values can be automatically fed into the *Context Broker* once the *Terra Gateway* receives new readings right from the physical device.

The *'4th Step'* comprises the activities of modeling and creation of situations by the Situation/Rule Expert according to the analysis of the domain problems and requirements in the view of the Domain expert. Eight specific situation types were identified: (i) '*Observed Situation*', characterized when a Person indicates his/her interest in monitoring a specific product. (ii) '*Absent Sensor Reading*' that describes a situation in which there is a breakdown in communication or data reception for a pre-determined period of

[4] https://github.com/laura-architecture/LAURA-Context-Broker.

Table 2
LAURA Context Broker JSON Example for Person entity.

```
{
  "id": 56534,
  "kind": "Person",
  "descriptor": "Dr. Mary Sue",
  "contexts": [
    {
      "id": 6553,
      "kind": "Geolocation",
      "value": {
        "timestamp": 1531208030,
        "entries": {
          "latitude": -20.3582821,
          "longitude": 40.333145
        }
      }
    }
  ],
  "attributes": {
    "name": "Mary Sue",
    "email": "marysue@laura.com",
    "role": "Doctor"
  }
}
```

Table 3
The ***AbsentSensorReading*** situation declaration in SCENE.

Line	Instruction
01	declare **AbsentSensorReading** extends **Situation**
02	container: Entity @part
03	end
04	rule **AbsentSensorReading**
05	@role(situation)
06	@type(AbsentSensorReading)
07	when
08	temperature: IntrinsicContext(kind=="temperature")
09	container: Entity(kind=="Container") from
10	temperature.bearer
11	not (
12	ContextValue(context == temperature)
13	over window:time(1m30s)
14	)
15	then
16	SituationHelper.situationDetected(drools);
17	end

time; (iii) '*Exceeding Threshold* Situation', when a temperature measurement is beyond the acceptable range for a certain product; (iv) '*Exceeding Safety Distance Situation*', when a Person is at a distance from the Container that is far to be considered safe; (v) '*Estimated Time of Arrival (ETA) Greater than Estimated Time-to-Threshold (ETT) Situation*', when the estimated time for a Person to reach a Container is greater than the estimated time that it would take for the Container to reach an unsafe temperature previously defined for that product; (vi) '*All Observers are with ETA Greater than ETT Situation*', when all Observers are found in situation (v); (vii) '*Busy Situation*', when an Observer indicates that he/she is busy or unavailable to act in response to a situation (vi); and (viii) '*Attending* Situation', when an Observer indicates that he/she is available to act in response to a situation (vi). Table 3 presents the specification of the '*AbsentSensorReading*' situation in SCENE. The complete list of situation type definitions can be found in the 'experiments'[5] repository of the LAURA project at GitHub.

All SCENE situation type specifications were written in a single DRL file which was then uploaded to the *SCENE Situation Engine*. From that moment on, the *MQC Service* is all set and able to detect new situations based on incoming sensor data. In addition, the *MQC Service* provides a *Situation Subscription API* along with a specific *websocket* channel for client applications and/or services to listen for alerts about situations they are interested in.

[5] https://github.com/laura-architecture/experiments.

Table 4
Overview of part of the running process code that establishes the connection between jBPM and SCENE.

Line	Instruction
01	Switch(situation)
02	{
03	Case "AllObserversETAGraterThanETT":
04	CreateRuntimeManager
05	("AllObserversETAGraterThanETT.bpmn2");
06	RuntimeEngine engineAllObserversETAGraterThanETT
07	=getRuntimeEngine();
08	KieSession ksessionAllObserversETAGraterThanETT =
09	engineAllObserversETAGraterThanETT.getKieSession();
10	ksessionAllObserversETAGraterThanETT.startProcess(
11	"MQC.AllObserversETAGraterThanETT");
12	break;
13	Case "AbsentSensor":
14	....

5.2. MQC business-aware modeling application

This modeling application is based on the same application scenario and situation types that were previously presented. The purpose of this application is to show that it is relatively simple to configure a business-aware application using LAURA architecture. We assume that the activities performed by the Situation/Rule and System experts in the section above were successfully performed. The following paragraphs present the main steps for configuring a final BPM application from the point of view of the BPM expert.

The '*1st Step*' comprises the activities of specifying and modeling BP in BPMN in order to define the actions that must be executed from the triggered situations. Based on the previously defined situations, it is necessary to define the processes that must be executed to achieve the desired goals. The following steps present examples of processes for this application scenario using a java-based tool called jBPM.

The '*2nd Step*' comprises the activities of configuring the communication channels between the jBPM and the LAURA architecture in order to enable interoperability between the APIs of the Situation Layer (SCENE platform) and the jBPM. Therefore, to start the processes from situation activations in the SCENE platform, a permanent Pub/Sub communication channel is established through a websocket connection. In addition, a REST call is needed to query situations and its participants. In this way, jBPM subscribes to SCENE and waits for messages containing the name of the situation that is configured in SCENE. As soon as messages are received, the situation name is verified to select the commands to be executed.

Another activity that should be performed in this step is programming the process code to establish the connection between jBPM and SCENE. According to the received message, this process executes a loop to identify the situation that should be triggered. It is expected that the System expert makes the initial programming of this code and that the BPM expert changes or includes a new 'case' block code that appears soon after the 'switch(situation)' command, as can be seen in Table 4. Note that each 'case' indicates the process that will be started from the triggered situation, indicating the file with extension '.bpmn2' that contains the processes such as the BP that will be presented in Fig. 7.

The line that starts with 'CreateRuntimeManager' creates all processes into the knowledge base with the default settings. Lines 6 and 8 create a running instance of the Rule

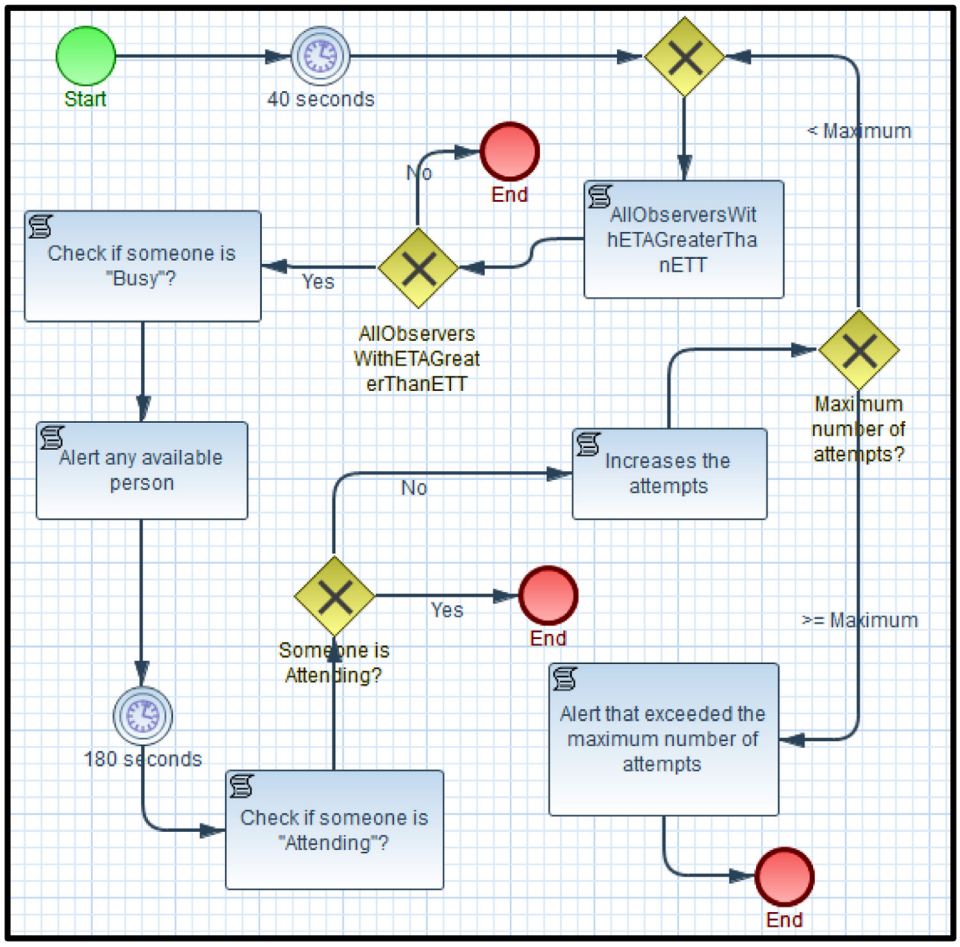

Fig. 7. The process in BPMN that started when the situation 'vi' is activated.

Engine as a KieSession. Finally, line 10 initializes the process 'MQC.AllObserversETAGraterThanETT' into the session previously created.

The *'3rd Step'* comprises the activities of configuring the BP in BPMN that have been modeled in the *'1st Step'* in order to access the communication channels to interoperate with the MQC service ('Situation/Rule Layer') as previously reported.

When working with a BP from a triggered situation, it is expected that there are several processes running simultaneously, since we may have more than one situation occurring at the same time. The BPM expert models processes defining the actions that should be performed when situation notifications are received. Some processes are simpler or more direct and others demand more actions according to the situation.

For the sake of brevity and objectivity, only the most common BPMN elements were used. Thin-border circles labeled 'Start' and thick-bordered circles labeled 'End' indicate, respectively, the start and end of a process. In the proposed approach, the 'Start' events are triggered when a situation is triggered. It was also used time events that are associated with the circles with the image of a clock and the description of the time that will elapse during the flow between two elements as can be seen in Fig. 7.

The diamond shape represents the gateway element that is used for divergent or convergent flow control behavior in a process. The rounded corner rectangle represents the activity that is performed in a BP. This activity can be atomic or non-atomic. The activity can be a task or a sub-process. In this application, we used only the '*Script tasks*' that is executed by a system. Finally, the arrow connecting two elements is used to represent a sequence flow direction.

Fig. 7 presents the process that is started when Situation 'vi' - All Observers with ETA Greater than ETT Situation' is activated, that is, when everyone monitoring the product is not at a close enough distance to prevent the product from spoiling. There is a 40 s delay to avoid a false alert. After 40 s, a check is made to see if Situation 'vi' is still active. If not, the process is terminated, as indicated by a thick-bordered circle labeled 'End' at the top of the Fig. 7. If the situation remains active, a check is made to verify if there is someone in Situation 'i' (a Person who has indicated interest in monitoring the product). At the same time, a simultaneous check is made to verify if Situation 'vii' is active (when an Observer indicates that he/she is busy or unavailable to act in response to a situation), making it possible to alert all available parts, as shown on the left side of the Fig. 7.

Once a person has been notified, there is a waiting time of 180 s to allow a minimum response time until he/she sees and responds to the alert. After the delay, a check is made to verify if a response has been received indicating there is at least one person available and able to solve the problem in both Situations 'i' and 'viii' (when an Observer indicates that he/she is available to act in response to a situation). In this case, the process is terminated, as indicated by a thick-bordered circle labeled 'End' in the middle of the Fig. 7.

If there is no available observer, there is an increase in the number of attempts. A total of 20 attempts were stipulated, corresponding to approximately one hour (maximum time that justifies

warning before the product is lost). Until reaching the maximum number of attempts, the process returns to the initial checkpoint and continues to verify if the Situation 'vi' is active. Once the maximum number of attempts has been reached, a final alert is sent to all observers and the process terminates, as shown on the right side of the Fig. 7.

This Modeling application has shown that it is relatively easy for the BPM experts to configure their own business-aware application to use LAURA architecture, without having to know the lower-level technical aspects.

5.3. LAURA architecture performance evaluation

The evaluation presented here aims at verifying if the proposed architecture is capable of processing and delivering sensed data to final applications within a satisfactory end-to-end latency. The evaluation consists of various experiments using different topologies, configurations and sample times. Measurements (in milliseconds) were taken to record the time that the package takes to be processed between the Terra Gateway and the Terra Web Control final application. Breakpoints to register time are placed at data arrival in the Terra Gateway and in the Terra Web Control. The difference between these time measurements is calculated in order to find the length of the latency. The average latency is calculated using the simple arithmetic mean based on the latency of each received package. An average latency can be calculated for temperature and another for the luminosity.

This evaluation consists of three experiments: Experiment 3.1 evaluates the average delay when packets are sent in bursts to Terra Gateway. Experiment 3.2 evaluates the average delay in a real-world application scenario comparing different sample times. Experiment 3.3 evaluates the influence of the dissemination process in the running application. New node codes (bytecodes) are disseminated into the WSN to observe if there is any significant delay or loss of sensed data in the running application during this process.

These experiments analyzed latency for different types of topologies. Since Terra System is the WSN chosen for the LAURA applied architecture, we have used Terra-based WSN topologies. Moreover, an average latency can be stipulated for each WSN and an overall average measurement for all WSNs used in this evaluation. In addition, it was not taken into account the latency between the reading of the sensed data in the node until the arrival of the package in the sink node. According to (Delsing et al., 2010), a WSN based on IEEE 802.15.4, with a transmission rate of approximately 250kbps, has a latency variation of 5 to 20ms for each communication hop also taking into account any communication interferences. Therefore, the average delay when processing sensed data using LAURA architecture is equivalent to one more communication hop in a WSN. Thus, if we add 20ms (for the worst-case hop) to the final latency obtained in our experiments, we will have the end-to-end latency of the LAURA applied architecture from reading the sensed data until the arrival of the data in the final application.

The LAURA architecture proposal is designed to be a generic platform, capable of supporting different application types with different requirements, mainly in interactive applications that are typical in IoT application scenarios. Most users of different types of low-latency interactive multimedia streaming applications find acceptable a delay of 40 ms. Taking into consideration that the time between a mouse click until the message arrives in the application is at least 30 ms, 70 ms is an acceptable delay for these applications (ITU-T, 2014). There is no fixed threshold for acceptable delays in interactive applications, however, an end-to-end latency of less than 100ms is considered acceptable for most interactive applications (Mandjes et al., 1999). For example, the high-quality gaming scenario experiences responses of at least 100 ms (Petlund, 2009). The ITU G.114 recommendation (Brosh et al., 2010) states that the maximum delay (worst case scenario) required for interactive application is 400 ms. For an audio conferencing application, ITU-T guidelines indicate that users are satisfied with a latency of less than 150 ms (Petlund, 2009).

In Experiment 3.1, we evaluate the average latency in high traffic conditions determining the average delay in a data burst scenario to Terra Gateway. This experiment was performed with three Terra System-based WSNs, containing a Sink Node linked to a Mib600CA (MEMSIC, 2016) gateway with Ethernet connection in each WSN. Each node was fitted with temperature and luminosity sensors. All WSNs were connected by Ethernet connections to a computer running Terra Gateway, Terra Web Control and other components and specific modules in LAURA applied architecture. Table 5 presents each WSN configuration, the components and extra content related to the experiment 3.1. All results, additional data and information about experiment 3.1 are available in the 'Experiment'[6] repository in the LAURA Architecture GitHub Project. Each experiment or sampling period created a text file for each WSN containing all the data packages received in the Terra Gateway. All the data generated in the modeling applications and experiments are also available in GitHub.

Each node was programmed to send bursts every 5 s to its corresponding Sink Node. Each burst had 5 temperature readings within 800 ms intervals. The reading ranges used in our experiments were chosen because they are smaller than what is normally used for a group of applications similar to the application scenario. This allowed Terra Gateway to receive 15 packages every 5 s from 3 WSNs containing the temperature readings. Over a 5 min period, 277 packages were captured from WSN1, 335 packages from WSN2 and 363 packages from WSN3. The difference in the number of received packages was expected due to each node having a different processing capacity. The Terra node code used in this experiment is also available in the same GitHub repository mentioned above. The average delay of all WSNs in this scenario was 11 ms. Adding a 20 ms for the latency within the WSN (from the reading of data to the sink node) we have an end-to-end latency of 31 ms. Therefore, the end-to-end latency observed in this evaluation was 70 ms or 100 ms, as previously discussed. Furthermore, non-interactive monitoring real-world IoT applications as previously presented in Fig. 1 of the introduction section are usually more flexible and do not require such a short latency.

Experiment 3.2 evaluates the average delay in WSNs topologies, such as the one depicted in Fig. 8. In order to evaluate the communication performance, it was decided to configure each WSN with a linear topology following the specifications of Alfayez et al. (2015). We chose the use of Linear topology for this experiment given the latency and energy consumption challenges imposed by these types of WSN. For example, the linear topology limits the number of neighbors that a node can have or forces a single route in which data are sent from one node to the next. Thus, the nodes closer to the Sink node tend to get overloaded and can even lose packages.

This experiment measured temperature and luminosity using the same nodes from experiment 3.1. However, we have used an increased number of nodes within each Terra System WSN as shown in Fig. 8. Table 6 presents each LWSN configuration, components, and other information. All results, data and additional information about experiment 3.2 are available in the 'Experiments'[7] repository in the LAURA Architecture GitHub.

[6] https://github.com/laura-architecture/experiments/tree/master/Experiment-3/Exp3.1.

[7] https://github.com/laura-architecture/experiments/tree/master/Experiment-3/Exp3.2.

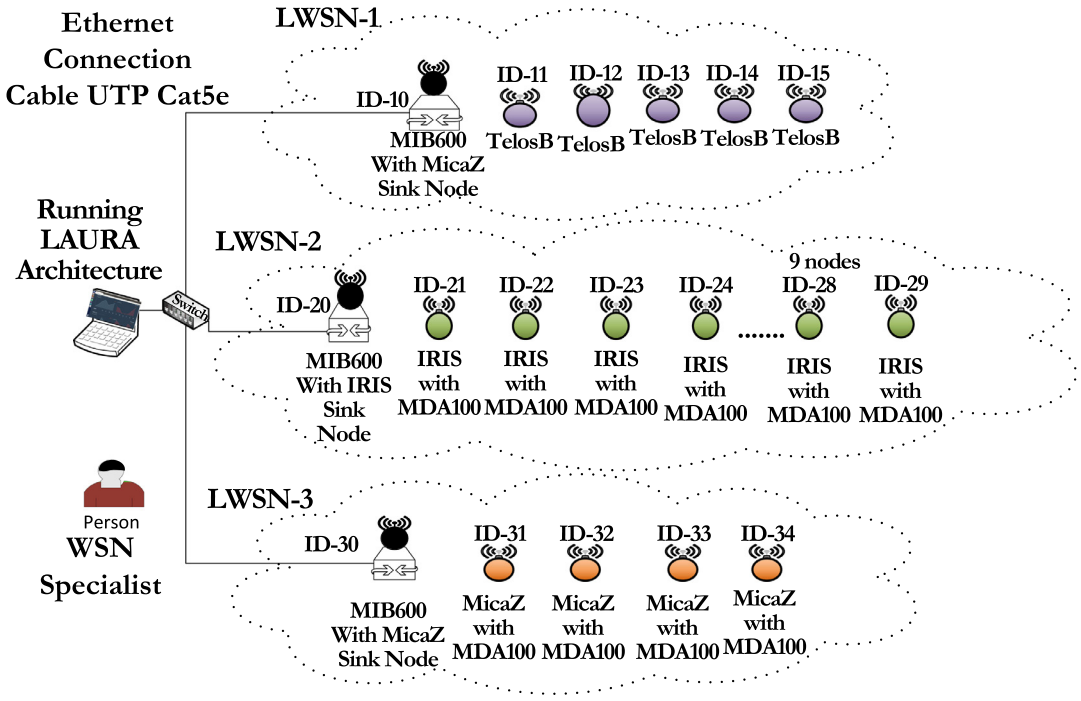

Fig. 8. Performance evaluation- Experiment 3.2 e 3.3 – Scenario topology with 3 linear WSNs.

Table 5
Performance evaluation – Experiment 3.1: Throughput analysis in high traffic conditions for 5 min.

WSN	Sink node	Unique Node	Temperature Average Delay
01	Mib600ca + Micaz - ID:10	TelosB – ID:11	11 ms
02	Mib600ca + IRIS - ID:20	IRIS – ID:21	10 ms
03	Mib600ca + Micaz – ID:30	MicaZ – ID:31	11 ms

Table 6
Performance evaluation – Experiment 3.2: Throughput analysis in typical LWSN applications for 5 min.

Sampling times: Temperature every 5s and Luminosity every 8s			
WSN	Sink node	Nodes	Average Latency
01	Mib600ca + Micaz - ID:10	TelosB – ID:11 to 15 (5 nodes)	Temperature: 14 ms Luminosity: 12 ms
02	Mib600ca + IRIS - ID:20	IRIS with MDA100 sensor - ID 21 to 29 (9 nodes)	Temperature: 13 ms Luminosity: 16 ms
03	Mib600ca + Micaz – ID:30	MICAZ with MDA100 sensor - ID 31 to 34 (4 nodes)	Temperature: 17 ms Luminosity: 14 ms

In this experiment, the measurements used three different configurations for the sample times. Table 7 shows temperature readings taken every 5 s and luminosity readings every 8 s per node for each WSN. The average data transfer latency was 14 ms after a period of 5 min. This average delay can be considered satisfactory according to the same arguments previously presented in experiment 3.1. While recording the length of the delay in the Terra Web Control, it was observed that, over a shorter period of under 5 min, the differences among average delays were small, with a tendency to remain stable over longer periods. Therefore, 5 min was considered enough time to make a reliable calculation of the average delay time. This was also true for the other experiments in the performance evaluation.

Table 7
Performance evaluation – Experiment 3.2: All general average **delays** of each **sampling time for** 5 min.

Sample	Sampling times	General Average Delay of all WSN
01	Temperature: every 5 s Luminosity: every 8 s	14 ms
02	Temperature: every 35 s Luminosity: every 38 s	6 ms
03	Temperature: every 55 s Luminosity: every 58 s	6 ms

The Terra node code used in these experiments and the other data and information are available in the same GitHub repository previously mentioned along with all the sensed data captured for each sample time. Table 7 shows the general average delay of all WSN for each sampling time executed for 5 min.

These experiments indicate that the LAURA Architecture introduces an average latency that is acceptable for the groups of applications we have considered. Our experiments were performed under a controlled environment to simulate the behavior of real-world applications; however, we have introduced more intense traffic than what is usually observed in this particular group of applications. Fig. 8 summarizes the Scenario topology used in experiments 3.2 and 3.3.

Experiment 3.3 evaluates communication performance under unusual conditions that may occur in IoT Real-World application scenarios when it is necessary to update the code of the nodes simultaneously to the operation of the network. For this experiment, significant loss of sensed data was registered. The bytecode from Sample 3 (see Table 7) replaced the code from Sample 1 in all nodes of the LWSNs. Bytecode dissemination lasted an average of 16 seconds for each LWSN. In this way, it was possible to verify that approximately 3 temperature and luminosity readings have been lost due to the sampling time of sample 1. Dissemination time and package loss were considered satisfactory as the dissemination of a new bytecode is something that can be programmed when stakeholders are able to monitor the process avoiding any days or moments when data loss would be critical. Moreover, this process allows the node codes to be updated in a faster and more practical way that a manual update process. All results, raw data, additional information and some videos that show the realization of experiment 3.3 are available in the 'experiments'[8] repository in LAURA Architecture GitHub.

5.4. Empirical evaluation

This section presents an empirical evaluation of LAURA architecture from the point of view of two of its stakeholders: the BPM Expert and the System Expert. This evaluation was conducted from December 2018 to February 2019, with a sample of 30 participants divided into two groups, according to their professionals' profile.

To select the participants, open invitations were sent to BPM professionals through associations such as ABPMP (ABPMP International, 2018) and other affiliates scattered around the globe. In addition, invitations were sent to system developers in universities and companies in general mainly through LinkedIn.

The entire process of planning, executing and analyzing the results of this evaluation was based on Juristo and Moreno (2010) and Wohlin et al. (2012). The steps of the experimentation process are: (i) Definition of a general hypothesis and transformation of this hypothesis into evaluation questions; (ii) Definition of the scope, planning and description of the experiment design; (iii) Setting up the profile and the 'point of view' of the participants; (iv) Setting up a period for the evaluation in which the participants perform the proposed experiments and evaluate the LAURA architecture; (v) In this step, data are collected, tabulated and categorized; (vi) This step refers to data validation to avoid inconsistencies, possible errors or failures in the entire evaluation process; (vii) During this step, the data are analyzed and interpreted to verify if the results are enough to confirm the hypotheses; (viii) in the last part of the evaluation process, the results are presented and discussed. The details of each step of the empirical evaluation are presented below.

5.4.1. Hypothesis

The general hypothesis is that LAURA architecture simplifies the task of developing IoT applications for those stakeholders of LAURA 'Application/Business Process Layer' who are not aware of the lower-level technical aspects related to the development and deployment of IoT solutions.

This general hypothesis was divided into more specific hypotheses that were transformed into evaluation questions, using three factors of the Technology Acceptance Model (TAM) (Avilés-López and García-Macías, 2009; Venkatesh and Davis, 1996): the perceived application '*Ease-of-Use*', '*Usefulness*', and '*Intention*' of use of the LAURA architecture. Additionally, a question was elaborated to verify if the presented problem is equivalent or comparable to real-world problems that normally are treated in a corporate environment. We use the Likert (Harpe, 2015) rating scale from *'strongly disagree'* as the value '1' to *'strongly agree'* as the value '5'. Table 8 presents the evaluation question and their respective rationales.

For each evaluation question presented in Table 8, it was made a qualitative question so that the participant could make relevant observations. Each open question with free text has the main objective of obtaining information from the participants that can contribute to the improvement of the evaluation (to be done during the pilot test) and mainly with comments that can contribute to the improvement of the LAURA architecture. Table 8 presents the qualitative question and its rationale that is made shortly after each evaluation question described in table 9.

For this Empirical evaluation, two evaluation forms were developed, one for each group of professionals - System Experts and BPM Experts - which served to guide the participants in conducting the experiment. The forms are very similar, but present some specific details for each group. The complete forms are available in Teixeira (2018a,b), respectively.

5.4.2. The scope, planning and design

The BPM expert is hired to develop a BPMN-based application called *Medicine Quality Control* (MQC), whose purpose is the real-time monitoring of medical products, as previously presented at the beginning of Section 5. In order to perform the experiment, the BPM Expert must use a BPMS (jBPM tool was used). According to the guidelines in its respective form, the BPM Expert must configure a business process described in BPMN to integrate it with LAURA, accessing sensed data and configuring the contextual situations defined in the problem specification.

The System Expert is hired to develop a similar final application, which is not BPM-based, but he/she must perform the same functionality as proposed in the application scenario. The System Expert must develop a client application using a programming language of his/her choice, to solve the same problem. Like the BPM Expert, the System Expert must follow the guidelines described in its respective form.

After defining the application scenario, the steps and guidelines necessary to carry out the evaluation experiment were defined. Initially, there was a validation phase of the evaluation forms with the support of the Pilot participants. During this phase, adjustments and improvements were made to the guidelines contained in the documents, since the participants reported a lack of information to be able to complete the evaluation. The constant concern, in this stage, was to adjust the forms so that the participants were able to understand the proposed problem and the necessary steps to make the evaluation, but without losing the essence that the proposed evaluation seeks to portray a scenario equivalent to those found in the corporate environment, in which the professional experiences adverse circumstances and needs to seek information to perform a certain task. In this way, the form was gradually improved in order to provide the minimum information necessary for the participant to make the evaluation.

In this step, it was also necessary to adjust the form in relation to the guidelines of the practical performing part of the experiment in the Virtual Machine (VM), since the pilot participants also reported difficulties in the use of VM LAURA. Examples of improvements made to the documents include improved API documentation and additional information about using jBPM.

5.4.3. The profile of the participants, execution and evaluation

The target participants of this empirical evaluation (BPM Experts and System Experts) are professionals who usually have little or no prior knowledge of lower-level aspects of IoT programming

[8] https://github.com/laura-architecture/experiments/tree/master/Experiment-3/Exp3.3.

Table 8
The evaluation questions and their respective rationales.

Evaluation question	Rationale
Taking into account your background professional experience, is the problem presented in the application scenario considered similar or equivalent to a real-world problem?	Taking into consideration that LAURA architecture aims to support real-world IoT solutions and considering that this evaluation proposes a simulated real-world scenario, it is important to know if the problem described in the application scenario is equivalent to those commonly found in an enterprise environment. The participant of this experiment is a qualified professional to confirm if the problem presented is comparable to a real-world problem.
BPM expert evaluation: Is it easy to configure/adapt the BP with the use of LAURA to solve the proposed problem through a BPM workflow? System expert evaluation: Is it easy to configure/adapt the application you developed with the use of LAURA to solve the proposed problem?	Evaluation of the perceived ease-of-use. According to Davis it can be defined as *"the degree to which a person believes that using a particular system would be free of effort"*. In this way, this question seeks to evaluate whether the APIs, provided by LAURA architecture, is easy to use and offer simple abstraction when used by the developer of the IoT final solution.
Was LAURA useful to solve the proposed problem?	Evaluation of the architecture usefulness. According to Davis, it can be defined as "*the degree to which a person believes that using a particular system would enhance his or her job performance*". In this way, this question seeks to evaluate whether the LAURA allows relatively easy integration between LAURA and the final application that the developer intends to use and that is capable of solving the proposed problem in the best way possible.
Assuming you have to use an IoT architecture to solve a problem, like to the one previously presented through a BP in BPMN, would you use LAURA architecture to solve it?	Evaluation of the intention of use. It is easy to use (configure or reconfigure) the BP to solve a problem using LAURA architecture.

Table 9
The qualitative question for each evaluation question.

Qualitative observations	Rationale
We will be grateful if you have any comments, suggestions, aspect or characteristic related to the above question. If you have nothing to say, simply write 'no comments' in the field below.	The evaluation performed by professionals who had never used LAURA is relevant and important to scientific research. The labor market professional has a critical and pragmatic view of a solution. The participant can provide rich and consistent feedback with insights from a different point of view than the people involved in developing the LAURA architecture. In this way, this qualitative open question can provide relevant information to the improvement of the solution.

or infrastructures, and who has never worked with LAURA architecture. Thus, the 'point of view' of the participants is related to the final application developer which focus is related to the higher-level aspects of the architecture.

The evaluation was divided into two cycles: pilot evaluation and regular evaluation. Pilot evaluations were performed to validate the whole cycle of the experiment. After these pilot tests, the experiment was released to the regular participants. After reaching the minimum number of desired participants (at least 15 of each group, totaling 30 participants), the results were tabulated and discussed.

The spreadsheet containing all raw data, tables and some graphics, without exposing confidential data, is available in repository ''experiments/Empirical-evaluation'[9] of LAURA project at GitHub.

For the remainder of this paper and in the spreadsheet containing all raw data, each participant, acting as a pilot test, is identified with '&Pilot-part-' string plus an ID number. Each participant of the regular evaluation is identified with '&Part-' string plus an ID number.

5.4.4. Results and discussion

The evaluation participants reside mostly in Brazil (97%), have an academic background, and 30% have a master's or doctorate degree. The predominant area is services (50%), followed by the academic area (43%). Most (73%) have basic knowledge of IoT, however, most (67%) do not have advanced IoT knowledge. Participants with the System Expert profile used JavaScript (27%), Java (13%) or Node.js (10%). The average age of the participants is 32.3 years and they have a large professional background, on average, 6.6 years. To perform the experiment itself (from Stage 3 of the form) both groups took, on average, 1.8 h.

These data reflect the professional profile of professionals who were contacted by LinkedIn, the university UFES or other institutions that attended the calls. After the contact with each participant, it was verified that all the evaluation takes approximately four hours to be made, because it takes an average of three hours to understand the problem and then, between one and two hours to perform the evaluation. Thus, the time required to perform the evaluation is an important aspect that may inhibit the participation of professionals with little experience. However, the participants' report, through direct contact, the analysis of the open questions and the verified result of the research questions, showed that the participants found it relatively easy to use LAURA to solve the problem proposed.

The text of the form and the proposed application scenario were carefully thought out and planned to propose a context similar to a real-world problem that might reflect the natural difficulties that occur when a professional needs to develop a solution to any problem. In this case, there are some aspects that are new to most participants: (i) working on the development of a situation-aware final application for a scenario involving IoT; (ii) use of tools or technologies that they are not familiar with; (iii) working with a health care setting that is not common in their work environment. Therefore, an evaluation that involves these aspects requires time for understanding the problem considered consistent with the time spent by the participants.

Table 10 presents the results of the evaluation questions by both groups. It was computed the average (Avg) and its respective Standard Deviation (SD) for each TAM factors from each group.

Table 10 shows that the average Likert rating scale of both groups for all TAM factors are between 4 - 'agree' and 5 'strongly agree'. The SD was low (between 10 and 14% for both groups),

[9] https://github.com/laura-architecture/experiments/tree/master/Empirical-evaluation.

Table 10
Results of evaluations for both groups of professionals.

TAM factor	For BPM expert		For System expert		Both groups	
	Avg.	SD	Avg.	SD	Avg.	SD
Ease-of-Use	4.47	0.64	4.33	0.62	4.40	0.62
Usefulness	4.60	0.51	4.73	0.46	4.67	0.48
Intention of use	4.60	0.51	4.47	0.64	4.53	0.57

indicating a uniformity in responses that are similar to the results of the pilot test. In addition, the results of the answers to the first evaluation question indicate that the participants consider the problem equivalent to the ones commonly found in the real world.

Thus, according to the participants' views, the LAURA architecture has scored well for the TAM factors. This indicates its suitability to developers to solve similar problems in other IoT scenarios. The good averages obtained in both groups indicate that it is straightforward to integrate LAURA into the professionals' development environment. Moreover, the good averages of *'Usefulness'* and '*Intention*' show a positive evaluation of the overall functionality offered by the architecture, particularly the facilities related to situation-awareness, suggesting that LAURA is useful to develop IoT situation-awareness applications in corporate environments.

5.4.5. Limitations of this evaluation

Carrying out a complete evaluation of an IoT architecture is not a simple task, since it involves the participation of various stakeholders working in several aspects of the solution, from the lowest to the highest level. In addition, an evaluation of this magnitude, involving the simultaneous participation of several professionals, would take longer to complete and sometimes the participants might not finish it. In any case, it would be interesting to evaluate the architecture from the point of view of the other LAURA stakeholders, for example, the IoT Expert and the Situation/Rule Expert. Their feedback and assessments in other levels of abstractions of the LAURA architecture could contribute to improving the proposed solution.

6. Conclusions

One of the central challenges addresses by LAURA is the capacity of implementing changes in business rules (in response to business demands), using high-level languages and approaches. LAURA addresses this challenge by providing a Situation/Rule layer, which employs a distinguishing situation platform to support the representation and management of situations in a more expressive and friendly way than traditional CEP approaches. In addition, at Application Layer, LAURA provides automatic node code generation from BPMN specifications, freeing the developer of business models from the task of knowing details of lower level programming aspects. Also, a web application allows high-level users to remote update the node code image using communication facilities provided by lower layers, particularly the Core Layer, speeding up the work and facilitating the interaction of high-level stakeholders with LAURA.

The ability to deal in a simpler manner with IoT real-world scenarios, composed of numerous contextual sources, was another major challenge, which has been addressed by LAURA mostly at the Physical and Core Layers. The solution relies on VM-based approaches – and Terra System in particular to make high-level programming resources and functions available in a uniform way, regardless of the hardware platform.

We have also identified the complex challenge of engaging a multidisciplinary team in the process of developing real-world IoT applications in business environments. LAURA solution for this challenge includes: (i) a "syntactic" view, represented by a layered architecture that provides a set of software components and well defined interaction points, which implement facilities and abstractions that are used by various professionals, with different roles and technical backgrounds; (ii) a "semantic" view, represented by LAURA reference model and the domain application model, together with the list of situations of interest. The LAURA reference model and the UML application model, in addition to the contextual situations specified, offer stakeholders a common ground of understanding of the essential concepts necessary to develop their final IoT solutions using LAURA.

The choice for a fully decoupled architectural design also helped to tackle the challenge of dealing with adaptive environments. This allows, for example, developers to choose technologies that are easier to adapt or integrate with their existing IT solutions.

LAURA Empirical Evaluation attested that our proposal obtained a very positive acceptance from the BPM Experts and the System Experts professionals, who evaluated LAURA architecture with grades between 4 - 'agree' and 5 - 'strongly agree' for the perception of 'Usefulness', and 'Intention of Use'. This suggests the suitability of LAURA as a support infrastructure for the development of situation-aware or business-aware final IoT applications.

Future work can occur on several levels and aspects, however, some issues deserve to be highlighted by the relevance in IoT or by the importance of the improvements that can be made in LAURA. Among them, we can point out: (i) sensor data filtering and aggregation based on QoC (Quality of Context) parameters, which would bring to LAURA greater quality of data awareness; (ii) development of a graphical language for modeling situations, which would provide the developer with a great readiness to specify situation types considering aspects such as composition of situations and their temporal reasoning; (iii) privacy and security support, relevant aspects of the IoT domain that were not exploited in this version of LAURA; (iv) improvements in the automatic node code generation solution to fully explore BPMN notation elements, and the inclusion of additional facilities, such as the creation of a plug-in that can be used in different BPMS.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Sergio Teixeira: Project administration, Writing - original draft, Writing - review & editing, Conceptualization, Methodology, Software, Investigation, Validation, Data curation. **Bruno Alves Agrizzi:** Software, Resources, Investigation, Data curation. **José Gonçalves Pereira Filho:** Writing - review & editing, Conceptualization, Supervision. **Silvana Rossetto:** Writing - review & editing. **Isaac Simões Araújo Pereira:** Software, Resources, Methodology, Conceptualization. **Patrícia Dockhorn Costa:** Writing - review & editing. **Adriano Francisco Branco:** Software, Resources, Writing - review & editing. **Ruan Rocha Martinelli:** Software, Resources.

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior– Brasil (CAPES) – Finance Code 001. A special thanks to Claire Henderson Venables and Geraldina Malbar Moscon, the enthusiastic language teachers, who revised the English language, to Vinicius Rossi Vacari who drew the images that visually represent our thoughts and ideas,

and to the ABPMP for the support in the dissemination of the invitations for the Empirical Evaluation.

References

(OMG), O.M.G., 2015. Business Process Modeling Notation (BPMN) [WWW Document]. http://www.omg.org/bpmn/ (Accessed 1 January 2015).

Aazam, M., Huh, E.N., 2014. Fog computing and smart gateway based communication for cloud of things. In: 2014 International Conference on Future Internet of Things and Cloud, pp. 464–470. doi:10.1109/FiCloud.2014.83.

ABPMP International, 2018. ABPMP International [WWW Document]. https://www.abpmp.org (Accessed 29 September 2018).

Acciona S.A., 2013. ACCIONA heads the make Sense R&D project to reduce Wireless Sensor Networks costs by 40% [WWW Document]. https://www.acciona.com/pressroom/news/2013/february/acciona-heads-the-makesense-rd-project-to-reduce-wireless-sensor-networks-costs-by-40/ (Accessed 22 June 2019).

Adi, A., Botzer, D., Etzion, O., 2000. Semantic Event Model and its Implication on Situation Detection. ECIS.

Agrizzi, B.A., 2018. Terra generation [WWW Document]. https://github.com/laura-architecture/terra-generation (Accessed 22 October 2018).

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E., 2002. A survey on sensor networks. IEEE Commun. Mag. 40, 102–105. doi:10.1109/MCOM.2002.1024422.

Al-Doghman, F., Chaczko, Z., Ajayan, A.R., Klempous, R., 2016. A review on Fog Computing technology. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1525–1530. doi:10.1109/SMC.2016.7844455.

Alexopoulos, K., Sipsas, K., Xanthakis, E., Makris, S., Mourtzis, D., 2018. An industrial Internet of things based platform for context-aware information services in manufacturing. Int. J. Comput. Integr. Manuf. 31, 1111–1123. doi:10.1080/0951192X.2018.1500716.

Alfayez, F., Hammoudeh, M., Abuarqoub, A., 2015. A survey on MAC protocols for duty-cycled wireless sensor networks. Procedia Comput. Sci. 73, 482–489. doi:10.1016/j.procs.2015.12.034.

Aliverti, E., De Maio, M., Salatino, M., 2016. Mastering Jboss Drools 6. Packt Publishing.

Almeida, J., 2006. Model-Driven Design of Distributed Applications. Springer.

Arduino [WWW Document], 2016. https://www.arduino.cc/ (Accessed 1 January 2016).

Avilés-López, E., García-Macías, J.A., 2009. TinySOA: a service-oriented architecture for wireless sensor networks. Serv. Oriented Comput. Appl. 3, 99–108. doi:10.1007/s11761-009-0043-x.

Ayala, I., Amor, M., Fuentes, L., Troya, J.M., 2015. A software product line process to develop agents for the IoT. Sensors 15, 15640–15660. doi:10.3390/s150715640.

Bai, Y., Ji, H., Han, Q., Huang, J., Qian, D., 2007. MidCASE: a service oriented middleware enabling context awareness for smart environment. In: 2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07), pp. 946–951. doi:10.1109/MUE.2007.152.

Balani, R., Han, C.-C., Rengaswamy, R.K., Tsigkogiannis, I., Srivastava, M., 2006. Multi-level software reconfiguration for sensor networks. In: Proc. 6th ACM IEEE Int. Conf. Embed. Softw, pp. 112–121. doi:10.1145/1176887.1176904.

Baldam, R., Valle, R., Rozenfeld, H., 2014. Gerenciamento de Processos de Negócios – BPM. Elsevier, Brasil, Rio de Janeiro.

Bali, M., 2009. Drools JBoss Rules 5.0 Developer's Guide. Packt Publishing Ltd., Birmingham, UK B27 6PA.

Baronti, P., Pillai, P., Chook, V.W.C., Chessa, S., Gotta, A., Hu, Y.F., 2007. Wireless sensor networks: a survey on the state of the art and the 802.15.4 and ZigBee standards. Comput. Commun. 30, 1655–1695. doi:10.1016/j.comcom.2006.12.020.

Bassi, A., Bauer, M., Fiedler, M., Kramp, T., van Kranenburg, R., Lange, S., Meissner, S., 2013. Enabling things to talk: designing IoT solutions with the IoT architectural reference model. In: Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model doi:10.1007/978-3-642-40403-0.

Bauer, M., Boussard, M., Bui, N., Carrez, F., Jardak, C., De Loof, J., Magerkurth, C., Meissner, S., Nettsträter, A., Olivereau, A., Thoma, M., Walewski, J.W., Stefa, J., Salinas, A., 2013. Deliverable D1.5 – final architectural reference model for the IoT v3.0, Internet of Things – architecture (IOT-A).

Bermudez-Edo, M., Elsaleh, T., Barnaghi, P., Taylor, K., 2017. IoT-Lite: a lightweight semantic model for the internet of things and its use with dynamic semantics. Pers. Ubiquitous Comput. 21, 475–487. doi:10.1007/s00779-017-1010-8.

Bizagi, 2018. Bizagi [WWW Document]. https://www.bizagi.com/ (Accessed 12 October 2018).

Bonomi, F., Milito, R., Zhu, J., Addepalli, S., 2012. Fog computing and its role in the Internet of Things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC '12. ACM, New York, NY, USA, pp. 13–16. doi:10.1145/2342509.2342513.

Branco, A., Sant'anna, F., Ierusalimschy, R., Rodriguez, N., Rossetto, S., 2015. Terra: flexibility and safety in wireless sensor networks. ACM Trans. Sens. Netw. 11. doi:10.1145/2811267, 59:1–59:27.

Branco, A.F., 2017. Terra IoT System [WWW Document]. http://afbranco.github.io/Terra (Accessed 16 February 2018).

Brosh, E., Baset, S.A., Misra, V., Rubenstein, D., Schulzrinne, H., 2010. The delay-friendliness of TCP for real-time traffic. IEEE/ACM Trans. Netw. 18, 1478–1491. doi:10.1109/TNET.2010.2050780.

Buchholz, T., Küpper, A., Schiffers, M., 2003. Quality of context: what it is and why we need it. In: Proc. Work. HP OpenView Univ. Assoc., pp. 1–14 10.1.1.147.565.

Butler Consortium, 2013. D3.2 Integrated System Architecture and Initial Pervasive BUTLER proof of concept.

Chen, R.-Y., 2017. An intelligent value stream-based approach to collaboration of food traceability cyber physical system by fog computing. Food Control 71, 124–136. doi:10.1016/j.foodcont.2016.06.042.

Chen, Y.C., Chang, Y.C., Chen, C.H., Lin, Y.S., Chen, J.L., Chang, Y.Y., 2017. Cloud-fog computing for information-centric Internet-of-Things applications. In: 2017 International Conference on Applied System Innovation (ICASI), pp. 637–640. doi:10.1109/ICASI.2017.7988506.

Choi, H., Rhee, W., 2012. Distributed semantic sensor web architecture. In: TENCON 2012 IEEE Region 10 Conference, pp. 1–6. doi:10.1109/TENCON.2012.6412239.

Cisco, 2015. Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are San Jose, CA, USA.

Compton, M., Barnaghi, P., Bermudez, L., García-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W.D., Phuoc, D.Le, Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., Taylor, K., 2012. The SSN ontology of the W3C semantic sensor network incubator group. Web Semant. Sci. Serv. Agents World Wide Web 17, 25–32. doi:10.1016/j.websem.2012.05.003.

Corredor, I., Martínez, J.F., Familiar, M.S., 2011. Bringing pervasive embedded networks to the service cloud: a lightweight middleware approach. J. Syst. Archit. 57, 916–933. doi:10.1016/j.sysarc.2011.04.005.

Costa, P.D., Almeida, J.P.A., Pereira, I.S.A., van Sinderen, M., Pires, L.F., 2016. Rule-based support for situation management. In: Rogova, G., Scott, P. (Eds.), Fusion Methodologies in Crisis Management: Higher Level Fusion and Decision Making. Springer International Publishing, Cham, pp. 341–364. doi:10.1007/978-3-319-22527-2_16.

Cubo, J., Nieto, A., Pimentel, E., 2014. A cloud-based internet of things platform for ambient assisted living. Sensors 14, 14070–14105. doi:10.3390/s140814070.

Da, K., Roose, P., Dalmau, M., Nevado, J., Karchoud, R., 2014. Kali2Much: a context middleware for autonomic adaptation-driven platform. In: Proceedings of the 1st ACM Workshop on Middleware for Context-Aware Applications in the IoT, M4IOT '14. ACM, New York, NY, USA, pp. 25–30. doi:10.1145/2676743.2676748.

De Maio, M.N., Salatino, M., Aliverti, E., 2014. jBPM6 Developer Guide, Community Experience Distilled. Packt Publishing.

Delsing, J., Eliasson, J., Leijon, V., 2010. Latency and packet loss of an interferred 802.15.4 channel in an industrial environment. In: Proc. - 4th Int. Conf. Sens. Technol. Appl. SENSORCOMM, 2010, pp. 33–38. doi:10.1109/SENSORCOMM.2010.12.

Dener, M., Bay, O.F., 2016. TeenySec: a new data link layer security protocol for WSNs. Secur. Commun. Netw. 9, 5882–5891. doi:10.1002/sec.1743.

Dey, A.K., 2001. Understanding and using context. Pers. Ubiquitous Comput. 5, 4–7. doi:10.1007/s007790170019.

Dockhorn Costa, P., 2007. Architectural Support for Context-Aware Applications: From Context Models to Services Platforms. Universal Press.

Doddapaneni, K., Ever, E., Gemikonakli, O., Malavolta, I., Mostarda, L., Muccini, H., 2012. A model-driven engineering framework for architecting and analysing Wireless Sensor Networks. In: 2012 Third Int. Work. Softw. Eng. Sens. Netw. Appl., pp. 1–7. doi:10.1109/SESENA.2012.6225729.

Dong, W., Chen, C., Bu, J., Liu, W., Dunkels, A., Finne, N., Eriksson, J., Voigt, T., Eicken, T.Von, Culler, D.E., Schauser, K.E., 2014. Optimizing relocatable code for efficient software update in networked embedded systems. ACM Trans. Sens. Netw. 11, 1–34. doi:10.1145/2629479.

Eicken, T.Von, Culler, D.E., Goldstein, S.C., Schauser, K.K., 1992. Active messages: a mechanism for integrated communication and computation. In: Proceedings of the 19th International Symposium on Computer Architecture, pp. 1–20. doi:10.1109/ISCA.1992.753322.

El-Mougy, A., Al-Shiab, I., Ibnkahla, M., 2019. Scalable personalized IoT networks. Proc. IEEE 107, 695–710. doi:10.1109/JPROC.2019.2894515.

Endsley, M.R., 1995. Toward a theory of situation awareness in dynamic systems. Hum. Factors J. Hum. Factors Ergon. Soc. 37, 32–64. doi:10.1518/001872095779049543.

EsperTech Inc., 2017. Complex event processing streaming analytics [WWW Document]. http://www.espertech.com/.

Expressjs, 2017. Express framework for Node.js [WWW Document]. https://expressjs.com/ (Accessed 27 November 2017).

Flasiński, M., 2016. Introduction to Artificial Intelligence. Springer, Cham, Switzerland doi:10.1007/978-3-319-40022-8.

Flouris, I., Giatrakos, N., Deligiannakis, A., Garofalakis, M., Kamp, M., Mock, M., 2017. Issues in complex event processing: status and prospects in the Big Data era. J. Syst. Softw. 127, 217–236. doi:10.1016/j.jss.2016.06.011.

Fonseca, J., Ferraz, C., Gama, K., 2016. A policy-based coordination architecture for distributed complex event processing in the Internet of Things: doctoral symposium. In: Proceedings of the 10th ACM International Conference on Distributed and Event-Based Systems, DEBS '16. ACM, New York, NY, USA, pp. 418–421. doi:10.1145/2933267.2933431.

Forsström, S., Kanter, T., 2014. Enabling ubiquitous sensor-assisted applications on the internet-of-things. Pers. Ubiquitous Comput. 18, 977–986. doi:10.1007/s00779-013-0712-9.

Frömel, B., Kopetz, H., 2016. Interfaces in evolving cyber-physical systems-of-systems. In: Bondavalli, A., Bouchenak, S., Kopetz, H. (Eds.), Cyber-Physical Systems of Systems: Foundations – A Conceptual Model and Some Derivations: The AMADEOS Legacy. Springer International Publishing, Cham, pp. 40–72. doi:10.1007/978-3-319-47590-5_2.

Fülöp, L.J., Beszédes, Á., Tóth, G., Demeter, H., Vidács, L., Farkas, L., 2012. Predictive complex event processing: a conceptual framework for combining complex event processing and predictive analytics. In: Proceedings of the Fifth

Balkan Conference in Informatics, BCI '12. ACM, New York, NY, USA, pp. 26–31. doi:10.1145/2371316.2371323.
Gartner, 2017. Gartner IT Glossary [WWW Document]. http://www.gartner.com/it-glossary/complex-event-processing (Accessed 16 July 2017).
Gartner, I., 2017. Business Process Management Suites (BPMSs) [WWW Document]. https://www.gartner.com/it-glossary/bpms-business-process-management-suite (Accessed 17 October 2017).
Gray, P., Salber, D., 2001. Modelling and using sensed context information in the design of interactive applications. In: LNCS 2254 Proc. 8th IFIP Int. Conf. Eng. Human-Computer Interact, pp. 317–335 2254/2001.
Gyrard, A., Datta, S.K., Bonnet, C., Boudaoud, K., 2015. A semantic engine for internet of things: cloud, mobile devices and gateways. In: 2015 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 336–341. doi:10.1109/IMIS.2015.83.
Happ, D., Karowski, N., Menzel, T., Handziski, V., Wolisz, A., 2017. Meeting IoT platform requirements with open pub/sub solutions. Ann. Telecommun. 72, 41–52. doi:10.1007/s12243-016-0537-4.
Harpe, S.E., 2015. How to analyze Likert and other rating scale data. Curr. Pharm. Teach. Learn. doi:10.1016/j.cptl.2015.08.001.
Hilal, A.R., Basir, O.A., 2015. A scalable sensor management architecture using BDI model for pervasive surveillance. IEEE Syst. J. 9, 529–541. doi:10.1109/JSYST.2014.2334071.
Hoffman, J.S., 2016. Um Serviço Para Controle e Avaliação de Informações de Redes de Sensores Sem Fio na Internet. Federal University of Espirito Santo.
Hu, P., Dhelim, S., Ning, H., Qiu, T., 2017. Survey on fog computing: architecture, key technologies, applications and open issues. J. Netw. Comput. Appl. 98, 27–42. doi:10.1016/j.jnca.2017.09.002.
Hui, J.W., Culler, D., 2004. The dynamic behavior of a data dissemination protocol for network programming at scale. In: Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems, SenSys '04. ACM, New York, NY, USA, pp. 81–94. doi:10.1145/1031495.1031506.
IDC, 2015. Explosive Internet of Things spending to reach $1.7 trillion in 2020 [WWW Document]. Explos. Internet Things Spend. to Reach $1.7 Trillion 2020. https://www.businesswire.com/news/home/20150602005329/en/Explosive-Internet-Things-Spending-Reach-1.7-Trillion (Accessed 1 Januray 2015).
IDC, 2017. Internet of Things spending forecast to grow 17.9% in 2016 led by manufacturing, transportation, and utilities investments, according to new IDC spending guide [WWW Document]. http://www.idc.com/getdoc.jsp?containerId=prUS42209117 (Accessed 4 Febrauary 2017).
IHME, 2016. Global spending on health is expected to increase to $18.28 trillion worldwide by 2040 but many countries will miss important health benchmarks [WWW Document]. http://www.healthdata.org/news-release/global-spending-health-expected-increase-1828-trillion-worldwide-2040-many-countries (Accessed 23 June 2019).
Integrated, M., 2018. DS18B20 [WWW Document]. https://www.maximintegrated.com/en/products/sensors/DS18B20.html (Accessed 10 July 2018).
International, E., 2013. The JSON data interchange format [WWW Document]. http://www.json.org/ (Accessed 27 February 2017).
ITU-T, I.T.U.-, 2014. Recommendation ITU-T F.746.1 – Requirements for low-latency interactive multimedia streaming.
Jantunen, I., Laine, H., Huuskonen, P., Trossen, D., Ermolov, V., 2008. Smart sensor architecture for mobile-terminal-centric ambient intelligence. Sens. Actuators A Phys. 142, 352–360. doi:10.1016/j.sna.2007.04.014.
Jara, A.J., Lopez, P., Fernandez, D., Castillo, J.F., Zamora, M.A., Skarmeta, A.F., 2014. Mobile digcovery: discovering and interacting with the world through the Internet of things. Pers. Ubiquitous Comput. 18, 323–338. doi:10.1007/s00779-013-0648-0.
Juristo, N., Moreno, A.M., 2010. Basics of Software Engineering Experimentation, first ed. Springer Publishing Company, Incorporated.
Karlof, C., Sastry, N., Wagner, D., 2004. TinySec: a link layer security architecture for wireless sensor networks. In: Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems, SenSys '04. ACM, New York, NY, USA, pp. 162–175. doi:10.1145/1031495.1031515.
Kingatua, A., 2017. IoT in HVAC systems [WWW Document]. https://electronicsandict.com/iot-in-hvac-systems/ (Accessed 27 August 2019).
Kuo, S.-P., Lin, C., Lee, Y.-F., Fang, H.-W., Hong, Y.-W.P., Lin, H.-C., Tseng, Y.-C., King, C.-T., Wang, C.-L., 2008. The NTP experimental platform for heterogeneous Wireless Sensor Networks. In: Proceedings of the 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities, TridentCom '08. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium 35:1–35:6.
Leach, P., Mealling, M., Salz, R., 2005. A Universally Unique IDentifier (UUID) URN namespace [WWW Document]. Netw. Work. Gr. https://www.ietf.org/rfc/rfc4122.txt (Accessed 28 July 2019).
Levis, P., 2012. Experiences from a decade of TinyOS development. In: Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation, OSDI'12. USENIX Association, Berkeley, CA, USA, pp. 207–220.
Levis, P., Gay, D., Culler, D., 2005. Active sensor networks. In: Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation – Volume 2, NSDI'05. USENIX Association, Berkeley, CA, USA, pp. 343–356.
Luk, M., Mezzour, G., Perrig, A., Gligor, V., 2007. MiniSec: a secure sensor network communication architecture. In: 2007 6th International Symposium on Information Processing in Sensor Networks, pp. 479–488. doi:10.1109/IPSN.2007.4379708.
Mandjes, M., van der Wal, K., Kooij, R., Bastiaansen, H., 1999. End-to-end delay models for interactive services on a large-scale IP network. In: Proceedings of the 7th Workshop on Performance Modelling and Evaluation of ATM & IP Networks (IFIP99), pp. 28–30.
Marín-Tordera, E., Masip-Bruin, X., García-Almiñana, J., Jukan, A., Ren, G.-J., Zhu, J., 2017. Do we all really know what a fog node is? Current trends towards an open definition. Comput. Commun. 109, 117–130. doi:10.1016/j.comcom.2017.05.013.
Martinelli, R.R., 2017. Terra Gateway e Terra Core: Uma solução de suporte ao desenvolvimento de aplicações finais para Redes de Sensores sem fio. Federal University of Espirito Santo.
Matthews, K., 2018. 6 exciting IoT use cases in healthcare [WWW Document]. https://www.iotforall.com/exciting-iot-use-cases-in-healthcare (Accessed 27 August 2019).
Mellor, S., Scott, K., Uhl, A., Weise, D., 2004. MDA distilled: principles of model-driven architecture. Addison-Wesley Prof.
MEMSIC, 2016. MEMSIC powerful sensing solutions – Wireless Sensor Networks [WWW Document]. http://www.memsic.com/wireless-sensor-networks/ (Accessed 1 January 2016).
Microsoft, 2017. Microsoft StreamInsight [WWW Document]. Microsoft. https://www.microsoft.com/en-us/download/details.aspx?id=30149.
murabet, A.El, Abtoy, A., Touhafi, A., Tahiri, A., 2018. Ambient Assisted living system's models and architectures: a survey of the state of the art. J. King Saud Univ. – Comput. Inf. Sci. doi:10.1016/j.jksuci.2018.04.009.
Murray, G., Schaub, K., Vancil, R., Leclair, A., 2016. IDC FutureScape IDC FutureScape: worldwide chief marketing officer advisory 2016 predictions 1–18.
MySQL, 2017. MySQL [WWW Document]. Oracle Corp. https://www.mysql.com/ (Accessed 27 November 2017).
Nations, U., 2019. Growing at a slower pace, world population is expected to reach 9.7 billion in 2050 and could peak at nearly 11 billion around 2100 [WWW Document]. https://www.un.org/development/desa/en/news/population/world-population-prospects-2019.html#targetText=The world's population is expected, United Nations report launched today. (Accessed 29 August 2019).
Oxford Economics and Cisco, 2017. The A.I. Paradox How Robots Will Make Work More Human San Jose, CA, USA.
Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F., 2008. Service-oriented computing: a research roadmap. Int. J. Coop. Inf. Syst. 17, 223–255. doi:10.1142/S0218843008001816.
Paphitou, A.C., Constantinou, S., Kapitsaki, G.M., 2015. SensoMan: remote management of context sensors. In: Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, WIMS '15. ACM, New York, NY, USA doi:10.1145/2797115.2797121, 19:1–19:6.
Pereira, I.S.A., Costa, P.D., Almeida, J.P.A., 2013. A rule-based platform for situation management. In: 2013 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), pp. 83–90. doi:10.1109/CogSIMA.2013.6523827.
Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E., 2002. SPINS: security protocols for sensor networks. Wirel. Netw. 8, 521–534. doi:10.1023/A:1016598314198.
Petlund, A., 2009. Improving Latency for Interactive, Thin-Stream Applications Over Reliable Transport. University of Oslo.
Pietro, R.Di, Guarino, S., Verde, N.V, Domingo-Ferrer, J., 2014. Security in wireless ad-hoc networks – a survey. Comput. Commun. 51, 1–20. doi:10.1016/j.comcom.2014.06.003.
Plummer, D.C., Baker, V.L., Austin, T., Smulders, C., Tully, I., Valdes, R., Sarner, A., Moyer, K.R., Karamouzis, F., Andrews, W., Heiser, J., Fabre, S., McIntyre, A., Scheibenreif, D., Hobert, K.A., Sussin, J., Marshall, R., Smith, R., Kihn, M., Revang, M., Leow, A., Wong, J., Hicks, T., Morello, D., Furlonger, D., Brant, K.F., Poitevin, H., 2015. Top Strategic Predictions for 2016 and Beyond: The Future Is a Digital Thing [WWW Document]. Gartner https://www.gartner.com/binaries/content/assets/events/keywords/symposium/sym26/gartner_top_strategic_predictions_2016.pdf.
Rahmani, A.M., Gia, T.N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., Liljeberg, P., 2018. Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. Futur. Gener. Comput. Syst. 78, 641–658. doi:10.1016/j.future.2017.02.014.
Redhat, 2015. jBPM [WWW Document]. Red Hat Inc. http://www.jbpm.org/ (Accessed 1 February 2016).
RedHat, 2017. Drools [WWW Document]. http://www.drools.org/ (Accessed 9 March 2017).
Rescorla, E., 2000. RFC 2818 – HTTP Over TLS. In: Network Working Group, IETF. http://tools.ietf.org/html/rfc2818.
Ribeiro, F.S., 2016. Terra DIA: Uma solução de suporte a atualização dinâmica de bytecodes em redes de sensores sem fio baseadas em maquina virtual. Federal University of Espirito Santo.
Rodrigues, T., Delicato, F.C., Batista, T., Pires, P.F., Pirmez, L., 2015. An approach based on the domain perspective to develop WSAN applications. Softw. Syst. Model. 1–29. doi:10.1007/s10270-015-0498-5.
Sánchez López, T., Ranasinghe, D.C., Harrison, M., McFarlane, D., 2012. Adding sense to the Internet of Things. Pers. Ubiquitous Comput. 16, 291–308. doi:10.1007/s00779-011-0399-8.
SantAnna, F., Rodriguez, N., Ierusalimschy, R., Landsiedel, O., Tsigas, P., 2013. Safe system-level concurrency on resource-constrained nodes. In: Proc. 11th ACM Conf. Embed. Networked Sens. Syst. – SenSys, 13, pp. 1–14. doi:10.1145/2517351.2517360.

Sha, K., Wei, W., Yang, T.A., Wang, Z., Shi, W., 2018. On security challenges and open issues in Internet of Things. Futur. Gener. Comput. Syst. 83, 326–337. doi:10.1016/j.future.2018.01.059.
Shu, Z., Wan, J., Zhang, D., Li, D., 2016. Cloud-integrated cyber-physical systems for complex industrial applications. Mob. Netw. Appl. 21, 865–878. doi:10.1007/s11036-015-0664-6.
Silvestre, B.O., 2017. TOSSAM – TinyOS Serial AM for Lua [WWW Document]. http://www.inf.ufg.br/~brunoos/tossam/ (Accessed 14 July 2017).
Socket.io, 2017. Socket.io [WWW Document]. Socket.io community. https://socket.io/ (Accessed 27 November 2017).
Stamford, C., 2016. Gartner says by 2020, more than half of major new business processes and systems will incorporate some element of the Internet of Things [WWW Document]. Gartner.com. https://www.gartner.com/en/newsroom/press-releases/2016-01-14-gartner-says-by-2020-more-than-half-of-major-new-business-processes-and-systems-will-incorporate-some-element-of-the-internet-of-things (Accessed 29 July 2015).
Su, K., Li, J., Fu, H., 2011. Smart city and the applications. In: 2011 Int. Conf. Electron. Commun. Control. ICECC 2011 – Proc, pp. 1028–1031. doi:10.1109/ICECC.2011.6066743.
Teixeira, S., 2018a. LAURA – empirical evaluation form for the System expert [WWW Document]. https://docs.google.com/forms/d/e/1FAIpQLSefkbpcYrwbocb9VLY3oc44WOBoL4VjneyJpvLmWfJmVJ_iGQ/viewform (Accessed 23 January 2019).
Teixeira, S., 2018b. LAURA – empirical evaluation form for the BPM expert [WWW Document]. https://docs.google.com/forms/d/e/1FAIpQLScVWkkV-fED2df9mbjweV6J6dqXFP6ZuAo2yaU03o7ueqlu-A/viewform (Accessed 23 January 2019).
Teixeira, S., Agrizzi, B.A., Martinelli, R.R., Branco, A.F., Pereira, I.S.A., Al., E., 2017a. LAURA architecture: codes, experiments, documentation, videos and others related files [WWW Document]. https://laura-architecture.github.io (Accessed 1 July 2017).
Teixeira, S., Agrizzi, B.A.A., Filho, J.G.P.G.P., Rossetto, S., Baldam, R.D.L.L., 2017b. Modeling and automatic code generation for Wireless Sensor Network applications using model-driven or business process approaches: a systematic mapping study. J. Syst. Softw. 132, 50–71. doi:10.1016/j.jss.2017.06.024.
Toninelli, A., Corradi, A., Montanari, R., 2009. A quality of context-aware approach to access control in pervasive environments. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, pp. 236–251. doi:10.1007/978-3-642-01802-2_18.
Townsend, M., 2019. IDC TechScape: worldwide life science commercial IoT technologies, abstract [WWW Document]. IDC Res. https://www.idc.com/getdoc.jsp?containerId=US45061619 (Accessed 23 June 2019).
Tranquillini, S., Spiess, P., Daniel, F., Karnouskos, S., Casati, F., Oertel, N., Mottola, L., Oppermann, F.J., Picco, G.P., Roemer, K., Voigt, T., 2012. Process-based design and integration of wireless sensor network applications. Bus. Process Manag. BPM 2012 7481, 134–149. doi:10.1007/978-3-642-32885-5_10.
Venkatesh, V., Davis, F.D., 1996. A model of the antecedents of perceived ease of use: development and test. Decis. Sci. doi:10.1111/j.1540-5915.1996.tb01822.x.
Weiser, M., 1999. The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev. 3, 3–11. doi:10.1145/329124.329126.
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2012. Experimentation in software engineering. Springer, New York doi:10.1007/978-3-642-29044-2.
Xi, L., Yuzhi, Z., 2014. Efficient innovative teaching scheme of Internet of Things based on practice. In: Zhong, S. (Ed.), Proceedings of the 2012 International Conference on Cybernetics and Informatics. Springer New York, New York, NY, pp. 493–497. doi:10.1007/978-1-4614-3872-4_63.
Ye, J., Dobson, S.A., McKeever, S., 2012. Situation identification techniques in pervasive computing: a review. Pervasive Mob. Comput. 8, 36–66. doi:10.1016/j.pmcj.2011.01.004.
Yi, S., Li, C., Li, Q., 2015. A survey of fog computing: concepts, applications and issues. In: Proceedings of the 2015 Workshop on Mobile Big Data, Mobidata '15. ACM, New York, NY, USA, pp. 37–42. doi:10.1145/2757384.2757397.
Yick, J., Mukherjee, B., Ghosal, D., 2008. Wireless sensor network survey. Comput. Netw. 52, 2292–2330. doi:10.1016/j.comnet.2008.04.002.
Yu, J., Bang, H.-C., Lee, H., Lee, Y.S., 2016. Adaptive Internet of Things and Web of Things convergence platform for Internet of reality services. J. Supercomput. 72, 84–102. doi:10.1007/s11227-015-1489-6.
ZeroMQ, 2017. ZeroMQ [WWW Document]. iMatix Corp. http://zeromq.org/ (Accessed 27 November 2017).
Zirpins, C., 2016. Situational data-analytics for the web-of-things. In: Proceedings of the 1st International Workshop on Mashups of Things and APIs, MOTA '16. ACM, New York, NY, USA 6:1–6:4. doi:10.1145/3007203.3007218.

Sergio Teixeira received his PhD in Computer Science from Federal University of Espírito Santo (UFES), in Vitória, Brazil. He received a post-graduate degree in Computer Networks in 1997 and went on to complete a M.Sc. in 2005, also at UFES. His research interests include computer networks, wireless sensor networks, Situation-aware and Business-aware computing and the Internet of Things. He has 18 years' experience in academia and 33 years in Computer Science, working mainly in Networks and Enterprise Systems. He is a lecturer at the Catholic University Center of Vitória (UCV).

Bruno Alves Agrizzi is a Computer Science M.Sc. candidate at Federal University of Espírito Santo (UFES), in Vitória, Brazil. He received his B.Sc. degree in Computer Science from UFES in 2016. His research interests include computer networks, wireless sensor networks, situation-aware, business-aware computing and Internet of Things.

José Gonçalves Pereira Filho is a Full Professor at Federal University of Espírito Santo (UFES), in Vitória, Brazil. He received his D.Sc. in Electrical Engineering from University of São Paulo (USP) in 1996. Part of the doctoral research was developed at the University of British Columbia, in Vancouver, Canada, in multimedia systems. He held a postdoctoral position at the Faculty of Electrical Engineering, Mathematics and Computer Science (EWI) of the University of Twente, in Enschede, The Netherlands, from Oct-2002 to Sep-2003 and from Sep-2014 to Aug-2015 (SCS Research Group). His research interests include computer networks, wireless sensor networks, context-aware computing and Internet of Things.

Silvana Rossetto received her PhD in Computer Science from Catholic University of Rio de Janeiro (PUC-Rio) in 2006. She holds the position of Associate Professor at the Computer Science Department of the Federal University of Rio de Janeiro (UFRJ), in Rio de Janeiro, Brazil, where she teaches for undergraduate courses and works as a researcher. Her current areas of interest are distributed and parallel systems and wireless sensor networks.

Isaac Simões Araújo Pereira is a Computer Science M.Sc. candidate at Federal University of Espírito Santo (UFES), in Vitória, Brazil. He received his B.Sc. degree in Computer Science from UFES in 2012. His research interests include context-aware applications and situation management.

Patrícia Dockhorn Costa is an Associate Professor at the Department of Computer Science of the Federal University of Espírito Santo, Brazil. She received her MSc and PhD degrees from the University of Twente, the Netherlands. Her research interests include the development of situation models, situation specification languages and implementation platforms. She investigates model-driven approaches to allow realization of situation detection from high-level situation specifications. Her team has been working on a framework to support the development of Situation-Aware applications, which includes (i) a Situation Modeling Language (SML) to allow graphic expression of primitive situations and complex situations involving the composition of situations; and (ii) a distributed implementation platform to allow rule-based situation specification and situation lifecycle management. She has coauthored more than 30 peer-reviewed papers on context- and situation-awareness and served as reviewer for numerous scientific conferences and journals.

Adriano Francisco Branco received his PhD in Computer Science from Catholic University of Rio de Janeiro (PUC-Rio) in 2015. His research focus on Wireless Sensor Network (WSN) and Internet of Things (IoT) in distributed system area. He also got his master in computer science at PUC-Rio in 2011 working with WSN. He undergraduates in Electronic Engineering at CEFET/RJ in 1992. From the undergraduate course he worked as electronic engineer and system developer at CBPF/CNPq (Brazil) and CERN (Switzerland). At CPBF, in the LAFEX laboratory, he worked on the parallel computer program from Fermilab collaboration group. At CERN he spent two years working in the New Trigger Project for LEP Delphi Experiment. After that he had worked more than 12 years in system integration consulting projects (as developer and project manager) for large companies. Mainly for the Industrial Automation and Telecommunications industries, including an international project in Manila/Philippines.

Ruan Rocha Martinelli is a Computer Science Bachelor at Federal University of Espírito Santo (UFES), in Vitória, Brazil. He received his B.Sc. degree in Computer Science from UFES in 2017. His research interests include wireless sensor networks, distributed systems, systems architecture and Internet of Things.