Future Generation Computer Systems 94 (2019) 250–271

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A systematic method for building Internet of Agents applications based on the Linked Open Data approach

Pablo Pico-Valencia [a,*], Juan A. Holgado-Terriza [b], Patricia Paderewski [b]

[a] Programming and Development of Software, Pontifical Catholic University of Ecuador, Esmeraldas, Ecuador
[b] Software Engineering Department, University of Granada, Granada, Spain

HIGHLIGHTS

- A novel systematic method to build Internet of Agents ecosystems is proposed.
- Our method allows building smart, proactive, collaborative and adaptable agents.
- Our reactive agent can model smart behaviors at microscopic and macroscopic levels.
- Semantic descriptors contribute to perform reasoning for agent dynamic discovery.
- Our methods describe agent ecosystems using contracts represented with linked data.

ARTICLE INFO

Article history:
Received 19 September 2017
Received in revised form 3 June 2018
Accepted 26 November 2018
Available online 30 November 2018

Keywords:
Internet of Things
Linked Open Data
Semantic agent contract
Software agile methodology
Agent-based method

ABSTRACT

The Internet of Agents (IoA) is an emerging paradigm whose objective is to mitigate the deficiencies of devices of Internet of Things (IoT) in terms of reasoning and social capabilities, in order to improve proactivity, intelligence and interoperability. This paper presents the guidelines to develop IoA applications based on described semantic agents following the Linked Open Data (LOD) approach and the specifications of the IoA-OWL ontology—a specialized full ontology that formally defines the main aspects related to a novel approach such as IoA. These guidelines have been drawn up via a systematic method created from the best practices of Agile Software Development Methodologies. This method creates smart, autonomous, collaborative IoA applications based on novel Linked Open Agents (LOAs) that are driven by Linked-Agent Contracts (LACs) and Workflows for Agent Control (WACs). From a practical perspective, our method separates the modeling of components using two levels, microscopic (at agent level) and macroscopic (at agent society level) facilitated the planning, configuring and implementation of each agent in the IoA ecosystem. Moreover, the method facilitates the agent creation automation process, reducing the time required for its development and simplifying the design complexity. These achievements were demonstrated through the modeling of an Ambient Intelligence (AmI) scenario on an office composed by a set of collaborative agents in order to provide smart comfort.

1. Introduction

The Internet of Things (IoT) is a booming paradigm that arose at end of the nineties in order to control the physical world by the use of a global network of smart objects which are interconnected through the Internet [1–3]. With the aim of covering a wide spectrum of real applications, to date a variety of smart objects have proliferated based on different standards. However, when IoT networks are integrated, some operational and communication conflicts emerge. In addition, the limitations of computation resources of some objects obstruct the dynamic and intelligent management of an IoT network [4]. In consequence, researchers have recently been interested in models and applications based on the integration of agent technologies and IoT infrastructures [5–11].

As a consequence of the integration process among software agents with the IoT technologies, a new approach related to the Future Internet, called Internet of Agents (IoA), has emerged [12]. This approach is basically oriented towards adding a component of intelligence, autonomy, collaboration and easily adaptation to objects connected in IoT networks [12,13]. Thus, these objects become "smart" and can operate coherently in complex and dynamic scenarios as demanded by the IoT [14,15].

An important contribution to this line of research is undoubtedly the *Agent of Thing* concept proposed by Mzahm et al. [15], who suggest that each object connected to an IoT network has

* Corresponding author.
E-mail addresses: ppico@pucese.edu.ec (P. Pico-Valencia), jholgado@ugr.es (J.A. Holgado-Terriza), patricia@ugr.es (P. Paderewski).

https://doi.org/10.1016/j.future.2018.11.042
0167-739X/

a special software agent embedded in order to add intelligence capabilities. From this idea other related approaches have been proposed, such as agent-based smart objects [6,16], IoT agent-based models [7,9,17], and Multi-Agent Systems (MASs) for IoT [10, 18]. However, most of these proposals do not define a formal methodology or method for building these types of applications as Software Engineering recommends.

In the Agent Oriented Engineering of Software (AOSE) field some tools which cover issues related to the analysis, design, development, implementation, testing, and documentation of MASs have been proposed. There are several Agent Oriented Methodologies (AOM), including MAS-CommonKads, Gaia, Tropos, Promethe -us, Message, Ingenias, Desire, or MaSE [19–21]. Nevertheless, most of these only support specific domains and, as a result, current AOMs do not consider important issues that the IoA applications must model in a bundled fashion, such as context, social circles and collaboration, intelligence, service orientation, and management of the resources associated with the IoT-objects. Based on these limitations, and taking into account the assertion of Tran and Low [19] that the "best" methodology depends on the target application, it is essential to have a structured and systematic method which guides the development process of systems in the IoA domain; especially if a new approach has been adopted, as is the case in this proposal.

The main contributions that this paper makes to the literature in this field are the following: (i) introducing a set of mechanisms to describe proactive, collaborative and adaptable agents – Linked Open Agents (LOAs) – through semantic contracts – Linked Agent Contracts (LACs) – by applying the Linked Open Data (LOD) approach [22], (ii) defining an agent model based on Workflows for Agent Control (WACs) and dynamic semantic discovery processes with the aim of adapting the behaviors of the LOAs at runtime in accordance with the changes in the IoA ecosystem, and finally (iii) a systematic method based on the philosophy of the agile methodologies to describe the general process for building IoA systems.

This paper is organized in six sections. Section 2 describes the main guidelines, architectures, mechanisms and tools employed to model the IoA paradigm based on the LOD approach. Section 3 presents a brief overview of the systematic proposed method, its stages and steps and its main inputs, outputs and tools required to support the building process of an IoA application. In Section 4, we present the modeling of a smart office-automation system to provide comfort-levels in a particular office of a research center. In Section 5, the potential benefits and applications of our methods for building IoA applications in domains such as smart city and smart healthcare are analyzed and discussed. Finally, the conclusions and future studies are outlined in Section 6.

2. Background

The IoA paradigm is founded on the principle of the *Agent of Thing* concept [15]. This novel concept highlights the integration of a component of intelligence and standard communications in order to enable objects to interact with other objects connected to heterogeneous IoT networks [17]. This concept is one of the first approximations towards the IoA approach, together with the ideas of Yu et al. [13] regarding the inclusion of mechanisms that allow the customization of agents by end-users to be carried out. Nevertheless, because the current web has an increasing tendency towards the Semantic Web, this work proposes the addition of smart units capable to handle Linked Open Data (LOD) in the IoA domain [22].

The use of linked data provides a more flexible publishing approach which makes it easier for data consumers to discover and integrate data sources distributed on the web [22,23]. This is possible by features of LOD such as: use of a uniform data model

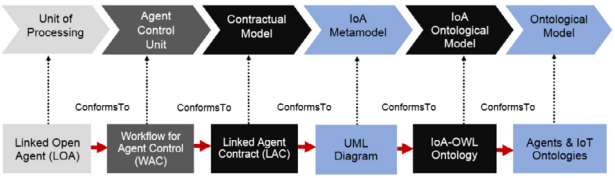

Fig. 1. Process and main tools oriented towards modeling the Internet of Agents paradigm based on the Linked Open Data approach.

to describe data at a global web level, use of a standardized data access mechanism to access data, use of a hyperlink-based data discovery to refer to data stored in a distributed way, and use of self-descriptive data to manage heterogeneous data sources. Against this background, the IoA level can be benefited from the use of LOD, contributing to agents with the ability to manage data transparently and also assisting them by providing uniform and global data from which tasks such as collaborative work, semantic decision-making and semantic interoperability are improved. Hence, agents described by using LOD are able to support semantic matching and reasoning processes. Consequently, an improved level of intelligence, autonomy and level of collaboration – as IoA ecosystems claim – is achieved.

Fig. 1 briefly details the main abstraction levels and tools introduced in this proposal to model the IoA domain based on the LOD approach. We aim to model an IoA ecosystem by means of a set of Linked Open Agents or LOAs – as a processing unit –, their corresponding LACs – as a semantic description unit – and their respective WACs—as an agent action control unit. Both the descriptive unit (LAC) and the functional unit (WAC) linked to each LOA are indispensable to provide capabilities to collaborate with other counterpart LOAs recovered by a discovery process as well as controlling directly the associated IoT-objects.

On the one hand, a WAC describes a sequence of actions that the agent must execute in order to achieve its goals. On the other hand, a LAC includes semantic descriptions of the agent properties supporting a contractual model based on a meta-model known as IoA-Contract. This meta-model has been formalized through an UML diagram composed of the classes that represent the concepts associated with the main issues that the IoA must handle, and which has been defined following the guidelines of the IoA-OWL ontology [12]. This full ontology has been created following the good practices of the Semantic Web, which suggests reusing concepts to describe entities on the web whenever possible. Hence, the IoA-OWL ontology has reused the most meaningful ontologies and Linked Open Vocabularies (LOV) published on the web to depict agents and IoT features.

2.1. Linked Open Agent (LOA)

We define a Linked Open Agent (LOA) as the main processing unit for the IoA approach. A LOA is an autonomous software unit that is semantically described by employing a special contract from which the resources associated to the agent itself are managed, allowing it to interoperate semantically with heterogeneous agents running over both intra-platforms and inter-platforms of agents. The adoption of a semantic description unit such as the LAC enables a LOA to be able to support reasoning, collaboration, negotiation, autonomous and proactive processes. These capabilities can be achieved assuming that a LOA manages a contract which saves extra information regarding the context, service ecosystem, social abilities, non-functional properties, IoT-resources, agent artifacts, and real-time constraints [12]. As result, LOAs may behave as sociable, collaborative, semantic, context-aware, smart, lightweight, auto-adaptable, standard-based, and open entities [24].

2.1.1. Reactive versus deliberative agents

The reactive agent architecture proposed by Russell and Norvig [25] can be employed to model entities that execute programmed or repetitive tasks based on events triggered by the global system. In addition, this model may also be a good alternative when the computation resources are not sophisticated or the entity is simpler to model. Moreover, the reactive architecture represents the best option whenever a high level of interaction as on IoA ecosystems is required. Therefore, this agent architecture is useful when collaborative tasks must be modeled.

On the contrary to the reactive architecture, the deliberative agent architecture modeled from using BDI artifacts such as Belief, Desire, Intention [26] represents a better option when the intelligence is managed at the level of an individual agent. This is because "traditional BDI model does not provide any specification of agent communication (or any other aspects at the social level)" [27]. In addition, this architecture is characterized by its inherent complexity, high power-computation requirements, and high efforts that developers must dedicate to create these types of agents. Therefore, the deliberative architecture is suggested in cases where the agent must act independently with a low level of interaction. But this is not the case for IoA ecosystems.

Against the conceptual basis of both reactive and deliberative architectures, we recommend the use of one model or another to implement the agent execution environment taking into account the social capabilities, the availability of computation resources where the agent will run, the complexity of the scenario and its level of dynamism, and, finally, the level of intelligence required by the developed system. Because it is very complex to coordinate a high number of deliberative agents as demanded by IoA, this proposal is specifically aimed at LOAs that adopt reactive architecture to implement the agent execution environment. However, if the development team believes that the use of deliberative agents is appropriate, the model can also be scaled for the linked data support.

2.1.2. Agent architecture

We have designed a novel agent architecture to support the LOA model. As is illustrated in Fig. 2, this agent architecture includes five additional components with respect to generic reactive architectures in order to handle the intrinsic semantic descriptors, service ecosystem, agent artifacts and social circles managed by LOAs. Among these new components that a LOA must coordinate are the following:

- A unit of semantic description that stores and manages the knowledge of the LOA and from which the agent requests to perform semantic reasoning and decision-making are built. This unit is responsible to handle the LOA contract, Linked Agent Contract (LAC) whose content is gradually updated in order to keep the agent up-to-date of changes that occur in the IoA ecosystem.
- A unit of control in order to know the control logic to be followed by LOA to manage the IoT-objects as well as the interactions with other agents of IoA ecosystem. This unit is on charge of the Workflow for Agent Control (WAC) whose content includes the sequence of actions that the LOA must execute.
- An agent scheduler or event trigger that coordinating a set of supporting agent tasks from which the LOA can achieve its goals. A generic LOA is composed by four tasks such as: an *Asynchronous Agent Response Task* that intercepts new FIPA-ACL (Foundation for Intelligent Physical Agents-Agent Communication Language) messages from external entities (e.g. external agents, users), a *Injection Task* that allows the agent to add new tasks at runtime as part of its behavior (same as it has been performed in Agent Dynamic EvoLutionary at runtimE (ADELE) middleware [28]), a *Workflow Execution Task* that executes the actions contained into a WAC and from which it can have control over the IoT ecosystem, and an *Agent Discovery Task* that enables the LOA to discover agent counterparts on the IoA ecosystem in order to achieve its goals collaboratively.
- A communication interface based on a REST interface that supports on the one hand the conventional FIPA communication standard language for the LOA in order to exchange messages with agents and on the other hand the conventional http communications to access to IoT-objects through Uniform Resource Identifier (URIs). Accordingly, the LOA can perform collaborative tasks based on uniform communication processes.
- Finally, the proposed LOA architecture includes an agent execution environment (e.g. Java Agent DEvelopment Framework (JADE) [29]) to provide hosting for the created LOA. This component acts as a platform of IoA agents that can provide hosting to several agents. With the purpose of maintaining compatibility with external LOAs, it is mandatory for the agent execution environment to support the FIPA-ACL standard for LOAs. This allows LOAs to be able to establish communication at both the intra-platform and the inter-platform IoA level.

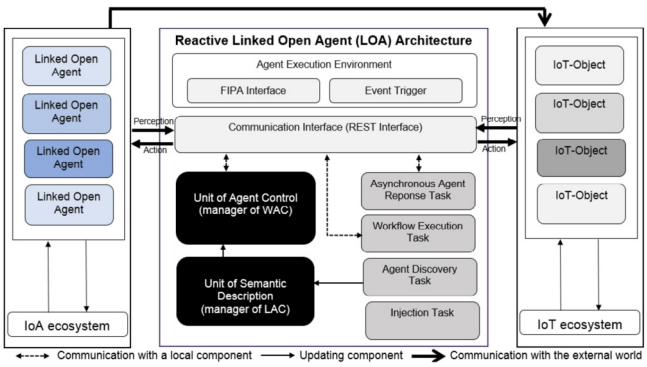

Fig. 2. General agent architecture proposed for building reactive Linked Open Agents.

2.2. Contract model

Each LOA instance has a specific contract known as LAC associated to it. This contract contains a finite set of semantic descriptors-values related to the six issues that an IoA ecosystem must handle: context-aware, social context, service ecosystem, IoT resources, agent artifacts and non-functional data regarding agents linked to IoT-objects.

As Fig. 3 shows, descriptors-values associated with the IoA approach are organized within the contract of LAC type by merging them in seven open linked documents – one general contract and six contractual sections associated with each IoA issue – formalized by following the grammar of JavaScript Object Notation for Linked Data (JSON-LD) which is a popular method of encoding bound linked data on the current web because this is easily comprehensible by both humans and machines [30].

2.3. IoA-Contract metamodel

The schema adopted by the contract model for creating LACs strictly follows the guidelines of a metamodel called IoA-Contract. This metamodel – implemented through the IoA-building Tool

```
{
  "@context":
  {
    "ioa": "http://core.ugr.es/ioa/owl/ioagent.owl#",
    "aprof": "http://core.ugr.es/ioa/owl/agentprofile.owl#",
    "acont": "http://core.ugr.es/ioa/owl/agentcontext.owl#",
    "asoci": "http://core.ugr.es/ioa/owl/agentsocial.owl#",
    "aserv": "http://core.ugr.es/ioa/owl/agentservice.owl#",
    "athing": "http://core.ugr.es/ioa/owl/agentosmartthing.owl#",
    "amodl": "http://core.ugr.es/ioa/owl/agentmodel.owl#",
    "repo": "http://rt.ugr.es:5984/ioahost/",
    "@language": "en"
  },
  "_id": "ioa_27ad68d2b2eb00fcfd8b82263403209",
  "aprof:AgentProfile": {
    "@type": ["aprof:AgentProfile"],
    "@id": "repo:profile_27ad68d2b2eb00fcfd8b82263403209"
  },
  "acont:AgentContext": {
    "@type": ["acont:AgentContext"],
    "@id": "repo:context_27ad68d2b2eb00fcfd8b82263403209"
  },
  "athing:AgentSmartThing": {
    "@type": ["athing:AgentSmartThing"],
    "@id": "repo:smartthing_27ad68d2b2eb00fcfd8b82263403209"
  },
  "amodl:AgentModel": {
    "@type": ["amodl:AgentModel"],
    "@id": "repo:model_27ad68d2b2eb00fcfd8b82263403209"
  },
  "aserv:AgentService": {
    "@type": ["aserv:AgentService"],
    "@id": "repo:service_27ad68d2b2eb00fcfd8b82263403209"
  },
  "asoci:AgentSocial": {
    "@type": ["asoci:AgentSocial"],
    "@id": "repo:social_27ad68d2b2eb00fcfd8b82263403209"
  }
}
```

Fig. 3. General contract of a Linked Open Agent instance.

– includes the main classes that, on the one hand, allow the data structures and datasets handled by the IoA approach to be organized and managed and, on the other hand, classes that allow the automatic generation of contractual sections and the general contract associated to each LOA are also included. All of these classes belong to a specific category such as the following: (i) general-contract class, (ii) contractual-section classes, (iii) structural-section classes and (iv) data-structure classes.

Regarding the classes categorized as (i), (ii) and (iii), these include the suitable methods aimed at executing the actions specialized in performing both the process of semantic enrichment of LACs (filling each descriptor with its corresponding value) and the automatic generation of the general contract (creating the suitable linked documents and their content). The classes of category (iv) are focused on creating the datasets related to the IoA approach (e.g. agent providers, arguments, real-time parameters) which are the input to carry out the semantic enrichment of contracts.

2.4. IoA-OWL ontology

The ontological model implemented through the IoA-OWL ontology [12] shown in Fig. 4 integrates the suitable vocabulary to describe LOAs, based on six essential components associated with six elemental issues that the IoA paradigm must manage. These components are summarized as follows: the context where the agent acts (*AgentContext*), the social circles with which the agent interacts (*AgentSocial*), the resources offered by the agent as specialized services (*AgentService*), the complementary IoT-resources that the agent must handle (*AgentSmartThing*), the descriptive information and real-time constraints associated with the agent (*AgentProfile*), and finally the description of artifacts that compose each agent and which define its behavior (*AgentModel*) [12].

2.5. Ontologies reused

The IoA-OWL ontology [12] described previously was built by reusing existing semantic models (e.g. ontologies and LOV) which have defined specialized vocabulary closely related to software

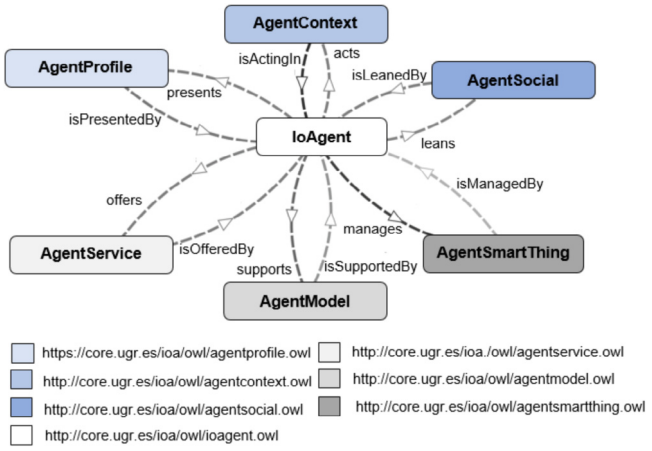

Fig. 4. General schema of the IoA-OWL core ontology.

agents and IoT issues. Nonetheless, the creation of a full model like the IoA-OWL also required new concepts, attributes, and relationships to be defined, in order to integrate and describe the main issues associated with the IoA.

On the one hand, the main ontologies employed to define new concepts within the IoA-OWL model, we have selected the following: (i) *Time* to specify temporal properties, (ii) *Cobra* to define spatial attributes of physical objects, (iii) *CoDaMos* to formalize contextual elements, (iv) *INRIA SNA* to detail social issues based on the Social Network Analysis (SNA) theory, (v) *AgentOWL* to describe components of agents following the guidelines of JADE [29], (vi) *SOUPA* to formalize BDI components of deliberative agents, (vii) *OWL-S* to describe Semantic Web Services (SWSs), (viii) *general ontology for IoT* to introduce technical issues of IoT hardware, and finally (ix) *BDI-workflows* to describe the workflows of deliberative agents.

On the other hand, some LOVs have also been reused, including: (i) *Friend Of A Friend (FOAF)* to describe interpersonal relationships, (ii) *VCard* to describe people and organizations, (iii) *Dublin Core* to introduce metadata of web resources, (iv) *Computer* to detail resources associated with computers, and finally (v) *M3-lite* to describe the attributes of IoT-objects.

2.6. Workflow for Agent Control (WAC)

The WAC is the control unit of the LOA defining the agent's behavior. It contains a description of the sequence of actions that the agent must execute to accomplish its goals. Although the workflow concept may differ among developers, and also be implemented in many ways and with different technologies, we have formalized the workflow using a document that follows the grammar employed by JavaScript Object Notation (JSON)—a popular method of encoding bound data on the current web. The general schema of a workflow is illustrated in Fig. 5.

A workflow can include specific control actions, invocations of operations executed over the IoT-objects through web services and even requests made to partner agents. The control actions can be specified through one or more rules IFThese-ThenThat [31]. Each rule is represented by an antecedent (IF-THESE) and a consequent (THEN-THAT). The antecedent of the rule can be determined by a set of one or more simple conditions. Each simple condition is defined through a channel – a service offered by an IoT-object or an agent that can be invoked when LOA has to access to sensor data or has to manipulate actuators through an agent or directly through a specific IoT-object – , a relational operator and a nominal value. All simple conditional blocks of the antecedent are then checked by

```
1. {
2. "workflow": {
3.    "workflow name": "control-HVAC",
4.    "workflow description": "collaborative control HVAC",
5.    "if-these": [
6.     {
7.        "channel": {
8.            "channel-data-needed": "humidity",
9.            "channel-data-unit-measurement": "%",
10.           "channel-agent": {
11.              "agent-name": "to be discovered",
12.              "agent-host": "to be discovered",
13.              "agent-port": "to be discovered"
14.           }
15.       };
16.       "if-then relational operator": "GREATER THAN",
17.       "if-then value": "value"
18.    },
19.    {
20.      "channel": {
21.          "channel-data-needed": "temperature",
22.          "channel-data-unit-measurement": "C",
23.          "channel host": "https://xx.smartoffice.com",
24.          "channel object": "hvac_office",
25.          "channel resource": "state"
26.       },
27.       "if-then relational operator": "GREATHER THAN",
28.       "if-then value": "value"
29.     }
30.     ],
31.    "then-that": {
32.       "channel": {
33.           "channel-data-needed": "temperature",
34.           "channel-data-unit-measurement": "C",
35.           "channel host": "object network",
36.           "channel object": "object name",
37.           "channel resource": "state"
38.       },
39.    "then-that value": "value"
40.   }
41. }
42.}
```

Fig. 5. General schema of a LOA workflow.

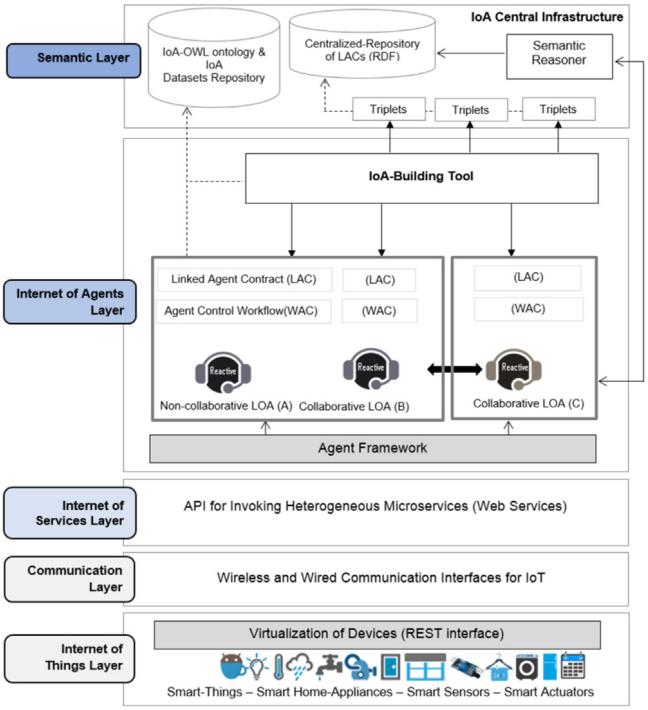

Fig. 6. A reference architecture for the Internet of Agent based on the Open Linked Data approach.

means of a logical operator. According the Boolean value returned by the antecedent, the consequent part of the rule is executed. The consequent can have one or more blocks to be executed depending also on the defined logical operator. An AND operator implies the execution of the whole block of actions defined in the consequent rule and OR indicates that any of the defined blocks may be executed. For example, as shown in the implementation of the WAC in Fig. 5, both IoT-object-based channels (lines 20–26 and 32–38) and agent-based channels (lines 7–15) can be combined within a WAC when an automated process on IoT is needed.

When the WAC is newly linked to a LOA, the values of the workflow descriptors initiate gaps (e.g. lines 11–13) as their definition is generic. The LOA is the entity which autonomously and in accordance with the data provided by the IoT-objects and agent (e.g. real-time constraints, location context and bandwidth) seeks the most suitable counterparts in the IoA ecosystem to execute each required channel. Depending on the metadata of the counterparts recovered by the agent, the LOA, through its control unit, steadily completes the content of its associated WAC.

During the process of completing the LOA workflow, the following two cases may occur: (i) direct completeness, where the LOA only explores once the IoA ecosystem and from it, the agent finds the counterparts that allow all the channels of the WAC to be executed, (ii) progressive completeness, where the LOA requires to explore the IoA ecosystem more than once because not all requirements defined by the WAC of the LOA are found.

Likewise, when the LOA achieves its WAC level of completeness, two additional cases may arise: (i) decompletion by failure of an IoT-object, and (ii) decompletion by failure of an agent. In both cases, the LOA, which previously maintained its full WAC, must fully re-explore the IoA ecosystem to replace the entities associated with the faulted channels. This is the technique used by the LOAs to recover from possible failures in the IoT or IoA ecosystem. Furthermore, the LOA can be maintained by continuously exploring the IoA ecosystem in search of agents and IoT-objects with better Quality of Service (QoS).

2.7. Reference architecture for the IoA approach

Due to the fact that there is not currently a standard reference architecture oriented at developing IoA applications, we have implemented a proposed architecture driven by both LOAs, WACs and LACs, organized in five layers: physical Internet of Things, Communication, Internet of Services (IoS), Internet of Agents (IoA), and the Semantic layer at the higher abstraction level. These layers, illustrated in Fig. 6, were defined taking into account two approaches related to the Future Internet [32] – IoT [33] and IoS [34] –, and technologies associated with the Web Semantic. Thus, the data processing carried out by the agents is based on their content and not only according to their syntax.

It is important to highlight that is not the intention of this paper to introduce new changes at the IoT level (Internet of Things, and Communication layers), but instead to introduce the LOD approach in layers with a higher level of abstraction, such as the service, agent and semantic layers. Additional details of this integration process are described as follows:

- *Internet of Things Layer.* At the lowest abstraction level, this layer manages the sensors and actuators of IoT-objects connected to a physical IoT infrastructure [3]. These objects must be compatible with wired or wireless communication interfaces of the IoT so that their linked agent (LOA) can establish communication processes and consequently, the agent may execute the suitable control actions where the specific object is involved.
- *Communication Layer.* This layer establishes communication between IoT-objects and the next-up layer Internet of Services Layer by means of wireless communication such as ZigBee, WiFi, WiMax, 3G and 4G cellular, among others [3]. Both Internet of Things and Communication Layers have been employed in IoT Reference Architectures. Therefore, it is possible that communications can be resolved by using any currently-existing IoT middlewares (e.g. UbiSOAP, Ubiware and OpenHAB [35]).

- *Internet of Services Layer.* This proposal highlights the use of service-orientation technology [36]. As a consequence, an *Internet of Services Layer* has been included in order to support in the IoA some mechanisms aimed at executing distributed tasks by invoking microservices implemented as web services. These web services are deployed in heterogeneous Service-Oriented Architectures (SOA) and Resource-Oriented Architectures ROA [37].
  The model employed to develop each service depends on the requirement and the technological resources to be deployed. Nonetheless, it is recommended to employ a tool for consuming heterogeneous web services such as the proposed in [38]. This tool enables agents to access IoT-objects by invoking Representational State Transfer (RESTful) [39], Device Profile for Web Services (DPWS) [40], Simple Object Access Protocol (SOAP) [36] and Dynamic Open Home-Automation (DOHA) web services [41], respectively. This order allows obtaining a better performance in the processes executed by LOAs and any other entity that operates in IoA ecosystems [38,42].
- *Internet of Agents Layer.* This layer focuses on coordinating a set of LOAs as well as managing their components such as LAC and WAC, as described in Fig. 2. Part of the coordination process is oriented to coordinating the tasks associated with LOAs, such as Injection, Asynchronous Agent Response, Workflow Execution and Agent Discovery tasks. These tasks have been strategically defined at development time. However, LOAs can support the addition of new tasks at runtime using the Injection task if required. This specific task is driven by the technique employed by ADELE middleware [28].
  In addition to the coordination of the LOAs' tasks, because each LOA must have a LAC to describe their properties and a workflow that addresses the LOA behavior, this layer must also handle these elements. Both, LAC and WAC are locally published in an agent execution environment that is part of the architecture. However, this layer also allows LOAs to store and update LAC as global data in order to enable agents to execute reasoning processes to discover eventual partner agents.
  It is important to underline that this layer – implemented by the IoA-Building Tool – has followed the conceptual baselines of the reactive agent architecture using JADE framework [29]. However, the employed agent framework can be replaced by other one, but it is required that the concepts adopted by the new framework are integrated into the IoA-OWL ontological model.
- *Semantic Layer.* This layer is responsible for managing the coexistence of semantic knowledge and linked data, in order to execute semantic reasoning requested by LOAs from the IoA Layer in a compact format. This layer is a centralized infrastructure called *IoA Central Infrastructure* that includes two repositories. The first one is the Centralized-Repository of LACs enabling semantic reasoning on IoA. On the other hand, the second repository, IoA-OWL and Datasets Repository, stores the IoA-OWL ontology and the specific datasets of the IoA (e.g. parameters, owners). Both the ontology and datasets can be reused to facilitating the semantic enrichment of LOAs. From the replica of LACs in a central host – in Resource Description Framework (RDF) triplets in order to maintain compatibility with the RDF graphs handled by the current LOD software tools such as SPARQL [43] – not only is it possible to perform reasoning process that LOAs can request, but also additional important tasks are possible at IoA global level, such as agent identification, testing of security, and auditing process. In this way it is possible to establish security control actions that are of utmost importance in the domain of IoT and consequently in IoA.

2.8. Routing method for the IoA

Multiple heterogeneous nodes (IoT-objects) participate in IoT networks. These nodes are capable of providing the data (sensing) and control (actuating) requirements needed by users. Therefore, for an IoT network to behave smartly and maximize network resources, it is important that each node adopts a routing mechanism that determines the optimal route to carry out data transmission from a source node to a destination node (Singh2016). In this respect, cognitive routing mechanisms have been proposed for IoT networks based on reasoning and learning (e.g. adaptive routing approach (ARA) [44], learning data delivery A* (LDDA*) and cumulative-heuristic accelerated learning (CHAL) [45])) from which metrics such as quality of services (QoS) [46] and quality of information (QoI) [45] of heterogeneous nodes connected to one or more networks have been optimized. However, these mechanisms operate at the physical network level; that is, updating the routing table associated with each node, which is outside the scope of the proposed IoA model.

The IoA model, on which this work is based, has not delved into the Internet of Things or the Communication layers. However, because the use of cognitive routing methods is a key element in IoT's smart network building, our approach employs a routing method at a higher level of abstraction. This method uses a cognitive mechanism handled by agents.

The proposed routing method based on agents of type LOAs is based on a semantic reasoning mechanism derived from the LOD approach that is driven by the descriptors of LACs that define the IoA ecosystem. This reasoning process, requested by the own LOA to the IoA Central Infrastructure, explores the current IoA ecosystem and thus, a LOA obtains the routes of IoT-objects with which it must communicate to achieve its goals.

Although the reasoning process addressed by a LOA is not focused on optimizing the QoS parameters at the network level (e.g. latency, reliability, accuracy, relevance and robustness), it obtains and selects the best links to other LOAs and IoT-objects. In general terms, the routing method at application level used in IoA is based on the QoI metric such as data recovered accuracy that external LOAs can provide to the requesting LOA. Therefore, the objective of the reasoning process is to identify the links to LOAs that provide the data required (e.g. temperature, humidity) in the suitable context in order to execute a process regardless of the path to be followed. Hence, a LOA carries out a set of unicast – delivers a message to a single specific node – processes to execute its WAC, and from which it reaches its goal.

2.9. Design aspects for modeling the IoA

Aspects related to the satisfaction of QoS and fault-tolerance requirements are intrinsic to approaches such as IoA and IoT because both approaches propose modeling dynamic networks in which the use of resources can vary according to the context [47]. Against this assertion, this section details the main implications and metrics considered for modeling IoA ecosystems in order to optimize the performance of this type of systems.

2.9.1. QoS aspects for modeling the IoA

QoS in communication networks is defined as the "network ability to achieve a more deterministic behavior, so data can be transported with a minimum packet loss, delay and maximum bandwidth" [48]. However, Li et al. points out that for IoT systems – and in our opinion also IoA systems – the definition of QoS is not completely clear [47]. This is mainly because the definition of service in this type of network differs from conventional networks (e.g. wireless sensor networks). For example, a service in IoT and IoA can be defined as an autonomous software entity providing a

way to access some specific resources (e.g. sensor data, actuators) by using a uniform interface, as well as its processing, the execution of control actions and communication among objects involved.

Traditional QoS metrics such as throughput, delay or jitter are not the most appropriate at application level even though they have dependence [47]. These metrics are specific to the network layer level, and the physical network level of IoT [49], that is, for routing or Medium Access Control (MAC) schemes, for aware routing, for energy-efficiency routing, and for traffic flows. But IoA is at a higher level and, therefore, it must handle other metrics. These metrics are generally similar to that of existing QoS in web services [47] – scalability, performance, reliability, availability, flexibility, accuracy, integrity, security and privacy [50] –, that is, to improve the accuracy of the recovered information and the lifetime of IoT-objects linked to LOAs.

The IoA is mainly focused on providing a more precise control process by using recovered more accurate data and taking care that data is transmitted and handled with a certain level of security to avoid the discriminated access to information. To do it, some metrics of QoS have been implemented in the IoA model. These metrics have the following scope:

- *Data recovery accuracy*. LOAs have associated a mechanism for discovering external counterparts based on both a semantic reasoning and a selection process. The first one recovers the counterpart candidates based on a set of semantic descriptors received as arguments (LAC descriptors) and the second one applies a selection criteria (e.g. better response time, specific context, social power of the counterparts) to provide the most suitable agents which are part of a collaborative process. Both processes enable the recovery of counterpart on IoA is more accurate than currently systems that operates based on the discovery of resources according the syntax of data [51].
- *Availability*. Since the IoA model uses a service layer (Internet of Service Layer) as part of its reference architecture, the service availability property is inherited by the IoA model. As a result, LOAs can use stateless components such as web services to carry out their control tasks over IoT-objects. Moreover, all the metrics associated with the previously specified web services are also inherited by the IoA.
- *Reliability and performance*. LOAs have the capability of supporting communication at an agent level with no loss of messages and in an efficient way. This is because the LOA architecture has been built adopting JADE framework as an agent environment which is a very efficient and reliable environment for developing agents [52]. Furthermore, because LOAs define a WAC that drives the agent behavior, LOAs have the capability to discover and select the counterparts with a better response time. Consequently, the actions executed over the IoA ecosystem are addressed by the most efficient agents.
- *Security and privacy*. LOAs can establish secure agent communication processes based on digital certificates handled at IoA level on secure channels. Additionally, with regards to data privacy, the LOAs handle a contract which stores semantic descriptors. These descriptors are stored according two levels of privacy such as private and public. This avoids that confidential data associated with the operation of the agent can be accessed by users of agents who are not authorized to do so. It is important to highlight that our proposed model is not aimed at fully covering the security and privacy aspects but at improving the accuracy of the retrieved information. Therefore, this metric can perhaps be improved by applying security mechanisms specific for agents or IoT such as more complex authentication methods, user informed consent, agent-level data encryption and communication mechanisms, new data privacy levels, authorization and access control, and among others [53,54].

In addition to the above described metrics, the IoA model also accomplishes other additional metrics inherent to the IoT technologies because the reference architecture includes a specific layer to handle the IoT-objects such as the Internet of Things Layer. Main metrics covered by the IoT layer depends on the technologies employed to implement it. However, the metrics of QoS that this approach inherit to the IoA cover issues such as: interoperability, scalability, flexibility, adaptability, and security [55].

2.9.2. Fault-tolerance aspects for modeling the IoA

As Yung-Li et al. assert, resilience is one of the major issues to provide QoS in IoT applications [56]. Therefore, this issue is also relevant in IoA. The fault-tolerance aspect, as well as those related to QoS, can be managed at different levels in IoT, for example, the physical and logical network level, service as well as the application level. To cover the fault-tolerance at the service and application level, some mechanisms focused on controlling fails from software procedures can be employed—redundancy and diversity, recovery block, and exception handling [56]. In the interest of covering this issue, the following mechanisms have been implemented by the IoA:

- *Handling exceptions*. All processes supported by an IoA platform are managed by the IoA-Building Tool. This tool is a specific software that enables developers and users to manage the elements (LOAs, LACs, and WACs) together with extra supporting agent tasks that agents execute by themselves during their life cycle.
  Regarding the IoA elements, the IoA-Building Tool has been developed to be able to intercept errors and warnings, and handling exceptions, that can be triggered during their managing. These processes are categorized according each element as follows: procedures related to LOAs – creation, deleting, communication, injecting new agent tasks, restoring –, procedures associated with LACs – creating, storing, querying and altering –, and procedures that handle WACs—creating, linking to LOA, invoking channels, agents and web services. On the other hand, with regard to the extra IoA processes, the IoA-Building Tool also performs an exhaustive control of the exceptions over the modules of requesting and processing of provided by the semantic reasoning, converting LACs from JSON-LD to RDF triplets, and executing each stage of the automatic generation of both LACs and LOAs (e.g. checking data and URIs, validating Universal Unique Identifiers (UUIDs), verifying the integrity of agent platforms).
- *Fault recovery*. The recovery from failures occurs in IoA at two levels: at LOA level and at IoT-object level. In the first case, when a LOA requests a counterpart agent that it is not available on the IoA ecosystem, it receives an intercepted exception. Then, the LOA checks its WAC and executes a discovery process in order to replace the failure agent with other functional LOA. On the other hand, the second case is produced when an IoT-object is not available. Our model proposes linking a LOA to each IoT-object. Therefore, if a specific object is not available, its manager agent can intercept the error and notify it to any other interested counterpart agent. This agent can keep executing providing previous values or actions from the corresponding IoT-object or can wait in order to reconnect IoT-object.

3. Systematic method

Agile methodologies for software development propose a model based on an iterative and incremental process. This process starts with the definition of requirements, their development, and their integration until the target goal is achieved, without establishing limits if new requirements are demanded in the future [57].

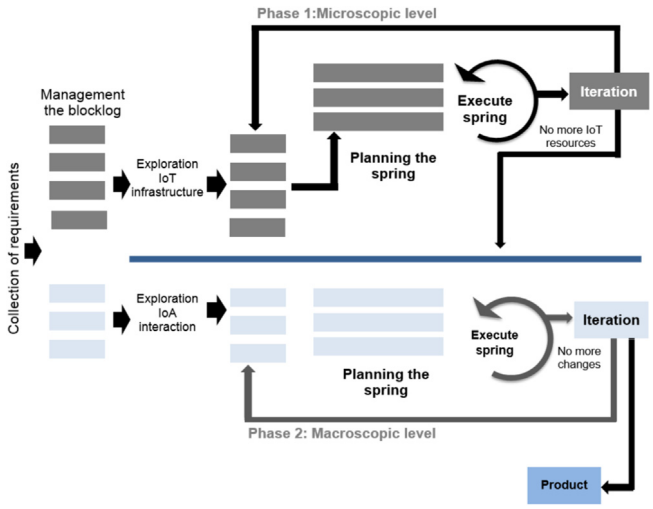

Fig. 7. General schema of the proposed method for building an IoA systems.

In dynamic scenarios like the IoT and the IoA, there is a constant demand for adaptation regarding processes and quality of service. This is due to the fact that the development processes generally start with objectives that are not clearly defined. The changes can be introduced at both component (IoT-objects and agents) and integration (IoT and IoA networks) level. For these reasons, the employment of atomic software components such as web services, and agile software development methodologies that carry out the adaptation process through the integration of new components in a software product is currently very helpful. Therefore, the method we propose follows the guidelines of agile methodologies.

The assumption is that an IoA system based on the LOD approach is governed by a collection of LOA entities, which are semantically described by using LACs. Each LOA can have associated an IoT-object connected to an IoT network which is controlled through a WAC oriented to the specific IoT-object. Fig. 7 details the general schema proposed by our method. This is a bottom-up method, based-on the principles followed by the SCRUM framework [58], and it is divided into two phases, macroscopic and microscopic levels.

The proposed method starts with a collection of the global requirements that must be introduced into an IoT network that needs to be scaled, in order to achieve a smarter behavior. The work team collects these requirements from users, customers and other stakeholders in a requirements elicitation phase. Once these requirements are defined, a set of tasks applied in phases at agent level (microscopic) and at infrastructure level (macroscopic) are generated.

The phase at *microscopic level* carries out an exploration process of the IoT infrastructure where the objects connected to the network are modeled as LOAs. For every single object, an iteration process must be performed in order to convert the generic IoT-object into a LOA entity. This type of agent has been grouped in one specific partition in order to better organize the agents belong to IoA ecosystems. This partition is outlined as follows:

- *Partition 1 - Agents of Things*. This partition only includes agents of type LOAs that have associated an IoT-object. The agent mission is to handle the resources that each object integrates (e.g. temperature agent). Thus, this stage only works at the individual LOA entity level.

On the other hand, the phase at *macroscopic level* takes the LOAs modeled in the first phase in order to integrate and coordinate them within a unique network. This second phase explores the requirements of LOA interaction, social circles, collaboration, coordination, data processing, user-interaction and intelligent behavior of the network, iteratively. In order to better organize the agents that form part of the network three more agent partitions have been defined as follows:

- *Partition 2 - Agents of Coordination.* This includes agents whose purpose is to perform acts of communication with other agents of *Agent of Things* type, in order to coordinate the processes needed to execute tasks that are aligned with the accomplishment of the goals proposed by the application (e.g. comfort agent).
- *Partition 3 - Agents of Data.* This kind of agent is basically oriented at processing any kind of data captured by the entities belonging to the *Agent of Things* with the purpose of generating valuable information that allows important decisions to be taken regarding the consumption or use of the resources associated with the IoT-objects (e.g. fuser agent).
- *Partition 4 - Agents of People.* This type of agent provides an interface (e.g. using URIs) for communication between the end-user and the IoA application. Thus, users can interact and personalize their requirements through this type of agent (e.g. user-setting agent).

The macroscopic modeling must finish when all requirements related to the global IoA ecosystem have been processed and stored in the task blocks. However, it is important to note that, if new requirements are introduced later, the method must be applied again in an integral way in order to introduce these changes.

Both phases that have been described previously are detailed in the general process of the proposed method shown in Fig. 8. Before starting the development process by applying the proposed method, it is necessary to have the general map of the IoT physical network as input, in order to identify each device connected to the IoT network which is being scaled towards the first phase of an IoA network. With regards to the structure of each phase, the first of these is comprised of six stages (MI-1–MI-6) and the second one is formed by seven stages (MA-7–MA-11). In both phases, the stages are categorized as semi-automated (MI-2, MI-3, MI-5), manual (MI-1, MI-4, MA-8, MA-9), automated (MI-6, MA-8, MA-9, MA-10, MA-11) or hybrid (MA-7).

For semi-automated stages, the work team must employ tools to discover public agents or services available on the web. For manual stages, the work team must focus on obtaining information (semantic descriptors) that describe the IoT, IoA, and IoS ecosystems. In the case of automatized stages they are supported by new libraries that allow the creation, management, and validation of the contract associated with the LOAs (IoA-Building Tool). Finally, regarding hybrid stages, various stages – manual, semi-manual, and automatic – are combined into a single procedure.

3.1. Method overview at microscopic level

The first phase of the proposed method illustrated in Fig. 8 is aimed at defining a LOA for each IoT-object connected to the IoT network studied. Next, the six stages that form part of this phase are detailed, giving a brief general description, their main inputs and outputs, as well as the main tools recommended in order to successfully accomplish this phase.

To begin this phase, it is necessary to dispose of a mapping of the currently-built physical IoT network which provides useful information about the objects, servers, and any other network devices connected to the physical infrastructure. From this information, it is possible to define the new requirements to give LOA a more intelligent behavior as well as its corresponding sociability with other agents. It is important to highlight that, if it is not possible to access the map requested by this stage, the IoT network must be

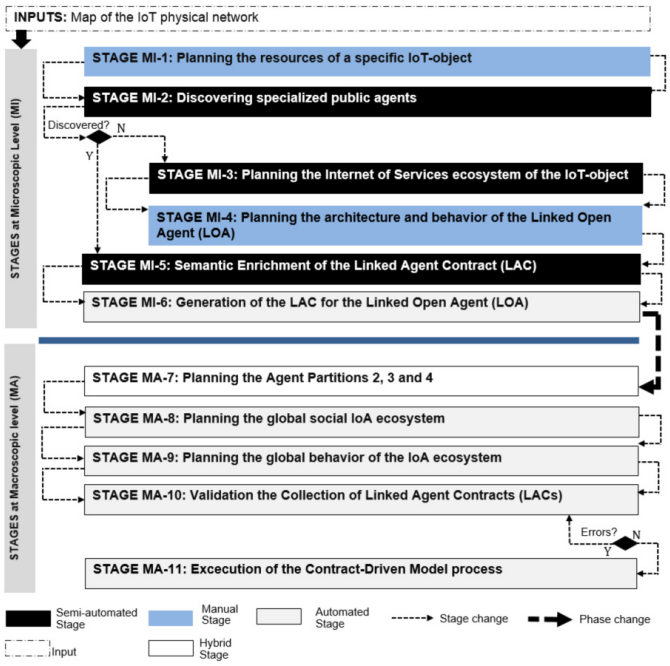

Fig. 8. Overview of the process of the proposed method.

analyzed and tested in order to build this map, including data such as: fixed and mobile IoT-objects, sensor and actuators, gateway nodes, or servers. In addition, it is important to collect data from each previously-listed element that describes the following issues: Internet Protocol (IP) address, host name, ports, communication protocol compatible, type of connection (wireless, wired), logical name, services managed, and general computation resources (e.g. memory, programming supporting, battery).

3.1.1. Planning the resources of a specific IoT-object (MI-1)

This stage is a manual process that must be carried out for every single object connected to the studied IoT network to be modeled as an IoA infrastructure. This stage is comprised of three steps and its main goal is to collect information related to the capabilities and resources associated with the specific IoT-object: type of node according to the mobility (mobile or fixed), type of node according to the privacy level (public or private), the context where it operates, the communication protocol employed, and the external networks with which the node communicates. This information is essential to carry out the conversion process of IoT-objects into *Agent of Thing* type LOAs, in order to cooperate in the optimization of the management and use of resources, and the quality of service offered by the IoT-objects.

Description of the IoT-object resources (MI-1.1). Basically, this must collect information that satisfies the concepts integrated into the *AgentSmartThing* component that takes part in the IoA-OWL model. Among the main descriptors that must be defined by the specific object are the following: *identifier* (id_i), logical name (ln_i), data generated (dp_i), and *brand* (b_i) of the IoT-object, and, additionally, some references to external entities such as geolocation (G_i), location (L_i), time (T_i), and a set of logical (e.g. applications, agents, services) and physical IoT-resources (R_i) (e.g. memory, battery).

Publishing datasets of the IoT-object resources (MI-1.2). This step consists of the publication of the information regarding the IoT-resources obtained in the previous step (MI-1.1) by adopting the LOD approach. Then, after this step is concluded, we provide a collection of datasets that semantically describe a set of IoT-objects and a set of public and private IoT-resources associated to specific objects, which will be reused at the time of building the contract. The persistence of this data as well as data generated in later stages must be managed by the IoA Central Infrastructure through the IoA Datasets Repository (Fig. 6). This repository must be implemented by using a documentary database based on the REST principles [59] in order to access the items through REST invocations, the same as linked data are accessed.

3.1.2. Discovering specialized public agents (MI-2)

This stage is comprised of three steps and is focused on discovering one or more software agents that partially or completely satisfy the tasks that the specific IoT-object must execute to accomplish its goal. The discovery process can be realized manually by developers when they have the necessity to define specific agents. Therefore, if there is such restriction, this stage must continue in a step-by-step process until each step is completed. All recovered agents are defined as counterparts of the modeled LOA through its WAC. Conversely, if the modeled system does not include any restrictions regarding the counterpart agents with which the LOA modeled may establish cooperation, the next stage (MI-4) is continued directly. In this case, the agent discovery process is performed by the LOA itself at runtime and the WAC of the agent is gradually completed as useful partner agents are discovered.

This stage starts with the functional requirements associated with the specific IoT-object and must deal with two situations. The first of these is aimed at discovering public agents which adopt the LOA model to accomplish one or more functional requirements associated with the specific IoT-object. This implies that the discovered agents will have the semantic descriptors employed by the IoA-OWL model, which gives them the capacity to discover other agents and also to perform automated processes based on their semantic descriptors. However, external conventional and public agents can also be discovered in the second situation. In this case, it is necessary to add the semantic descriptors of both the *AgentProfile* and *AgentContext* to the external agent. It is important to note that both processes may not be successful: in this particular case, step MI-3 must be followed.

Discovering public Linked Open Agents (MI-2.1). This step is focused on discovering agents that adopt the LOA model in order to add them to the modeled application, thus helping to solve one or more of the functional requirements that the IoT-object must satisfy. For each discovered agent, the URI of the general contract associated with the agent must be identified because it saves both the URI of the *AgentProfile* and the *AgentContext* component. To finish this process, if some LOAs are discovered then the process must go to step MI-2.3, otherwise it must continue sequentially with step MI-2.2.

Discovering public external conventional agents (MI-2.2). We only recommend applying this step when the discovered conventional agent is extremely specialized and provides data that is expensive to obtain through our physical infrastructures. It is important to note sometimes specialized actions are also provided through WSs. If this is the case then we advise adding this function to the IoS ecosystem and then processing them as services as detailed in step MI-3.

Defining the profile of the discovered agents (MI-2.2.a). To define the profile of an agent that does not support the LOA model, employing the concepts of the *AgentProfile* component is recommended. The main descriptors required to determine this component are the following: agent *identifier* (id_i), *name* (n_i), *description* (d_i), *status* (s_i), *version* (v_i), *date of submission* (ds_i), *date of publishing* (dp_i), *requests* (r_i), and *accomplished requests* (ar_i). Furthermore, it is necessary to include references to external entities, such as the model to create the agent (*createdFrom* (B_i)), owner (*hasOwner* (O_i)), credentials (*hasCertificate* (C_i)), and the real-time restrictions (*hasRTrestriction* (RT_i)) that the agent must satisfy.

Defining the context of the discovered agents (MI-2.2.b). In order to describe the context where the agent operates, we recommend using the concepts of the *AgentContext* component. The contextual data needed is focused on defining the following issues: a *identifier* (id_i), a *name* (n_i), and a *description* (d_i) of the agent context. In addition, it must include references to entities, such as geolocation (G_i), location (L_i), time (T_i), computation node (N_i), and the agent platform (P_i) where the agent runs. This latter information allows any LOA to know the technical parameters to establish communication with its counterpart agents and additionally, it allows them to complete the LOA WAC.

Testing the functional and non-functional requirements of the discovered agents (MI-2.3). In the Software Engineering field, the testing process allows errors or discrepancies between the current result and the expected result to be detected [60]. With the aim of preventing the reuse of failure-agents in terms of erroneous results or inadequate behaviors, this step recommends applying a unit test to every single agent discovered in this stage, in order to examine them individually and in an isolated way [60,61]. We recommend evaluating issues such as: response capability, capability of handling exceptions, FIPA communication, service quality, and data quality. However, non-functional issues such as reliability, usability and robustness of the agent can be inspected at an individual level.

We suggest that only those agents that overcome the proposed testing process, that is to say, the agents without failures – fatal, very serious, serious, or moderate – should be taken into account as part of the modeled application. Ultimately, this step provides a set of reliable agents which follow the LOA model, which is useful in the modeling of LOA behavior associated with the specific IoT-object (MI-4). However, if not all the required tasks are resolved by the recovered agents, then the next stage (MI-3) must be followed.

3.1.3. Planning the Internet of Services ecosystem of the IoT-object (MI-3)

This stage is carried out in a semi-automated way in order to try to discover already-developed microservices [62] that the specific object has previously employed to operate. We recommend employing specialized tools to carry out the discovery process of public services over both SOA [36] and ROA [37] architectures available on the web. Thus, we ensure that service- and resource-orientation technologies and their philosophy are supported with regard to the reuse of distributed components. In this way, some of the public components that are helpful to the modeled application can be integrated without the need to make additional efforts or to incur extra costs.

This stage begins with the collection of functional requirements related to a specific object. The main goal is focused on defining a network of microservices which satisfy each task that the object must execute at runtime. These components will form part of the structure of the agent WAC and the agent will model its behavior based on them. Then, they will solve perception and actuation tasks on IoT-objects. In order to contribute to simplifying the scalability of the system, it is recommended to only reuse atomic microservices [63].

Discovery of public microservices associated with the IoT-object (STEP MI-3.1). In this step, software tools must be employed to discover public microservices related to the domain of the target application at both levels, within the IoS ecosystems belonging to the IoA domain, and in conventional IoS ecosystems. The tools employed will depend on the domain of the modeled application, but we recommend using tools that guarantee the security of invoking, and that provide high-quality microservices in order to ensure the reliability and consistency of data offered by the modeled application.

All recovered services must be semantically enriched by employing the *AgentService* component because they do not preserve compatibility with the semantic description proposed by this component required by the ecosystem of services in the IoA domain. Among the main descriptors to add are the following: the URI associated with the description document of the service (*contract URI* (u_i)), real-time restrictions (*hasRTrestriction* (RT_i)), service profile (Pr_i), input parameters (IP_i), output parameter (OP_i), service operation (O_i), and resources (R_i) for RESTful services only.

We start from the premise that all services are operative, but if this is not the case, then the missing services must be built and semantically enriched, as detailed in this step, and, after that, it is possible to continue with step (MI-3.2).

Publishing the datasets of the ecosystem of IoS as open data (MI-3.2). This step takes the instances of the semantic description of each service carried out in step MI-3.1 to publish the descriptors as open data. When this step is finished, this stage provides a collection of both public and private datasets which describe the service ecosystem of the studied object, which can be reused to compose the agent contract later.

Testing the functional and non-functional requirements of microservices (STEP MI-3.3). Similar to the process detailed in step MI-2.3 where agents were tested, this step must follow the same process, but now it is aimed at testing the request–response of microservices.

3.1.4. Planning the architecture and individual behavior of the Linked Open Agent (LOA) (MI-4)

This stage is focused on defining the model components of the LOA according to the role that the agent plays in the IoA network, and organizing them in a logical way to model a more intelligent behavior, but only at entity level. To do this, it is needed to count with the IoT-network map and the IoS-ecosystem map associated to the IoT-object for which the agent is being modeled.

The four steps MI-4.1, MI-4.2, MI-4.3 y MI-4.6 represent independent steps of the agent model adopted by the LOA, but both steps MI-4.5 and MI-4.6 have dependency with respect to the agent model employed to build the LOA. Nevertheless, this method only emphasizes reactive LOAs.

Definition of the LOA role (MI-4.1). Before deciding which model to employ to build the LOA, it is important to determine its role with the purpose of comprehensively defining the main goals and tasks that it must perform at runtime. At this point, all modeled LOAs have an IoT-object linked and therefore the role of all agents at microscopic level is *Agent of Things*.

Definition of the LOA profile (MI-4.2). At this point, the agent to be modeled will be a new LOA, and therefore it is required to add the corresponding semantic descriptors to fill the *AgentProfile*. To add this information it is necessary to follow the same process as described previously in step MI-2.2.a.

Definition of the LOA context (MI-4.3). In this step, it is also required to add the corresponding semantic descriptors to fill the *AgentContext*. To do this it is necessary to follow the same process as described previously in step MI-2.2.b.

Defining the LOA WAC (MI-4.4). In this step, the nature of the IoT-object to be agentified (linked to an agent) is evaluated in order to determine the suitable control actions that this object should perform over the IoT ecosystem. These actions are managed through the LOA linked to the object.

The actions that the LOA must perform on the IoA ecosystem are specified by using a WAC. This element follows the scheme and syntax described in Fig. 5. In general terms, each WAC is composed of a set of channels that, when executed, allow communication with the corresponding external IoT-objects and agents and, consequently, the agent's collaborative goals are achieved.

For the definition of the content of a WAC, it is not required to specify the network metadata that allows communication with specific IoT-objects (e.g. lines 11–13, 23–25 in Fig. 5). These metadata are automatically discovered by the agent using semantic

discovery processes. The only mandatory metadata to be defined are those related to the data of interest for each channel (line 8.9 in Fig. 5). The discovery of IoT-objects and agents is carried out according to them. For example, a data of interest might be the temperature measured in °C, as illustrated in the WAC example illustrated in lines 21 and 22 of Fig. 5. In addition, complementary to the data of interest, the remaining elements that compose each rule of the WAC such as relational operators (e.g. greater than, less than, equal to) and nominal value (e.g. numerical, text) must be specified.

In order for agents to behave consistently, it is important that the control logic (rules) of each WAC is defined correctly. On the logic modeled by the WACs depends which partner agents will be discovered to control IoA ecosystems.

Defining the LOA tasks (MI-4.5). In the introduction of this stage we explained about two steps (MI-4.4 AND MI-4.5) which depend on the agent model selected to build the LOA. This is the first of these steps and it is focused on determining a set of agent tasks that address the LOA behavior.

This step proposes the use of agent tasks to structure the behavior of a reactive LOA. An agent task can be seen as an artifact consisting of a software component which performs any control action, data processing, and machine–machine and user–machine processes.

Defining the generic LOA tasks (MI-4.5-a). We have strategically defined four tasks such as: Asynchronous Agent Response, Injection, Workflow Execution and Agent Discovery Tasks. These specific tasks are generic for any LOA and they can be defined by using *InternalBehaviors*. These behaviors are integrated within the agent at the development stage by defining semantic descriptors, such as name (in_i), description (id_i), real-time constraints (iRT_i), arguments (iP_i), and the corresponding workflow (iW_i).

Defining additional LOA tasks (MI-4.5-b). The use of additional tasks is technically feasible for a LOA. A LOA has associated the Injection Task which enables it to integrate extra tasks if the functionalities provided by its generic tasks do not meet its goal. Additional tasks can be described by using *ExternalBehavior*. Therefore, they can be described by the same descriptors as *InternalBehavior*, but also including others such as the className (ec_i) which defines the class name of the agent artifact that the *on-the-fly* compiler will use as an argument to create the executable of the specific task. It is important to note that the integration of the described task is realized at runtime by applying metaprogramming techniques as is done in the ADELE tool [28].

Definition of the individual LOA behavior mechanisms (MI-4.6). This step focuses on the integration and modeling of the mechanism employed by the LOA to act coherently and autonomously based on the processing of the semantic descriptors stored in its LAC and the logic of control defined within its WAC. This is one of the most important steps, because it is the first approximation in terms of modeling the systems behavior more intelligent, and therefore it must provide agents with capabilities, in order to facilitate their coordination at a macroscopic level.

The implications for the intelligence of an entity-level LOA basically consist of determining how the LOA should behave in order to achieve its goals. To do this, we recommend modeling this issue by introducing the following mechanisms:

- *Characterizing the control actions that the agent must perform over the IoT ecosystem.* It is essential to define the control logic that the LOA must perform over the resources and objects of IoT in order for the automatic control process to be executed by the LOA, using its own resources or discovering the most suitable counterparts to help it achieve its goal. In other words, it is required to specify to the LOA its respective WAC.
- *Scheduling the Execution Workflows Task of the LOA.* The workflow (WAC) associated with the LOA provides the guidelines on which the LOA must be based to execute automatic processes. However, this component cannot be executed until the LOA temporarily defines and schedules a specific task called Agent Workflow Execution Task, which is part of a generic LOA. Then, it is necessary to schedule the frequency with which this task should be performed by the LOA.
- *Scheduling the Agent Discovery Task of the LOA.* Analogous to how the previous task was planned, it is necessary to also plan a specific task from all generic LOAs. This task is called the Agent Discovery Task and indicates to the LOA how often it must request from IoA Central Infrastructure the execution of semantic reasoning processes. This type of process is aimed at discovering counterpart agents that collaborate with it to achieve its goal.
- *Delimiting the Asynchronous Agent Response Tasks of the LOA.* This task is not a task for the LOA to plan. Its operation is executed whenever the LOA receives a message from other external LOAs. Therefore, and in order to avoid fraudulent communications that lead to violating the integrity of the LOA, of the data it handles, of the IoT-objects it controls and to whom it accesses, and of the counterpart agents with whom it communicates, it is necessary to define the vocabulary domain to which the agent must provide a response to involving or not a control or processing action on its environment.

For modeling the reactive-LOA mechanisms to support an intelligent behavior at microscopic level, we suggest the use of an UML Behavioral State Machine Diagram. This diagram is used to specify the discrete behavior of a part of a designed system through finite state transitions [64]. In addition, an UML Sequence Diagram can be employed to represent the tasks that the LOA executes when a message is received [64]. All tasks defined in the previous step (MI-4.4) must be integrated in a logical way in order to compose the behavior of the reactive-LOA. At end, these diagrams together with the generic WAC schema associated with the LOA will be useful for understanding the event-trigger mechanism that addresses each LOA on IoA.

Publication of datasets with the LOA components as open data (MI-4.7). The semantic descriptors associated with the LOA's artifacts defined in step MI-4.4 must be published following the same approach employed for the previous publication of data. All descriptors published in this step must be private for the developer of the application and they can be employed later to generate the agent.

3.1.5. Semantic enrichment of the Linked Agent Contract (LAC) (MI-5)

This stage is responsible for creating an instance of five of the components integrated by the IoA-OWL ontology – *AgentProfile*, *AgentContext*, *AgentService*, *AgentModel*, *AgentSmartThing* – that the LOA manages itself without taking into account the agent interaction issue. To fulfill the fields of these ontological instances primitive data types (e.g. strings, numeric, boolean) can be used. However, most of these must be introduced through URIs which link to the semantic descriptors previously published as open data.

In order to accomplish this stage, we recommend using the descriptors obtained in each one of the previous stages as follows: (MI-5.1) enrichment of the *AgentProfile* with descriptors defined in MI-2.2.a, (MI-5.2) enrichment of the *AgentContext* with descriptors defined in MI-2.2.b, (MI-5.3) enrichment of the *AgentService* with descriptors defined in MI-3.2, (MI-5.4) enrichment of the *AgentModel* with descriptors defined in MI-4.5, and finally (MI-5.5) enrichment of the *AgentSmartThing* with the descriptors defined in MI-1.1.

After the step MI-5.5 has been performed, this stage provides data one instance of the IoA-OWL ontology as a result, with the exception of the *AgentSocial* component, because this will be enriched in stage MA-8.

3.1.6. Generation of the LAC for the Linked Open Agent (LOA) (MI-6)

This stage outlines the process performed to auto-generate the LAC and its six contractual sections. The seven steps of this stage are: (MI-6.1) auto-generation of the *AgentProfile* contractual section, (MI-6.2) auto-generation of the *AgentContext* contractual section, (MI-6.3) auto-generation of the *AgentSocial* contractual section – empty if the LOA must dynamically discover its own counterparts at macroscopic level or including the specific counterpart agents discovered and statically tested at stage MI-2 –, (MI-6.4) auto-generation of the *AgentService* contractual section, (MI-6.5) auto-generation of the *AgentModel* contractual section, (MI-6.6) auto-generation of the *AgentSmartThing* contractual section, and (MI-6.7) auto-generation of the general IoA contract.

All previously specified stages must be performed in an automated way by using the IoA-Building Tool—the specialized software framework for building and handling LOAs and their elements such as LAC, WAC and agent environment. Therefore, this stage does not require that developers make additional efforts. However, it is important to note that the success of the contract generation process will depend on the consistency of the data and URIs entered in the process of the semantic enrichment carried out in the previous stage, MI-5.

3.2. Method overview at macroscopic level

The second phase, illustrated at the bottom part of Fig. 8, is composed of five stages, of which one is a hybrid stage (MA-7), and the four remaining are automatic stages (MA-8, MA-9, MA-10, MA-11).

This phase is basically aimed at modeling issues related to the coordination of the *Agents of Things* and the creation of the global infrastructure and behavior modeling issues such as social networks and circles, communication, and collaboration. Therefore, it is at this stage where each requirement defined in the exploration process at IoA-interaction level such as discovering and selection of counterparts that must be introduced, in order to materialize them in terms of mechanisms which provide an intelligent behavior, autonomy, proactivity and collaboration capabilities to an IoA ecosystem.

3.2.1. Planning the agent partitions 2, 3 and 4 (MA-7)

In this stage, the instances of the *Agents of Coordination* type must be planned in order to generate their corresponding LACs in a similar way as at microscopic level. In addition, it is needed that this type of LOAs has associated a WAC which includes communications with *Agents of Things*. Therefore, for this stage to be accomplished, stages MI-2–MI-6 must be carried out but oriented towards covering tasks related to establishing communication and coordination processes with the required *Agents of Things*, *Agents of Data* and *Agents of People*.

3.2.2. Planning the global social IoA ecosystem (MA-8)

This stage is essentially oriented towards modeling the social issues such as interaction and collaboration associated with each of the previously-defined LOAs. This stage is completely executed by the LOA at runtime based on its WAC and LAC, and of reasoning processes requested to the IoA Central Infrastructure. The information recovered by the LOA is the input to update the *AgentSocial* contractual section of the LAC. This information is stored according to the *AgentSocial* component.

In this step to cover security restrictions, the work team is free to define specific static agent counterparts. Nonetheless, the LOA also has the ability to automatically discover the suitable counterparts at runtime based on its LAC and WAC.

Definition of the LOA's communication protocols (MA-8.1). Some of the most important contributions that FIPA has provided to the agent developer community are the Agent Communication Language (FIPA-ACL) and a set of FIPA communication protocols – Request, Query, Propose, Subscribe, and Contract-Net – from which it is possible to establish interaction between standard agents [65].

In this step, for each defined LOA, the communication protocols must firstly be defined to establish the communication channel between the specific LOAs and the end-users. Moreover, extra social communication protocols for collaborative agents must also be defined in order to achieve an efficient cooperation with LOAs that share its social circle in a public way on the IoA ecosystem. In this last case, we recommend the use of the Contract-Net protocol whose purpose is to carry out the agent collaboration process [66].

Modeling the social network participation (MA-8.2). It is important to define whether each LOA must operate collaborative or not. This aspect is determined through the collaborative degree (c_i) descriptor. If it is a collaborative LOA, the information from the LAC is automatically stored in the Central Infrastructure of IoA in order for the LOA to be discovered by other LOAs belonging to the IoA ecosystem.

Generically, a LOA is integrated into a generic social network called IoA. However, in the case of a LOA to be integrated into a different network, a new social network must be described. To describe the social networks (SN_i) in which a LOA participates, it is necessary to enter the social network *name* (n_i) descriptor. Other descriptors, such as *density* (d_i), *adjacency matrix* (M_i), and the centrality measurements – *InDegree* (I_i), *OutDegree* (O_i), *Degree* (D_i), *Betweeness* (B_i), *Closeness* (C_i) – are automatically initialized according to the relationships and social circles that are being gradually defined.

The modeling of social networks is not a trivial task, hence we suggest employing the tool called sociogram [67,68] in order to facilitate the representation, analysis and management of the nodes within each group belonging to the IoA ecosystem.

Modeling the social circles (MA-8.3). This step is complementary to the previous step (MA-8.2) and it is focused on introducing the main social circles for LOAs. This process can be realized according two cases. The first one includes LOAs which developers have to define their social circles manually. By contrast, the second one includes LOAs that can automatically discover every LOA. Therefore, the second case is a fully automatic step addressed by the IoA-Building Tool. This step does not need the intervention of developers.

Regardless of whether this process is manual or automatic, to describe the LOA's relationships (R_i) it is necessary to define the following descriptors: relationship *name* (n_i), *description* (dx_i), *weight* (w_i), *date of creation* (d_i), *source* ($From_i$), *destination* (To_i), and the *path* (P_i) of the relationship (R_i). Additionally, some constraints must be checked in order to create only consistent relationships: (i) a relationship with an external LOA can only be created if both, source and destination save a minimum of one topic of interest (or data of interest), (ii) a relationship can be created only if the LOA's source forms part of a social network in which the LOA's destination participates, and finally (iii) it is not possible to create reflexive relationships because these do not contribute to the social model.

As a recommendation, relationships should be introduced by using the sociogram built in the previous step. In this way, it is possible to analyze all groups where each LOA operates closely with its counterparts with whom it must establish communication and collaboration.

3.2.3. Planning the global behavior of the IoA ecosystem (MA-9)

The aim of this stage is to coordinate the individual behavior associated with each LOA defined in step MI-4, and to extend them towards a global intelligent behavior. Thus, the behavior modeled by LOAs at global level can cover the requirements of the IoA

infrastructure such as interaction and collaboration with external counterparts.

The main mechanisms focused on modeling the behavior of a LOA at ecosystem of IoA are emphasized in the following aspects:

- *Discovering of counterpart LOAs*. One of the capabilities of a LOA is that it employs the most suitable counterpart LOAs operating in the IoA ecosystem. For this purpose, each LOA has the capacity to request the execution of semantic reasoning processes from IoA Central Infrastructure. This process is automatic and matches the knowledge associated with the LACs of the LOAs of the IoA ecosystem stored in the form of RDF triplets.
- *Selecting the suitable counterpart LOAs*. Once the IoA Central Infrastructure provides a list of LOA counterparts, the LOA, using its own capabilities, applies a selection operator based on a specific criteria (e.g. real-time restrictions, social power) allowing filtering and selecting the LOAs that offer the functionality to the LOA which had executed the discovery process with the best response time.
- *Defining the fault-recovery policy for IoT-objects and agents*. The discovery process of LOAs counterparts provides data to update the LAC and WAC of the LOA that made a discovery request. However, these LOAs may be subject to failures once they have begun to operate. In order to cover this aspect, it is necessary to define a recovery policy from failures of objects or agents (e.g. discover new LOAs to replace the failed LOAs, run counterpart LOAs and use them while the failed LOAs are recovered). The developer has the decision to decide on the most appropriate fault-recovery policy to ensure that the system maintains the functionality with the available resources.

Modeling the IoA communication ecosystem (MA-9.1). This step is performed in order to define the model of communication of every single LOA to be coordinated according to the social model and the requirements at IoA-interaction level, as requested by the client. This is a process that is independent of the agent architecture adopted to model the IoA ecosystem. Therefore, it is advisable to model the global behavior through the use of UML Communication Diagram, which shows interactions between objects and/or parts using sequenced messages in a free-form arrangement [64].

Scheduling the real-time tasks of the application (MA-9.2). In this step a scheduling test is applied in order to determine the degree of feasibility that the collection of tasks required by the system can be scheduled, based on any dynamic real-time scheduler such as: Earliest Deadline First (EDF), Least Laxity First (LLF) [69], and Receding Horizon Control (RHC) [70]. Tasks that drives the global behavior of the application can be represented by using an UML Sequence Diagram [64] in which the real-time restrictions must be included.

3.2.4. Validation of the collection of Linked Agent Contracts (LACs) (MA-10)

This stage is basically aimed at performing the analysis of each of the elements (e.g. LACs, WACs and LOA-features) generated by the IoA-Building Tool in the previous stages at microscopic level. In this way, it is possible to identify and diagnose errors that could cause inconsistencies within the general LAC and the six contractual sections that compose the LOA LAC that form part of the IoA ecosystem. The validation process will be performed taking into account the following concerns:

Validation of the LOA credentials (STEP MA-10.1). Before the LAC which describes each LOA is generated, data regarding the Universal Identifier of Agent (UIA) must be verified in order to validate them and prevent the inclusion of danger LOAs in the IoA ecosystem. Each UIA is characterized as a single identifier provided by the IoA Central Infrastructure after making the respective request. It is important to note that the credential verification is made in both cases, before and after the creation of the LOA in order to be able to then audit which of the requested UIAs were actually used to create LOAs that are currently operating on the IoA ecosystem.

Validation of the global referential integrity of WACs (STEP MA-10.2). The possible inconsistencies that a LAC will have an impact on the content of the LOA WAC. Therefore, it is important to validate the WAC data associated with each LOA in order to prevent actions required by the LOA itself from being carried out due to the presence of inconsistent data in the workflow content. To cover this aspect, each LOA has a mechanism that allows it to adapt and recover from inconsistencies in its WAC. In general, the adopted mechanism is capable to recover in the face of the unavailability of IoT-objects and counterpart agents.

Validation of social links (STEP MA-10.3). This step validates the suitable definition of the social component of the LOAs that participate in the social ecosystem of the IoA application. Then, for each LOA, it is necessary to validate aspects such as, the consistency of the social networks that have arisen, the consistency regarding the topics and data of interest, the referential integrity of the relationships between the LOA with other ones, the consistence of the adjacency matrices affected by the inclusion or removing of LOAs into the social networks, and the consistency of the computation of the centrality measurements of each social network composing the IoA ecosystem. We recommend considering that a LOA with more social-power is one which has defined its social circle adequately, and not the agents that model a high quantity of relationships with little relevance.

Validation of the global referential integrity of LACs (STEP MA-10.4). The main reason that inconsistencies in the LACs arise is because one or more URIs entered in the semantic enrichment process has included a referenced entity that is not available to it. Consequently, it is very important to test the availability of every single URI integrated in the LACs. If there are inconsistencies it is possible that the mechanisms of reasoning and discovery of agents will be affected and therefore the system reduces its level of intelligence and autonomy, and in the worst case the system fails.

3.2.5. Execution of the Contract-Driven Model process (MA-11)

This stage is focused on starting the main computational resources of logical and physical type in order to determine the basic parameters related to the IoA ecosystem (e.g. settings of servers, hosts, end-points) required to execute the generation process of the LOAs in the infrastructure.

Before carrying out the steps which correspond to this stage, it is important to be sure that the LACs and WACs are well-defined and validated. Additionally, we recommend checking the correct operation of complementary resources, such as centralized (e.g. IoA Central Infrastructure) and distributed servers (e.g. IoA platform), accessing to public datasets, availability of LACs previously published, WAC accessibility by each agent, suitable operation of both external agents counterparts IoT-objects, IoT-resources and IoT physical network.

Preparing the IoA physical infrastructure (MA-11.1). Taking into account that an IoA application needs some physical computation resources to operate, in this step it is essential to prepare the physical IoT infrastructure that is required by LOAs and the IoA network. Among the main physical resources which must be prepared the main ones are: the IoT-objects with supporting LOAs, the IoT network communications and functionalities, the servers employed by the application (e.g. servers for managing web services, agents, datasets, ontologies and linked data), and any other physical resources that form part of the IoA system.

Preparing the IoA logical infrastructure (MA-11.2). In a similar way as in the step related to the preparation of the physical resources, the logical resources are also important to successfully

starting the IoA application. Some of the logical resources that must be prepared in this step are the following: the platform for supporting the selected agent model, the IoA-Building Tool for managing LOAs, LACs and WACs, the middleware for adding new LOA tasks, this is, integrating uncoupled agent artifacts at runtime, the web services that enable the accessing and communication with the physical objects, microservices invoked by the IoA application and the server for supporting persistence semantic knowledge and reasoning processes.

Executing process of auto-generation of the IoA platform (MA-11.3). This step is automated and consists of processing the LACs in order to automatically generate the required LOAs that integrate the specific application. In synthesis, this process creates the target application and therefore, after this step the IoA-Building Tool will have started. All LOAs built execute their corresponding tasks from which they execute all tasks supported by the LOA over the IoA ecosystem. Additionally, in this step the work team must test the functionality of each LOA at an implementation level and it must also check both the entities and mechanism modeled at a microscopic and macroscopic level.

4. Case study

This section presents the application of the proposed method to scale a small IoT-network – smart-office focused on provide comfort – towards an IoA infrastructure—smart-office focused on provide comfort based on energy saving policy. Main issues are detailed in this section.

4.1. Description of the IoT scenario

The specific IoT network has been developed using OpenHAB technology, which is an open source tool developed in Java focused on integrating everything in the IoT and providing a high level of interoperability in smart-home scenarios [71]. This framework has been developed by supporting a strict separation between the physical IoT-objects and the application by employing the notion of things and items. A thing represents any IoT-object that can be accessible to any application, while an item is a component that represents the functionality that is used by the application in terms of user interfaces or automation logic over things [71]. Each item has a state associated, whose change can be supervised through events.

There are 14 items that compose the studied scenario – 2 illumination sensors, 3 temperature sensors, 3 light-bulbs compatible with color and brightness changes, 2 window mechanisms, 2 shutter mechanisms, 1 motion sensor and 1 Heating, Ventilation, and Air Conditioning (HVAC) system – categorized into 7 types of virtual objects – dimmer, contact, color, switch, rollershutter, string and number—installed at the IoT scenario.

4.2. Definition of the IoA-scenario

The specific scenario of IoA must cover the following the functional requirements: (i) the temperature comfort in the office must normally be set before and during the working day, and also after the working day if any researcher continues working there, (ii) application of energy-saving policies in the system is strictly needed at a lower priority to the temperature comfort service, (iii) during the working day a comfortable light level must be maintained so as not to affect to the users who frequently use computers. Finally, (iv) the system operation must be based on meteorological predictions in order to provide comfort in relation to the future weather conditions.

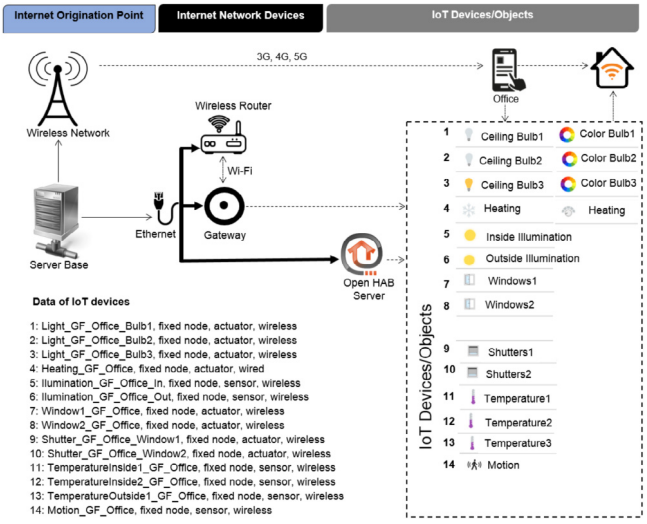

Fig. 9. IoT map of the specific physical IoT network.

4.3. Modeling at microscopic level

According to the scenario previously described, next in Fig. 9 the map of the physical IoT network is presented, including the components to compose the microscopic modeling (MI-1–MI-6) and its respective steps. This map depicts three types of components. The first (Internet access point) and the second (network devices) define the main resources associated with a global network of Internet and the resources to wirelessly connect in the specific network. On the other hand, the third component of the map (devices) corresponds to the IoT-objects connected in the specific scenario. In this case, 14 types of objects have been connected. This information is essential to carry out the microscopic phase where IoT-objects must be modeled as agents of Partition 1 (*Agent of Thing*).

Once the IoT map has been defined, the stages at microscopic level (MI-1–MI-6) have been carried out iteratively by every single IoT-object included in the IoT map of Fig. 9. Although the process has been performed by each IoT-object individually, in order to optimize the description of this process, next the description of the six first stages are presented including the information obtained by all IoT-objects.

4.3.1. MI-1: Planning the resources of the IoT-objects

The accomplishment of step MI-1.1 by each of the 14 objects that compose the IoT map has allowed us to gather the main descriptors associated with the resources of the IoT-objects, such as the identifier, data generated, actions performed, brand, location, geolocation, and time context based on the real environment where physical devices have been installed. This information is illustrated in Fig. 10-a.

On the other hand, because all objects are compatible with IoT interfaces implemented by means of OpenHAB framework, software agents can access these objects. Finally, after the completion of the step MI-1.3, a collection of datasets of IoT-objects are obtained. These data were stored in a CouchDB server.

4.3.2. MI-2: Discovering specialized public agents

Taking into account that this is the first application built by applying this proposal, step MI-2.1 could not be performed because public LOAs that could be integrated as part of the modeled application do not exist. Therefore, as the IoA ecosystem is currently void, the process continues to the next step (MI-2.2). Nonetheless,

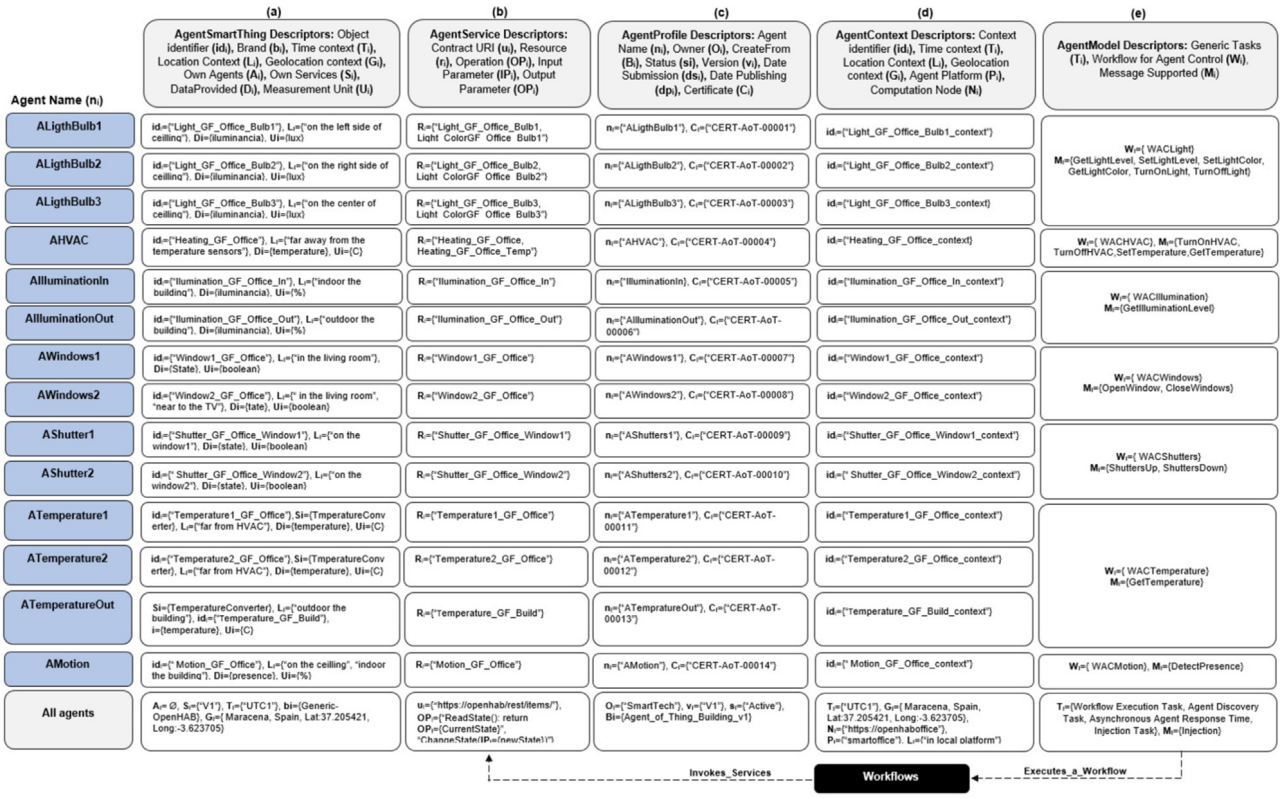

Fig. 10. Main descriptors of the (a) AgentSmartThing, (b) AgentService, (c) AgentProfile, (d) AgentContext and (e) AgentModel contractual sections of LACs that describes agents of Partition 1 modeled within the IoA ecosystem.

the execution of step MI-2.2 neither was successful. In summary, as it was not possible to discover generic agents that help us to solve tasks of comfort in smart-office systems, then this stage must be concluded without a specific result.

4.3.3. MI-3: Planning the internet of services ecosystem

As the OpenHAB framework was employed to model the IoT scenario, step MI-3.1 was easy to cover because this framework provides a REST API that allows the management of the action of the IoT-objects through RESTful services. This stage has also been applied iteratively by each IoT-object and a set of descriptors that depict the IoS ecosystem managed by the *Agents of Things* was obtained, as Fig. 10-b shows.

Further, it is important to highlight that after the completion of step MI-3.2 in which the datasets that describe the IoS ecosystem were published, a dataset with private description of RESTful services for the agents that operated within the smart-office system was published. Finally, in step MI-3.3 every service associated with each IoT-object interconnected within the specific scenario was tested. They were executed successfully.

4.3.4. MI-4: Planning the architecture and behavior of the LOA

Firstly, in step MI-4.1 we defined the *Agent of Thing* role as the main role for all agents modeled at this phase. The reactive agent architecture was defined as base to build each LOA modeled at microscopic level.

Secondly, in both steps MI-4.2 and MI-4.3 the descriptors related to the *AgentProfile* (Fig. 10-c) and *AgentContext* (Fig. 10-d) were defined iteratively for each *Agent of Thing* instance included by the IoT map after following steps MI-2.2.a and MI-2.2.b, respectively.

Thirdly, in step MI-4.4 we defined and built the specific WAC to each LOA following the schema proposed in Fig. 5.

Fourthly, we defined the agent tasks which drive the agent behavior following the LOA architecture illustrated in Fig. 2. In this way we complete step MI-4.5. It is important to emphasize that was not necessary to integrate additional tasks for any of the modeled LOAs because its logic of control was stored into their corresponding WAC. At end, the most remarkable information obtained after following this step are summarized in Fig. 10-e.

Step MI-4.5 was accomplished through the modeling of the agent behavior of each LOA instance by employing an UML State Diagram. To show an example of how using this diagram to depict the behavior of a reactive LOA, Fig. 11-a illustrates the main agent states and specialized tasks of the agent named AHVAC. This agent has associated a specific WAC focused on controlling the HVAC system to provide temperature comfort indoors at the relevant scenario.

In general terms, the agent *AHVAC* has three generic states – *initialized*, *active* and *waiting* – , three new states – *injecting*, *reinitializing* and *processing* – and three temporary tasks associated. The generic states are related to the normal operation of JADE agents, the new states *injecting* and *reinitializing* are executed when an injection process of external task is performed within the agent model and finally, the state *processing* is executed when an event is triggered (e.g. receiving message, activating scheduled task) and the agent must intercept it to maintain the suitable comfort level indoors at the office-automation system or to interact with the user.

Additionally, Fig. 11-b illustrates that the agent *AHVAC* is capable of reacting to temporal tasks that have been planned by the LOA scheduler (*Asynchronous Agent Response*, *Workflow Execution* and *Agent Discovery Tasks*) and also to react if the agent receives messages with content "*turnOnHVAC*", "*turnOffHVAC*", "*setTemperature*" or "*getTemperature*". These messages can be sent from external agents belonging to one of the categories of the partitions defined in the systematic method. For example, when the agent

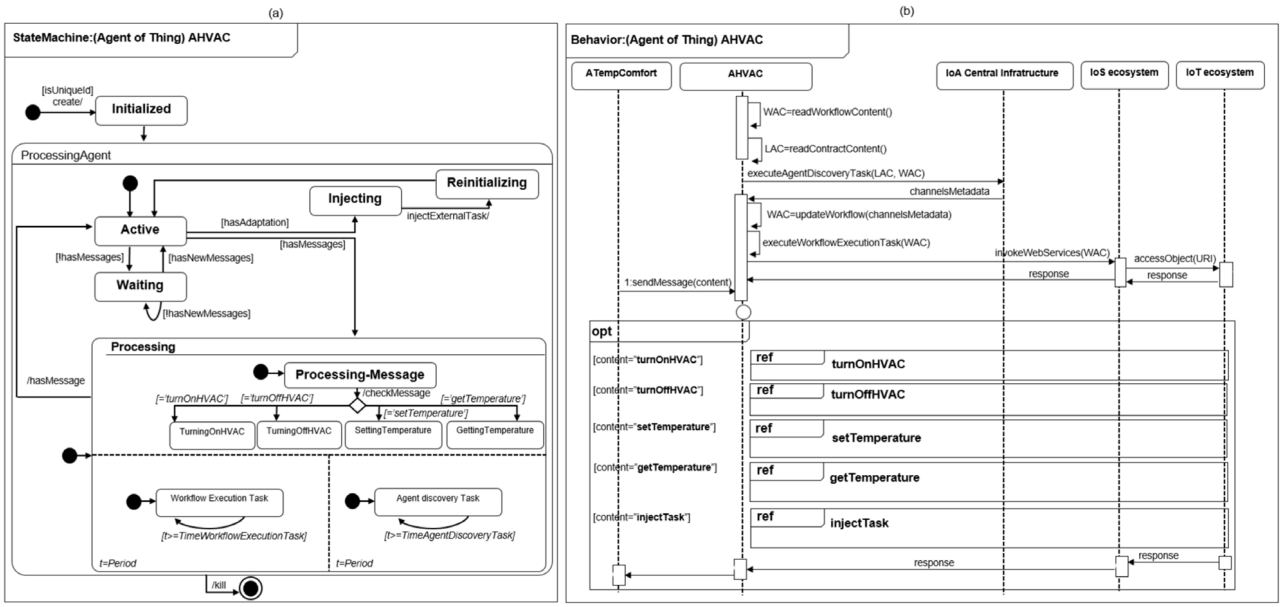

Fig. 11. Formal representation (a) UML state diagram and (b) UML activity diagram of the AHVAC agent instance.

AHVAC receives the message "*turnOnHVAC*", this agent executes the action "*turnOnHVAC*" which turns-on the HVAC system. This action is performed by means of the web service *Heating_GF_Office*. Remaining agents were modeled in a similar way to the agent *AHVAC*, but define their corresponding behaviors. That is how we complete step MI-4.6.

Finally, in step MI-4.7 two public datasets (parameters and real-time constraints) and one private dataset (behaviors/tasks) related to the agent model were published.

4.3.5. MI-5: Semantic enrichment of the LAC

The completion of stages MI-1, MI-2, MI-3 and MI-4 allowed for the instance of the IoA-OWL ontology to be supplied with data gathered from the procedure of definition of descriptors to perform the semantic enrichment process of the *AgentProfile*, *AgentSmartThing*, *AgentContext*, *AgentService* and *AgentModel* contractual sections. These contractual sections were enriched based on descriptors of Fig. 10-c, a, d, b. In addition, the list of tasks detailed by each agent of the IoA ecosystem listed in step MI-4.5 were also employed. This process was carried out by employing the *IoA-Building Tool*.

4.3.6. MI-6: Generation of the LAC for the LOA

This stage of automatic generation of contractual sections that comprise a LAC – with the exception of the social part which should be enriched in the macroscopic phase – has been carried out for each agent in a iterative way by using both the IoA-Building Tool and the descriptors specified in the process of semantic enrichment of the previous stage (MI-5). For example, in the specific case of the *ATemperature* LOA linked to the sensor *TemperatureInside1_GF_Office* the six contractual sections included in a single linked document were obtained, following the syntax of the general LAC illustrated in Fig. 3.

In addition, because all LOAs had to adopt a collaborative behavior, a replica of the descriptors stored into their LAC was also stored in a RDF graph over the *IoA Central Infrastructure*. This knowledge was used to perform reasoning processes at IoA global level. The LAC collection linked to the 14 objects that form part of the IoT network were modeled following the same structure adopted by the specific LOA *ATemperature*.

4.4. Modeling at macroscopic level

After the microscopic phase has been carried out, all agents in Partition 1, together with their corresponding contracts, have been modeled and generated. Nonetheless, modeling an IoA ecosystem implies the use of additional LOAs whose role is to coordinate the actions that LOAs of *Agents of Thing* type ought to execute in order to achieve the goal of the IoA platform. Following the proposed method, we defined 8 new agent instances according to the agent partitions already described.

4.4.1. MA-7: Planning the agent partitions 2, 3 and 4

Based on the agent partitions proposed by the method for modeling IoA applications, the following agent entities were defined in this stage: *Agents of Coordination* (ALightComfort, ATemperatureComfort and AGeneralComfort), *Agents of Data* (ADataFusion, AWeather and AHistoricize), and *Agents of People* (AMonitorOffice and AUserPreferences). These agents are organized as Fig. 12-a illustrates and their main functions are described as follows: (i) *ALightComfort* to control the suitable IoT-objects in order to search all type of light values to provide lighting comfort inside the smart-office, (ii) *ATemperatureComfort* similar LOA to *ALightComfort* but whose role is to search all type of temperature values, (iii) *AGeneralComfort* to coordinate both the *ALightComfort* and *ATemperatureComfort* LOAs to change to green comfort mode, and to change to extra comfort policy defined by the user, (iv) *AWeather* to obtain weather data from remote meteorological stations in order to obtain both the temperature and humidity conditions of a specific zone, (v) *ADataFusion* aimed at fusing data related to weather conditions inside and outside the specific scenario and fusing them in order to obtain more accurate information which is helpful to take decisions according to the comfort policy adopted by the system.

On the other hand, we have include two additional agents such as: (vi) *AHistorizice* to save information regarding the weather conditions and actions applied by the system, (vii) *AMonitorOffice:* to provide information regarding the comfort conditions in both text and graphical format of the specific scenario, and finally (viii) *AUserPreferences:* to manage end-user sets to control the configuration parameters by which the system will operate.

For each of the agents listed steps from MI-2 until MI-6 were carried out, in an iteratively similar way as with agents of partition

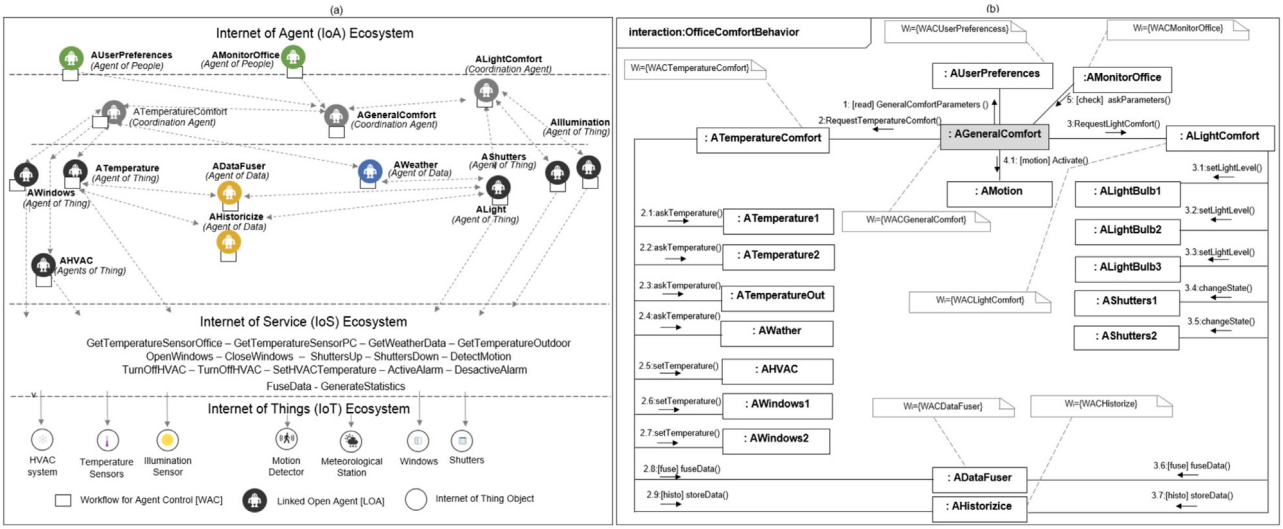

Fig. 12. Formal representation (a) the Internet of Agents ecosystem and (b) UML communication diagram to describe the global behavior of the smart-office system.

1, described in the application of phase 1. Regarding step MI-2, neither generic agents nor LOAs were discovered. Moreover, in step MI-3 no exiting services in the IoS ecosystem were discovered and, as a result, all services had to be created and new records in the dataset of RESTful services were published. In relation to step MI-4, the descriptors of the *AgentProfile*, *AgentContext* and *AgentModel* contractual sections, together with the *AgentService* descriptors specified in step MI-3, were defined. Agents were organized as shown in Fig. 12-b. Based on these descriptors, the enrichment process (step MI-5) and the contract generation (step MI-6) were carried out. It is important to highlight that each of the agents were defined following the similar mechanism of representation that is illustrated in Fig. 11-a and 11-b.

4.4.2. MA-8: Planning the global social IoA-ecosystem

Using the suitable communication protocol is key in order for agents to become social and collaborative. In step MA-8.1 the protocols for each category of agent were defined. That is to say, the use of the FIPA-Request-Protocol for agents of Partition 4 (*Agent of People*) because these only access to the setting of the user and they use these data to interact with agents of Partition 2. The same protocol is employed by agents of Partition 3 (*Agent of Data*) because they only act responsively when a request is made by agents of Partition 2 and 1. On the other hand, the social issue of both agents of Partition 1 and 2 is supported by the FIPA-Contract-Net protocol because they have to communicate with their counterparts according their WAC and employ the answer that works best in order for the system to provide a better quality of service that satisfies the needs of the end-user.

Regarding the topic and data of interest, the definition of social networks and social circles carried out in steps MA-8.2, these were defined by the LOAs themselves after the execution of reasoning processes, as illustrated in Fig. 12-b. This figure depicts the agents forming a social network (named *SNBuildingMaracena1*) and the relationship (named *collaborates*). Based on this information, the adjacency matrix (named *MBuildingMaracena1*) and the measurements of centrality of each actor (agent) were initialized by the application. In short, at the beginning, the leadership agent is the agent *ALightComfort* – degree=12, closeness=0.51, betweeness=0.43 –. Nonetheless, these values will change as new relationships which address the social ecosystem of the system are included within the social model. In conclusion, the information illustrated was used to supply the *AgentSocial* component that was previously generated but was empty until this stage.

4.4.3. MA-9: Planning the global behavior of the IoA ecosystem

In the first step (MA-9.1) of this stage we employed an UML Communication Diagram in which all interactions and messages between LOAs are defined. The information included in this diagram follows the alignments modeled within the defined social network. The general schema that drives the global behavior of the system is summarized in Fig. 12-b.

Although the diagram of this figure follows the bases of an UML communication diagram, we have left open ways which can be employed to update the model according to the changes of the social ecosystem that each agent manages. This adds dynamism to the agent model, which implies that agents can add new counterparts to optimize collaborative tasks and not limit them to static well-defined social issues. However, the adaptability of the system is not only achieved by updating the social ecosystem of each LOA, but also by reconfiguring the structural architecture of each agent through the deletion and injection of new agent artifacts aimed at adapting the behavior of agents as required.

4.4.4. MA-10: Validation the collection of LACs

The purpose of step MA-10.1 to validate agent credentials was realized by accessing the central repository managed by the IoA approach, in which all agents belonging to an ecosystem of this type are registered. This process was carried out successfully.

On the other hand, it was also checked that all *Agents of Things* and *Agents of Coordination* had their corresponding WAC linked to them. Each agent had its WAC associated with it. Therefore, the execution of step MA-10.2 was carried out without reporting any news.

Further, regarding step MA-10.3, all relationships that take part in the social networks managed by the system (*SmartComfortBuildingMaracena*) were totally consistent. Finally, the process of checking the referential integrity of LACs and their URIs was verified through the specialized tool Google to validate Linked Data [72] as the step MA-10.4 recommended. As a result, we obtained a collection of URIs which were consistent and well-structured.

4.4.5. MA-11: Execution of the Contract-Driven Model process

The IoA-Building Tool allowed us to perform the generation of LACs. From these contracts the corresponding LOAs that had formed the IoA ecosystem (smart office) were successfully and automatically created. The application was fully created and all agents operated according the target behavior, that is, the agents were capable of behaving self-adapting, collaborative and smartly. The final agent ecosystem behaved according to the modeled functional requirements.

5. Discussion

With the expansion of IoT technology, to date it is possible to access networks of objects providing real-time data to perform monitoring and control tasks in multiple application domains such as smart city, smart industry, and smart healthcare, and among others [1,73].

Data sensing and control actions carried out over IoT ecosystems are usually performed by using IoT-objects which integrate specific sensors and actuators. Such actions can be requested, parameterized and generally controlled through specific mobile applications or directly by users (using the IoT-object interface). However, IoT-objects are not able to act proactively to adapt to their environment by themselves. In this line, an IoA model has been proposed in this paper with the purpose of scaling IoT systems towards agentified smart IoT systems that enable proactive behavior over IoT ecosystems.

Next, an analysis of the main benefits of the proposed IoA model in terms of both functionality and its usage on real applications in areas related to smart cities and smart healthcare is discussed. In addition, we will also present the main and opportunities of the IoA model.

5.1. Potential general applications of the IoA model

In general, one of the main contributions that the IoA approach introduces to IoT is the addition of novel entities called LOAs allowing IoT-objects to be transformed from traditional passive objects into active ones. Accordingly, LOAs provide the capability of autonomously, proactively, socially, collaboratively and smartly acting over the IoT-object. Next, the main benefits of this approach and our specific model are described as follows:

- *Social and collaborative IoT applications (B_1).* The inclusion of a LAC together with the capacity of these agents to discover partners enable IoT-objects to be handled in a social and collaborative way. Subsequently, the collaboration among IoT-objects located at heterogeneous IoT ecosystems sharing their resources is just now possible under the control of the LOA itself. LOAs adopt a social model based on the dynamic discovery of agents as the smart objects do [6].
- *Smart IoT applications (B_2).* The discovery process helps LOAs to find the most suitable resources by a smart matching process based on the LAC descriptors meaning and not only in terms of its syntax. Therefore, information achieved from recovered LOAs is more accurate than most of the current IoT applications. In fact, the application of modeling rules on semantic searches allows retrieve data within a context that a syntactical search is not able to find. In addition, LOAs perform reasoning processes integrating a set of descriptors of different issues such as the included by the IoA-OWL ontology and not just context-based as the *Agent of Thing* model does [15].
- *Autonomous and proactive IoT applications (B_3).* The agentification of IoT-objects by means of LOAs, together with their corresponding LAC and WAC, enables the agent to autonomously control IoT-objects without human supervision. Hence, LOAs can also execute sensing and control actions over the IoT in a proactive way at any state of their life cycle. In short, LOAs take advantages of both proactive and autonomous features inherent to software agents in order to be able to achieve their goals by themselves, such as other agentified IoT ecosystems do [6].
- *More adaptable IoT applications (B_4).* LOAs apply a two-level adaptive mechanism to adjust their control goal. The first one is related to the ability of changing or substituting the WAC content of a LOA anytime at runtime, without penalizing the agent execution. This process is basically addressed by users. The second one depends on the current content of the WAC instance. Based on this content, the LOA can search the most suitable resources (e.g. external LOAs, IoT-objects) to satisfy the agent goals. In both cases, the LOA must then update its behavior from the selected recovered resources. Unlike most other agent models, the LOA model does provide agents with a mechanism to be personalized at runtime by the users as Yu et al. [13] suggest for adapting the IoT.
- *More flexible and distributed IoT applications (B_5).* SOA- and ROA-supporting by LOAs allows the decoupling of the agent functionalities and distribute them over the web or local servers. Thus, the modularity of agent artifacts can be exploited in order to provide a flexible mechanism facilitating the scalability of the LOA's behavior towards a better quality of service without changing the internal structure of LOAs [74]. In this way, LOAs become a more flexible component, since it distributes the processing actions (e.g. reasoning, access to sensors, and control of actuators, data processing and any other type of actions involving algorithms) over the web. In addition, new systems can reuse these components to be developed.
- *Uniform for accessing and controlling IoT applications (B_6).* The application of standards along for the development of IoA ecosystems is relevant to contributing to the improvement of interoperability. In this sense, LOAs support standards oriented to handle uniform mechanisms for accessing (REST interface), communicating (FIPA-ACL) and managing data (JSON-LD and RDF datasets) on IoT ecosystems. This facilitates the interoperability among heterogeneous agents and consequently the communication with objects connected to heterogeneous IoT ecosystems. Even though the use of all standards is important, FIPA communications are essential to interact with systems driven by heterogeneous models such IoT-a [75], agent of things [15], smart object [6] models, and even with other generic agent models such as JADE.
- *IoT Fault-Recovery systems (B_7).* The social, cognitive, autonomic and adaptive features together provide a new benefit over LOAs. This new benefit centers on entities capable of recovering from failures that occur in IoT environments, without the need for the user of supervising the adaptive processes that occur. This is a benefit acquired by dynamic systems such as the smart objects [6] or agent of things [15] models, both significant in the process of IoT agentification.
- *Additional security and privacy mechanisms that generic IoT (B_8).* Aspects associated with data security and privacy at IoA are paramount. To cover these issues, the LOA model manages secure IoA ecosystems in which agent-level communications are carried out based on agent credentials (security), and LACs with public and private descriptors (data privacy). From these mechanisms LOAs create their social circles and execute safe collaborative processes with the help of counterpart agents. However, the LOA model could be further optimized in terms of security and data privacy management with additional methods such as those listed in the QoS issues described in Section 2.

5.2. Potential general benefits of the IoA systematic method

Since this study is mainly focused on the formulation of a specialized systematic method for scaling IoT networks towards IoA networks, we have found additional benefits. These benefits, described below, are defined in terms of a software-oriented instrument that developers can exploit to create smart IoT systems by means of IoA.

- *Semi-Automatic generation of IoT applications (B_9).* The systematic method proposed in this paper enables developers to create IoA applications without the need of programming. Essentially, the method is focused on completing the semantic descriptions of the LAC by applying a semantic enrichment process, as well as, determining the agent behavior by the modeling of a specific WAC. The rigorous description of both documents and the use of semiautomatic tools will provide full IoA applications. As a result, developers can create agents more easily than the most current agent frameworks (e.g. JADE, Jason), where agents have to be fully programmed.
- *Systematic and easy to apply (B_{10}).* The proposed method is systematic because it clearly defines in details the guidelines and relevant design issues that developers must follow to create IoA applications. The developer does not have the degree of freedom that other, more generic methods provide to create a project based on IoT or agent approaches. Therefore, the developer is less likely to fall into ambiguities that could have negative effects on the development of IoA systems. Thus, IoA systems are then developed following the same guidelines and the same agent model and therefore, it is easier to carry out the integration process between systems of this type oriented to different domains.
- *Less complexity in the design of IoT applications (B_{11}).* Our method was designed in order to reduce the complexity of undertaking an IoA project, even if developers have not previously had any experience. To achieve that, the method establishes the set of steps developers should follow to build rich, smart, autonomous collaborative IoA applications from an IoT infrastructure. Unlike a general agent methodology, this method is specifically oriented to work with IoT ecosystems. Then, the stages are addressed to create specific LOA agents integrated with IoT-objects. As a result, the learning curve required is far minor. It is not our intention to position our method as a unique alternative to create intelligent applications of IoT, but rather to provide an easy-to-use alternative.

5.3. Main limitations of the IoA model

The development of IoA application based on the proposed method can have some limitations to be considered. Next, some of these limitations are detailed:

- *Model based on specific elements (L_1).* Although the agents in an IoA domain can be implemented by any agent framework to create the LOAs, they should be close related to the existence of both LAC and WAC document. This means that in order for LOAs to operate consistently over the IoA ecosystem, further efforts must be made to define the WAC of each agent. In this sense, it is required the development of a tool specialized in the construction of WACs templates for specific objects that support different levels of control over IoT ecosystems. Thus, developers can use simple or complex WACs according to their needs.
- *Requiring more usability and accessibility interface (L_2).* The success of the proposed IoA approach largely depends on the semantic descriptors that users enter to carry out the agents generation. In this line, the prototype of IoA-Building Tool provides a simple graphical interface based on GUI widgets to realize the process of semantic enrichment of LOAs. However, such interface could be improved to make user-dependent aspects and interaction with the easy and fun application. Moreover, it is recommended that the new interaction interface is designed and created in compliance with the current web accessibility and usability standards proposed by the W3C Consortium.
- *Federation of the semantic knowledge (L_3).* LOAs and their respective LACs are stored in a distributed way based on agent platforms. However, the semantic knowledge – used by the IoA Central Infrastructure to provide semantic reasoning – is currently stored in RDF graph format in a centralized manner. In order to also extend the concept of distribution to the field of the semantic knowledge persistence, model federated RDF graphs could be possible to allow the knowledge distribution and to minimize the complexity of the semantic reasoning processes requested by LOAs.

5.4. Potential specific usages of the IoA in real IoT applications

With the objective of defining the scope of the proposed mechanisms in terms of real applications at different fields where the IoT is currently used, an analysis of the potential benefits of these mechanisms has been presented. This analysis covers two of the three specific areas proposed by Borgia et al. [1] such as smart city and smart healthcare. These specific areas were also analyzed by Pico-Valencia et al. [73] in terms of applications for the IoA.

5.4.1. IoA for smart city

A Smart City system is aimed at increasing the citizens' quality of life through the optimization of both public services (e.g. transport, lighting, surveillance) and facilities (e.g. accessibility, sanitation, parks) [76,77]. Main general sub-areas covered by this field include: public safety and monitoring, smart home/building, smart grids, and smart mobility and tourism [1]. Under these sub-areas, a wide range of traditional public services has been optimized through the use of IoT technologies. Some of them are the following: smart lighting systems, environmental quality monitoring, smart recycling and waste management, traffic congestion, smart parking, city energy consumptions, automation of public buildings, etc. [77].

Regardless of the technology used, many of today's smart city systems still face problems related to issues such as: heterogeneity management, interoperability, scalability and integration of heterogeneous systems, fault tolerance, system maintenance, collaboration, reliability, and re-configuration [76]. Dealing with these problems, smart cities systems can scale up to a higher intelligence and the public data sensing tasks can be optimized in terms of trust and quality of both services and information [78].

Next, we present the most remarkable benefits of our IoA model to develop the systems for smart cities. The analysis presented in Table 1 is based on the potential general benefits of the IoA approach listed at the top of this section.

5.4.2. IoA for smart healthcare

Healthcare systems enable the management of electronic medical information associated with patients to provide early diagnosis and monitor, at real-time and under secure conditions, the physiological conditions (e.g. heart beat, blood pressure, temperature) of patients in different contexts (e.g. in-hospital, nursing homes and in-home) in order to improve their quality of life [79–81].

The Borgia's study emphasizes two general sub-areas of application in which the IoT technologies are being employed in healthcare such as medical and healthcare, and independent living [1]. In these sub-areas, services focused on patient care have been developed. Some of them are briefly summarized as follows: health conditions monitoring, assistance for patients with chronic diseases, elderly care, fitness programs, and so on. To provide these services, healthcare systems must seriously support aspects such as: secure communications, reliable actions, and data management with high levels of trust and privacy. Furthermore, other also important aspects to be managed by these systems are related to the following: interoperability, intelligence, and real-time [81,82].

Next, we present the most remarkable benefits of our IoA model to develop systems for smart healthcare following its specific issues. These benefits are detailed in Table 2.

Table 1
Some benefits of the IoA model for implementing smart city systems. Benefits (B) of the IoA model: B_1=social and collaborative, B_2=smart, B_3=autonomous and proactive, B_4=adaptable, B_5=flexible and distributed, B_6=uniform accessing and controlling, B_7=fault-recovery, B_8=handling security and privacy.

Remarkable aspects	B	Some specific behaviors
Interoperability. Interconnecting services at intra- and inter-smart city systems	5, 6	* Drug control police agents with capability of establishing inter-smart city communication in order to monitor the border of two countries where illegal drug trafficking has taken place. * Public building care control agents employing data of external systems to close windows and sheds on public building when there is a heavy rainfall, high levels of air pollution or when particles from natural phenomena such as volcanic eruptions fall.
Availability. Accessing services by citizens whenever possible even if failures occur	3, 4	* Metropolitan surveillance agents with the ability of proactively sending alerts to the nearest police station regarding the danger of public spaces caused by citizens and even to report if damages have been provoked by criminals. * Smart lighting agents that are automatically reconfigured by themselves in order to adapt their behavior according to the weather conditions and seasons offered by the public meteorological systems installed on the specific area.
Collaboration. Sharing complementary services at intra- and inter-smart city systems	1, 2, 3	* National security agents can share information on new wanted criminals in the surveillance systems of the nearest smart cities and traffic light networks. Then, the capture criminals will be collaboratively by using objects installed on public places (e.g. roads). * Irrigation agents of city parks can establish common irrigation strategies based on city weather forecasts and prediction of natural phenomena (e.g. volcanic eruption) in the area.
Cognition and reliability. Managing resources offered by cities coherently, consistently and in an optimized way	2, 3, 7	* Traffic agents who analyze the road conditions and traffic congestion information to recommend safer and more efficient routes on the roads and streets panels of nearby cities from a point of origin to a specific destination. * Parking agents who monitor the use of parking in a specific city according to restrictions (e.g. time limits and type of parking lot) and who estimate new parking spaces so that waiting customers can optimize the time needed to obtain a parking space.
Security and privacy. Handling secure communications protecting citizens' data	8	* Agents of attention to the citizen who play a role o intermediary among the citizens and the smart city services. These agents are capable to retrieve only public data of interest according their context, and status (permanent resident or visitor) to they can plan specific pathways. * Agents of criminal intelligence monitoring and alerting the irregular access to smart city sensors and agent communications in order to prognostic suspicious behaviors.

Table 2
Some benefits of the IoA model for smart healthcare systems. Benefits (B) of the IoA model: B_1=social and collaborative, B_2=smart, B_3=autonomous and proactive, B_4=adaptable, B_5=flexible and distributed, B_6=uniform accessing and controlling, B_7=fault-recovery, B_8=handling security and privacy.

Remarkable aspects	B	Some specific behaviors
Interoperability. Interconnecting heterogeneous body sensors and services of hospitals and ambient assisting living scenarios	5, 6	* Home environment agents that can provide comfort for patients based on data obtained from wearables of different manufactures. These agents are capable to take care of patients' health taking into account their health conditions, that is, for example no cold neither dust for patients with coughs or asthma. * First aids agents that may be able of constantly monitoring data of vital signs of elderly patients that live alone. In this way, these agents can analyze their daily activity, possible actions that may damage its integrity (e.g. falls, heart failure), state of mind, compliance with the treatment prescribed by the doctor, and the evolution in the state of their health.
Availability. Accessing medical services and data whenever required	3, 4	* Meta-monitor agents employed by machines (e.g. drug dispenser) providing health services so as to alert in a proactive way when they are not available to carry out the corrective maintenance actions and thus, to avoid risks on the patient life. * Medical appointment negotiation agents can deal with appointment manager agents in hospitals of a city or country to obtain a medical appointment in a specialty area.
Collaboration. Sharing data among intra- and inter-hospital	1, 2, 3	* Agents experts on diagnosis of various hospitals can share data among them when a patient presents a clinical condition that is difficult to diagnose so that the doctor can give a more accurate diagnosis based on the experience and knowledge of his virtual colleagues. * Agents of recommendation that are based on environmental conditions (e.g. air pollution, UV index, weather, traffic, pollen in the environment) provided with by external systems is capable of suggesting patients with health problems (e.g. allergies, asthma, flu, skin cancer) when and where to realize the outdoor activities.
Cognition and reliability. Managing clinical data of patients coherently, consistently and accurately	2, 3, 7	* Expert diagnostic agents with the ability of supporting doctors in the most accurate diagnosis of patients' condition (e.g. diabetic) who suffer from a chronic disease on the basis of their complete medical history and their normal daily routines and dietary habits. * Nutritionist agents who can connect with the smart fridge and to the nearest grocery stores to buy on-line and also provide food to suggest obese people a healthy diet. In this way, patients can prevent cholesterol levels from rising and to possibly develop other diseases.
Security and privacy. Handling secure communications and protecting data of patients	8	* Agent for monitoring the patients' physiological conditions can include a barrier oriented to control the authorized access and encrypt confidential data measured from wearables wears by patients and shared with external systems. * Consent informed agents who can be intermediate among agents of people (patients) and agents of things (wearables) so that data in healthcare can be shared with the consent of users in order to avoid the indiscriminate access to, dissemination and misuse of data.

6. Conclusions

The integration process between the IoT and agent technology is a current approach that has arisen as an alternative to introduce an intelligent component within generic IoT networks. This is the main reason why not only specialized mechanisms to manage the IoT resources smartly are required, but it is also essential to define formal and systematic methods that drive the process of building applications based on the IoA approach semi-automatically. This work proposes a specialized method based on agile methodologies that are capable of covering the modeling of IoT-objects as agents. Moreover, the aforementioned method covers the modeling of mechanisms of coordination and intelligence of agents through the use of semantic contracts, which are constructed based on a combination of descriptors, which represent the basic input to

execute the semantic reasoning processes which leads to decision-making.

Because the proposal is an agent-based method specialized in the integration of IoT networks and software agents, but driven by a semantic approach, we have employed the OpenHAB framework to model the IoT network of devices. Also, to cover the agent issue based on Linked Agent Contracts (LACs), we have developed a specific software tool called IoA-Building Tool. This method was perfectly compatible with the integration of both IoT and agent technologies. Therefore, the method can be employed to build IoA systems where entities such as *Linked Open Agent (LOA) of Thing* are the main processing unit. However, it is important to note that we have applied our proposal to integrate non-large IoT ecosystems, but only to improve the semantic interoperability of small IoT networks such as a smart-office.

Regarding future research, we believe that this area is still a field in the process of maturity and accordingly, based on the systematic method, a formal methodology and its corresponding CASE (Computer Aided Software Engineering) tool could be created in order to satisfy the entire process of developing IoA systems in a automatic way including: (i) scaling generic MAS applications towards applications based on the IoA approach based on LACs, Agent Control Workflows (WACs) and LOAs, (ii) integrating IoA ecosystems, (iii) complete testing process and (iv) issues related to implementation. That is to say, to include formal stages of the Software Engineering process. In addition, we are working to test our approach in developing IoA systems on a wide scale in domains such as smart communities and cities.

Acknowledgments

This study was funded by the Ecuadorian Ministry of Higher Education, Science, Technology and Innovation (SENESCYT). The authors would also like to thank the Research Center of the Pontifical Catholic University of Ecuador, Esmeraldas (PUCESE).

References

[1] E. Borgia, The Internet of Things vision: Key features, applications and open issues, Elsevier Comput. Commun. 54 (2014) 1–31.
[2] S. Li, S. Da Xu, Li, S. Zhao, The Internet of Things: A survey, Springer Inf. Syst. Front. 17 (2) (2015) 243–259.
[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of Things: A survey on enabling technologies, protocols, and applications, IEEE Commun. Surv. Tutor. 17 (4) (2015) 2347–2376.
[4] L. Atzori, A. Iera, G. Morabito, The Internet of Things: A survey, Elsevier Comput. Netw. 54 (15) (2010) 2787–2805.
[5] A.M. Mzahm, M.S. Ahmad, A.Y. Tang, A. Ahmad, Towards a design model for things in Agents of Things, in: Proceedings of the International Conference on Internet of things and Cloud Computing, ICC'16, New York, NY, USA, 2016, article 41, pp. 1–5, https://doi.org/10.1145/2896387.2896426.
[6] G. Fortino, A. Guerrieri, W. Russo, Agent-oriented smart objects development, in: Proceedings of the IEEE 2012 16th International Conference on Computer Supported Cooperative Work in Design, CSCWD, Wuhan, China, 2012, pp. 907–912, https://doi.org/10.1109/CSCWD.2012.6221929.
[7] N.J. Kaminski, M. Murphy, N. Marchetti, Agent-based modeling of an IoT network, in: 2016 IEEE International Symposium on Systems Engineering, ISSE, Edinburgh, UK, 2016, pp. 1–7, https://doi.org/10.1109/SysEng.2016.7753151.
[8] X. Xu, N. Bessis, J. Cao, An autonomic agent trust model for IoT systems, Elsevier Procedia Comput. Sci. 21 (2013) 107–113.
[9] J.C. Nieves, D. Andrade, E. Guerrero, MAIoT-An IoT architecture with reasoning and dialogue capability, in: Springer Applications for Future Internet, 2017, pp. 109–113.
[10] P. Leong, L. Lu, Multiagent web for the Internet of Things, in: 2014 IEEE International Conference on Information Science and Applications, ICISA, Seoul, South Korea, 2014, pp. 1–4, https://doi.org/10.1109/ICISA.2014.6847432.
[11] H. Al-Sakran, Intelligent traffic information system based on integration of Internet of Things and Agent Technology, Int. J. Adv. Comput. Sci. Appl. 5 (2) (2015) 37–43.
[12] P. Pico-Valencia, J.A. Holgado-Terriza, Semantic agent contracts for Internet of Agents, in: 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops, WIW, Omaha, NE, USA, 2016, pp. 76–79, https://doi.org/10.1109/WIW.2016.033.
[13] H. Yu, Z. Shen, C. Leung, From Internet of Things to Internet of Agents, in: 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, Beijing, 2013, pp. 1054–1057, https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.179.
[14] G. Kortuem, F. Kawsar, V. Sundramoorthy, D. Fitton, Smart objects as building blocks for the Internet of Things, IEEE Internet Comput. 14 (1) (2010) 44–51.
[15] A.M. Mzahm, M.S. Ahmad, A.Y. Tang, Agents of Things, AoT: An intelligent operational concept of the Internet of Things, IoT, in: 2013 13th International Conference on Intelligent Systems Design and Applications, Bangi, Malaysia, 2013, pp. 159–164, https://doi.org/10.1109/ISDA.2013.6920728.
[16] T. Leppänen, J. Riekki, M. Liu, E. Harjula, T. Ojala, Mobile agents-based smart objects for the Internet of Things, in: Springer Internet of Things Based on Smart Objects, 2014, pp. 29–48.
[17] G. Fortino, Agents meet the IoT: Toward ecosystems of networked smart objects, IEEE Syst., Man, Cybern. Mag. 2 (2) (2016) 43–47.
[18] C. Alexakos, A.P. Kalogeras, Internet of Things integration to a multi agent system based manufacturing environment, in: Proc. of the IEEE 2015 20th Conference on Emerging Technologies & Factory Automation, ETFA, 2015, pp. 1–8, http://dx.doi.org/10.1109/ETFA.2015.7301415.
[19] Q.N.N. Tran, G.C. Low, Comparison of ten agent-oriented methodologies, in: Agent-Oriented Methodologies, IGI Global, 2005, pp. 341–367.
[20] O. Shehory, A. Sturm, The landscape of agent-oriented methodologies, in: O. Shehory, A. Sturm (Eds.), Agent-Oriented Software Engineering, Springer, Berlin, Heidelberg, 2014, pp. 137–154, http://dx.doi.org/10.1007/978-3-642-54432-3_7.
[21] O.Z. Akbari, A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance, Int. J. Comput. Eng. Res. 1 (2) (2010) 14–28.
[22] C. Bizer, T. Heath, T. Berners-Lee, Linked data-the story so far, IGI Glob. Int. J. Semant. Web Inf. Syst. 5 (3) (2009) 205–227.
[23] T. Heath, C. Bizer, Linked data: Evolving the web into a global data space, Morgan Claypool Publ. Synth. Lect. Semant. Web: Theory Technol. 1 (1) (2011) 1–136.
[24] M. Wooldridge, N.R. Jennings, Agent theories, architectures, and languages: A survey, in: Wooldridge M.J., Jennings N.R. (Eds.), Intelligent Agents. ATAL 1994. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol. 890, Springer, Berlin, Heidelberg, 1995, pp. 1–39, http://dx.doi.org/10.1007/3-540-58855-8_1.
[25] S. Russel, P. Norvig, Artificial Intelligence: A Modern Approach, third ed., Prentice Hall Press, New York, 2009, ISBN-13: 978-0136042594, ISBN-10:0136042597.
[26] L. Braubach, A. Pokahr, W. Lamersdorf, Jadex: A short overview, in: Main Conference Net.ObjectDays 2004, Erfurt, Germany, 2004, pp. 195–207.
[27] T. Balke, N. Gilbert, How do agents make decisions? A survey, J. Artif. Soc. Soc. Simul. 17 (4) (2014) 13.
[28] P. Pico-Valencia, J.A. Holgado-Terriza, ADELE: A middleware for supporting the evolution of multi-agents systems based on a metaprogramming approach, in: Springer International Conference on Practical Applications of Agents and Multi-Agent Systems, Seville, Spain, 2016, pp. 297–310, https://doi.org/10.1007/978-3-319-40159-1_25.
[29] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, JADE: A software framework for developing multi-agent applications. Lessons learned, Elsevier Inf. Softw. Technol. 50 (1) (2008) 10–21.
[30] JSON-LD-Community-Group, JSON for Linking Data, 2014, URL http://json-ld.org/ (Accessed 3 January 2017).
[31] M.B. Hoy, If this then that: An introduction to automated task services, Taylor Francis Med. Ref. Serv. Q. 34 (1) (2015) 98–103.
[32] V. Issarny, N. Georgantas, S. Hachem, A. Zarras, P. Vassiliadist, M. Autili, M.A. Gerosa, A.B. Hamida, Service-oriented middleware for the future internet: State of the art and research directions, Springer J. Internet Serv. Appl. 2 (1) (2011) 23–45.
[33] R. Khan, S.U. Khan, R. Zaheer, S. Khan, Future internet: The Internet of Things architecture, possible applications and key challenges, in: 2012 10th International Conference on Frontiers of Information Technology, Islamabad, India, 2012, pp. 257–260, https://doi.org/10.1109/FIT.2012.53.
[34] J. Soriano, C. Heitz, H.-P. Hutter, R. Fernández, J.J. Hierro, J. Vogel, A. Edmonds, T.M. Bohnert, Internet of Services, in: Bertin E., Crespi N., Magedanz T. (Eds.), Evolution of Telecommunication Services. Lecture Notes in Computer Science, vol. 7768, Springer, Berlin, Heidelberg, 2013, pp. 283–325, http://dx.doi.org/10.1007/978-3-642-41569-2_14.
[35] M.A. Razzaque, M. Milojevic-Jevric, A. Palade, S. Clarke, Middleware for Internet of Things: A survey, IEEE Internet Things J. 3 (1) (2016) 70–95.
[36] T. Erl, Soa: Principles of Service Design, 1st edition, Prentice Hall Press, New York, ISBN: 978-0132344821, 2008.
[37] X. Guo, J. Shen, Z. Yin, On software development based on SOA and ROA, in: 2010 Chinese Control and Decision Conference, Xuzhou, China, 2010, pp. 1032–1035, https://doi.org/10.1109/CCDC.2010.5498067.
[38] P. Pico-Valencia, J.A. Holgado-Terriza, An agent middleware for supporting ecosystems of heterogeneous web services, Elsevier Procedia Comput. Sci. 94 (2016) 121–128.

[39] H. Hamad, M. Saad, R. Abed, Performance evaluation of restful web services for mobile devices, Int. Arab J. e-Technol. 1 (3) (2010) 72–78.

[40] D. Xu, B. Fang, H. Li, An abstract communication service interface of DPWS on embedded device in industrial automation, in: 2016 Chinese Control and Decision Conference, CCDC, Yinchuan, China, 2016, pp. 4695–4699, https://doi.org/10.1109/CCDC.2016.7531832.

[41] S. Rodríguez, J.A. Holgado, A home-automation platform towards ubiquitous spaces based on a decentralized p2p architecture, in: J.M. Corchado, S. Rodríguez, J. Llinas, J.M. Molina (Eds.), International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008). Advances in Soft Computing, vol. 50, Springer, Berlin, Heidelberg, 2009, pp. 304–308, http://dx.doi.org/10.1007/978-3-540-85863-8_36.

[42] J. Lee, S.J. Lee, P.F. Wang, A framework for composing soap, non-soap and non-web services, IEEE Trans. Serv. Comput. 8 (2) (2015) 1939-1374.

[43] J. Huang, D.J. Abadi, K. Ren, Scalable SPARQL querying of large RDF graphs, in: Proceedings of the VLDB Endowment, vol. 4, Seattle, Washington, 2011, pp. 1123–1134 (11), http://www.vldb.org/pvldb/vol4/p1123-huang.pdf.

[44] F. Al-Turjman, Price-based data delivery framework for dynamic and pervasive IoT, Elsevier Pervasive Mob. Comput. 42 (2017) 299–316.

[45] G.T. Singh, F.M. Al-Turjman, Learning data delivery paths in QoI-aware information-centric sensor networks, IEEE Internet Things J. 3 (4) (2016) 572–580.

[46] M.Z. Hasan, F. Al-Turjman, H. Al-Rizzo, Optimized multi-constrained quality-of-service multipath routing approach for multimedia sensor networks, IEEE Sens. J. 17 (7) (2017) 2298–2309.

[47] L. Li, S. Li, S. Zhao, QoS-aware scheduling of services-oriented Internet of Things, IEEE Trans. Ind. Inf. 10 (2) (2014) 1497–1505.

[48] M. Alreshoodi, J. Woods, Survey on QoE\QoS correlation models for multimedia services, Int. J. Distrib. Parallel Syst. (IJDPS) 4 (3) (2013) 1–20, http://dx.doi.org/10.5121/ijdps.2013.4305.

[49] D. Chen, P.K. Varshney, QoS support in wireless sensor networks: A survey, in: International Conference on Wireless Networks, ICWN '04, Las Vegas, Nevada, USA, 2004, pp. 1–7.

[50] S. Ran, A model for web services discovery with QoS, ACM Sigecom Exch. 4 (1) (2003) 1–10.

[51] Y. Blanco-Fernández, J.J. Pazos-Arias, A. Gil-Solla, M. Ramos-Cabrer, B. Barragáns-Martinéz, M. López-Nores, J. García-Duque, A. Fernández-Vilas, R.P. Díaz-Redondo, AVATAR: An advanced multi-agent recommender system of personalized TV contents by semantic reasoning, in: X. Zhou, S. Su, M.P. Papazoglou, M.E. Orlowska, K. Jeffery (Eds.), Web Information Systems – WISE 2004. WISE 2004. Lecture Notes in Computer Science, vol. 3306, Springer, Berlin, Heidelberg, 2004, pp. 415–421, http://dx.doi.org/10.1007/978-3-540-30480-7_43.

[52] K. Chmiel, M. Gawinecki, P. Kaczmarek, M. Szymczak, M. Paprzycki, Efficiency of JADE agent platform, Hindawi Sci. Program. 13 (2) (2005) 159–172.

[53] P. Dadhich, K. Dutta, M. Govil, Security issues in mobile agents, Int. J. Comput. Appl. 11 (4) (2010) 1–7.

[54] F.A. Alaba, M. Othman, I.A.T. Hashem, F. Alotaibi, Internet of Things security: A survey, Elsevier J. Netw. Comput. Appl. 88 (2017) 10–28.

[55] G. White, V. Nallur, S. Clarke, Quality of service approaches in IoT: A systematic mapping, Elsevier J. Syst. Softw. 132 (2017) 186–203.

[56] Y.L. Hu, Y.Y. Cho, W.B. Su, D.S. Wei, Y. Huang, J.L. Chen, Y. Chen, S.Y. Kuo, A programming framework for implementing fault-tolerant mechanism in IoT applications, in: G. Wang, A. Zomaya, G. Martinez, K. Li (Eds.), Algorithms and Architectures for Parallel Processing. ICA3PP 2015. Lecture Notes in Computer Science, vol. 9530, Springer, Cham, 2015, pp. 771–784, http://dx.doi.org/10.1007/978-3-319-27137-8_56.

[57] E.M. Schön, J. Thomaschewski, M.J. Escalona, Agile requirements engineering: A systematic literature review, Elsevier Comput. Stand. Interfaces 49 (2017) 79–91.

[58] K.S. Rubin, Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley, ISBN: 978-0-13-704329-3, 2012.

[59] R. Thomas, Architectural styles and the design of network-based software architectures, Ph.D. thesis, University of California, 2000.

[60] A. Spillner, T. Linz, H. Schaefer, Software Testing Foundations: A Study Guide for the Certified Tester Exam, fourth ed., Rocky Nook, Inc., 2014, ISBN: 987-1-937538-42-2.

[61] M. Jaffar-ur, F. Jabeen, A. Bertolino, A. Polini, Testing software components for integration: A survey of issues and techniques, J. Softw. Test., Verif. Reliab. 17 (2) (2007) 95–133, http://dx.doi.org/10.1002/stvr.357.

[62] J. Thönes, Microservices, IEEE Softw. 32 (1) (2015) http://dx.doi.org/10.1109/MS.2015.11, 116-116.

[63] V.D. Le, M.M. Neff, R.V. Stewart, R. Kelley, E. Fritzinger, S.M. Dascalu, F.C. Harris, Microservice-based architecture for the NRDC, in: Proc. of the IEEE 2015 13th International Conference on Industrial Informatics (INDIN), Cambridge, UK, 2015, pp. 1659–1664.

[64] OMG, OMG Unified Modeling Language TM (OMG UML) V 2.5, 2015, URL http://www.omg.org/spec/UML/25/PDF/ (Accessed 3 January 2017).

[65] FIPA, Interaction Protocol Specifications, 2012, URL http://www.fipa.org/repository/ips.php3 (Accessed 3 October 2016).

[66] M. Wooldridge, An Introduction to Multiagent Systems, second ed., John Wiley & Sons, ISBN: 978-0-470-51946-2, 2009.

[67] A. Drahota, A. Dewey, The sociogram: A useful tool in the analysis of focus groups, Nurs. Res. 57 (4) (2018) 293–297.

[68] J. Scott, Social Network Analysis, third ed., Sage, ISBN: 978-1-4462-0903-5, 2012.

[69] V. Shinde, S.C. Biday, Comparison of real time task scheduling algorithms, Int. J. Comput. Appl. 158 (6) (2017) 37–41, http://dx.doi.org/10.5120/ijca2017912832.

[70] A. Subramanian, M.J. Garcia, D.S. Callaway, K. Poolla, P. Varaiya, Real-time scheduling of distributed resources, IEEE Trans. Smart Grid 4 (4) (2013) 2122–2130.

[71] OpenHab-Fundation, OpenHAB 2 Documentation, 2016, URL http://docs.openhab.org/ (Accessed 13 January 2017).

[72] Google, Google Structured Data, 2017, URL https://search.google.com/structured-data/testing-tool? (Accessed 3 January 2017).

[73] P. Pico-Valencia, J.A. Holgado-Terriza, D. Herrera-Sánchez, J. Sampietro, Towards the Internet of Agents: An analysis of the Internet of Things from the intelligence and autonomy perspective, Ing. Investig. 38 (1) (2018) 121–129.

[74] D.I. Tapia, S. Rodríguez, J. Bajo, J.M. Corchado, FUSION@, a SOA-based multi-agent architecture, in: J.M. Corchado, S. Rodríguez, J. Llinas, J.M. Molina (Eds.), International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008). Advances in Soft Computing, vol. 50, Springer, Berlin, Heidelberg, 2009, pp. 99–107, http://dx.doi.org/10.1007/978-3-540-85863-8_13.

[75] F. Carlier, V. Renault, IoT-a, embedded agents for smart Internet of Things. Application on a display wall, in: 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops, WIW, Omaha, NE, USA, 2016, pp. 80–83, https://doi.org/10.1109/WIW.2016.034.

[76] F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, An edge-based platform for dynamic smart city applications, Elsevier Future Gener. Comput. Syst. 76 (2017) 106–118.

[77] A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of Things for smart cities, IEEE Internet Things J. 1 (1) (2014) 22–32.

[78] A.E. Al-Fagih, F.M. Al-Turjman, W.M. Alsalih, H.S. Hassanein, A priced public sensing framework for heterogeneous iot architectures, IEEE Trans. Emerg. Top. Comput. 1 (1) (2013) 133–147.

[79] P.A. Laplante, N. Laplante, The Internet of Things in healthcare: Potential applications and challenges, IEEE IT Prof. 18 (3) (2016) 2–4.

[80] F. Al-Turjman, S. Alturjman, Context-sensitive access in Industrial Internet of Things, IIoT healthcare applications, IEEE Trans. Ind. Inf. 14 (6) (2018) 2736–2744.

[81] S.R. Islam, D. Kwak, M.H. Kabir, M. Hossain, K.S. Kwak, The Internet of Things for health care: A comprehensive survey, IEEE Access 3 (2015) 678–708.

[82] Y. Yuehong, Y. Zeng, X. Chen, Y. Fan, The Internet of Things in healthcare: An overview, Elsevier J. Ind. Inf. Integr. 1 (2016) 3–13.

Pablo Pico-Valencia received his Master degree in Intelligent Systems and Numerical Applications in Engineering from the University of Las Palmas de Gran Canarias, Spain. He is currently a Ph.D. student at the University of Granada, Spain. His research interests lie in the development of Internet of Things (IoT) agent-based applications based on the Internet of Services and Linked Open Data aimed at improving the interoperability, collaboration and intelligence within IoT networks.

Juan A. Holgado-Terriza received his Ph.D. degree from the University of Granada. He is currently a professor of the University of Granada and his main research interests are focused on methodologies and techniques to develop real-time applications, operating systems, virtual machines, and frameworks for embedded and mobile systems. He also is interested on concurrent systems and ubiquitous distributed systems applied to ambient intelligence, instrumentation and industry.

Patricia Paderewski received her Ph.D. degree from the University of Granada. She is currently a professor in the Languages and Systems Department of the University of Granada. Her main research interests lie in the software architecture area, software evolving and adapting, and distributed software-based agents oriented to collaborative scenarios.