Intelligent Environments 2018
I. Chatzigiannakis et al. (Eds.)

doi:10.3233/978-1-61499-874-7-567

Integration of MultiAgent Systems with Resource-Oriented Architecture for Management of IoT-Objects

Pablo PICO-VALENCIA [a] and Juan A. HOLGADO-TERRIZA [b,1]
[a] *Pontifical Catholic University of Ecuador, Esmeraldas, Ecuador*
[b] *Software Engineering Department, University of Granada, Spain*

Abstract. Recently, scientists have shown a special interest in the integration of agent-oriented technologies with the Internet of Things (IoT) in order to manage smarter IoT-objects and their resources. This paper proposes an agent architecture of Multiagent Systems based on Resource-Oriented Architecture (MAS-ROA), where the behavior of each agent is driven by a specific control workflow. This workflow enables agents to be able to perform sensing and control actions over IoT-objects by means of collaborative processes. In this way, IoT-objects managed by MAS-ROA agents can behave proactively, collaboratively, adaptively and smartly. In order to validate the proposal, an IoT ecosystem composed of many "things" (IoT-objects) is modeled and managed by agents based on MAS-ROA. The Multiagent System based on MAS-ROA was then contrasted with an implementation based on Service-Oriented Architecture (MAS-SOA) addressed by Devices Profile for Web Services (DPWS) to get insight about the differences in capabilities and performance.

Keywords. Internet of Agents, Agentification, Internet of Things, Service-Oriented Architecture, Resource-Oriented Architecture, Agent-based Services

1. Introduction

Recently, some theoretical and practical approaches have been proposed to create smart objects for Internet of Things (IoT). Two of the most relevant ones are related to the agentification of the IoT [1] and the *Internet of Agents* (IoA) [2] approaches. Both approaches propose the integration of agents with IoT technologies as a novel paradigm to control automatically, autonomously and smartly the real world through dynamic network of heterogeneous devices interconnected via Internet [3]. Hence, traditional passive IoT-objects can become active and consequently, the new generation of IoT-objects will require minimal or no user intervention to adapt their behavior to the changes that occur in their environments [4].

The IoA approach can be seen as a smart agent interaction, defined in an upper level that governs the IoT resources [5]. In parallel to this novel approach, two perspectives

[1]Corresponding Author: Juan A. Holgado-Terriza, University of Granada, Daniel Saucedo Aranda Street 18015 Granada (Spain); E-mail: jholgado@ugr.es

have been defined to realize the agentification of the IoT. The first one consists of embedding a software agent within the architecture of each IoT-object such as IoT-a [6], agent of thing [7], and smart object [8] models. On the other hand, the second approach suggests the design of Multi-Agent Systems (MASs) that control actively IoT-objects [9].

Regardless of the perspective used to agentify the IoT, the target is focused on controlling the IoT-objects smartly. For this purpose, a dynamic component capable of modifying the agent behavior autonomously according to the changes that occur in the operating environment is included [2]. To facilitate this adaptation process, several mechanisms have been merged within MASs to facilitate the design of their functionalities using ecosystems of services based on Service Oriented Architecture (SOA) or Resource Oriented Architecture (ROA) [10]. As a result, several approaches of MAS based on Service-Oriented Architecture (MAS-SOA) has been proposed [11,13]. However, because SOA web services are trending towards RESTful web services, another equivalent approach based on Resource-Oriented Architecture [14] can be defined (MAS-ROA) [15].

In this work we have focused on MAS approaches based on ROA for the agentification of the IoT. The main contributions of this paper are summarized as follows: (i) definition of an agent model based on the MAS-ROA approach that enables communication and control operations at the agent-agent and agent-object level via resources implemented with RESTful services, (ii) formulation of a workflow model oriented to control IoT-objects collaboratively through the dynamic composition of resources, and finally (iii) development of an agent model driven by dynamic control workflows easily adaptable by end-users at runtime as Yu et al. envision agent technologies and IoT ecosystems[2].

This paper is organized as follows: Section 2 presents a brief description of the MAS-ROA approach. In Sect. 3, a proposal of a reference architecture for the MAS-ROA approach is defined. In Section 4, a working example is detailed in which the MAS-ROA approach is applied to develop a system for controlling an Ambient Intelligence (AmI) scenario. Finally, Section 5 outlines our conclusions.

2. From MAS-SOA to MAS-ROA approach

The popularity and acceptance of service-oriented technologies for the development of distributed systems (e.g. web, mobile and ubiquitous applications) has led to the extension of web services and Service-Oriented Architecture (SOA) to areas such as agent-based technologies. Consequently, the MAS-SOA [11] approach has been proposed in order to take advantage of the benefits offered by web services —autonomy, interoperability, encapsulation, availability and discovery— and software agents —adaptability, proactiveness, reactivity, rationality, sociability and autonomy [16]— within a single unit of process such as a service-based agent.

2.1. MAS-SOA approach

The MAS-SOA approach has been realized in specialized frameworks [17,18] and practical applications [13,19]. Its main goals are focused on developing interoperable appli-

cations through the modeling of agent actions based on web services that can be used independently of the platform in which they were deployed. To achieve the agent goals, the developer can adopt two mechanisms, the modeling of the agent behavior in base of agent actions as static invocations of web services as well as the modeling of the agents in terms of search strategies in order to discover dynamically the suitable services.

In the first mechanism, agents must invoke one or more web services to execute specialized processes (e.g. algorithms, reasoning processes, data accessing, data validation, data fusion). In this case, the execution of the agent actions is merely sequential and, consequently, if the control flow is blocked by a failure, then the agent could not recover from that failure by itself.

In the second mechanism, a generic behavior have to model in terms of external services required by the agent. The agent can recover these services directly querying on distributed repositories such as the Yellow Pages implemented by the Java Agent DEvelopment Framework (JADE). In this way, the agent can be recovered by itself if any service fails.

The two possible strategies described above have had undoubtedly a positive impact on the development of modular MASs. In fact, the main benefits arising from merging of MASs and SOA revolve around self-adaptive, lightweight, fault-tolerant agents. Nevertheless, distributed systems based on SOA are recently scaling towards ROA [20] because RESTful web services has greater flexibility and lower implementation overhead than SOAP web services [21,10].

2.2. MAS-ROA approach

Resource Oriented Architecture (ROA) is an architectural style for distributed computing that proposes the design and development of software modeled by means of resources [14]. A resource is a distributed component that has a standard common interface through which it is directly accessed [20]. In general, the resources associated with REST architectures are implemented by RESTful web services [21]. This kind of web services expose data-types and functionalities through Uniform Resource Identifiers (URIs) supported by four web methods such as POST, GET, PUT and DELETE. From these methods it is possible to create, retrieve, update and delete resources, respectively [21].

MAS-ROA can be defined as a novel approach in which a multi-agent system is modeled as a set of single agents that achieve their goals from invoking a set of resources (implemented as RESTful services) distributed on the web. The use of such components enables agents to enhance the same properties reached by MAS-SOA agents. However, new benefits can be achieved such as a higher scalability, an improved uniform accessing, and a better performance in contrast to MAS-SOA agents [21]. These features are described as follows:

- *Lighter-weight smart agents.* The intelligence of the agents can be distributed for both architectures, MAS-SOA and MAS-ROA. In both cases, agents can be designed by decoupling the complex actions of the agent's internal structure and, consequently, the agent will be more lightweight. However, in the case of agents that adopt the MAS-ROA philosophy, they provide a better response time in the actions executed by invoking RESTful services [21].
- *Improved uniform accessing*. MAS-ROA's agents are compatible with the invocation of resources through URIs —typical of the modern Web. Thus, agents can

communicate and control diverse type of entities (e.g. agents, IoT-objects) and applications (web, mobile, ubiquitous). In addition, the use of a uniform interface facilitates user-machine and machine-machine interaction as IoA and IoT ecosystems demand.

- *Adaptive agents*. MAS-SOA agents incorporate a service discovery mechanism from which agents can determine at runtime the web services required to achieve their goals. This mechanism can be easily extended to the MAS-ROA approach in order to provide a more flexible mechanism. Therefore, agents can discover external counterpart agents dynamically and execute collaborative tasks with them invoking their corresponding RESTful services, in a scalable way.

3. Multi-Agent System based on Resources-Oriented Architectures (MAS-ROA)

A MAS-ROA architecture can be applied for developing general distributed MAS for web, mobile or ubiquitous applications. However, the reference architecture proposed in this study is specific for accessing and controlling IoT objects. These architecture is illustrated in Figure 1.

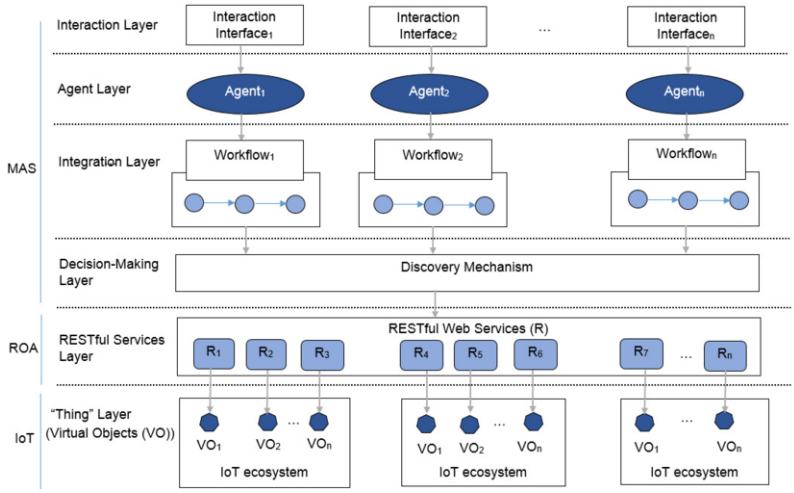


Figure 1. A reference architecture for MAS-ROA.

In general, the proposed MAS-ROA reference architecture (Figure 1) has been described using the N-layer architecture. Specifically, the architecture is composed of six layers such as: thing, service, decision making, integration, agent, and interaction layers.

At the lowest level of abstraction is the layer called *"Thing" Layer*. This layer is independent of the MAS-ROA architecture. It mainly consists of physical objects or "things" available in IoT ecosystems (e.g. sensors and actuators). This layer can include resources provided by middlewares such as openHAB —a specific platform that can access to resources associated with physical objects.

At the next level is the *Resources Layer* or ROA Layer. This layer manages and registers the RESTful services that enable the access to the available resources in IoT

ecosystems. The deployed services are only functional components that execute actions over the IoT objects. In this level, these services cannot operate proactively.

Then, the *Making-Decision Layer* is in charge of accessing the RESTful service repositories in order to discover the most suitable services to meet a specific objective. This process is performed based on the context and data required by agents (e.g. the temperature and the humidity to control HVAC system).

The data requirements used by the *Making-Decision Layer* are determined on the basis of a control logic (workflow) from which the objects connected to one or more IoT ecosystems are accessed and controlled. Therefore, a workflow defines the sequence of invocations of specific resources that must be executed to fulfill the goals associated with the *Agent Layer*.

Finally, at the highest level of abstraction is the *Interaction Layer*. This layer enables agents external users and applications (e.g. web, mobile) to make requests for monitoring the conditions of IoT ecosystems and objects connected to them.

3.1. Agent model

Based on the MAS-ROA reference architecture (Figure 1), each agent must execute a set of tasks to manage its life cycle as well the interaction to other external entities over four basic components including: communication interface, discovery mechanism, control workflow and the agent execution environment. These components are illustrated in Figure 2.

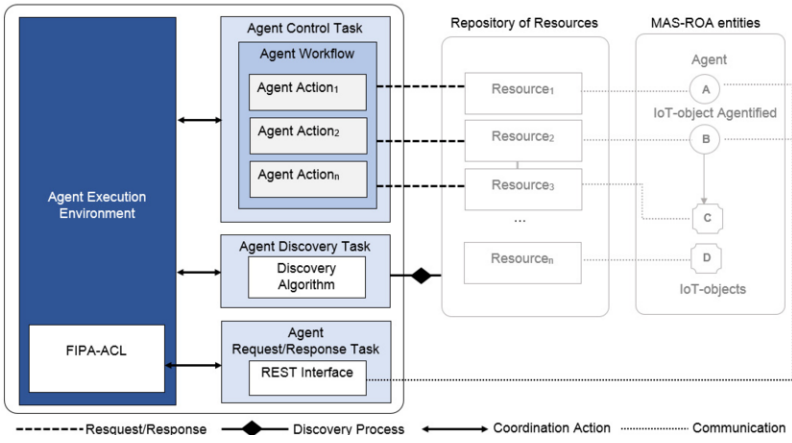


Figure 2. General schema of an agent based MAS-ROA approach.

3.1.1. Agent Request-Response Task

The communication process in a MAS-ROA is essential. An agent can communicate with external agents, IoT-objects and even with users. To address this issue, a specific task called *Agent Request-Response Task* is integrated into each agent to handle the interaction to other entity (e.g., IoT object or agent). In agent-agent interaction, the agent sender invokes a request to a specific REST interface identified by an URI in order to send a message to other counterpart agents using the Agent Communication Language (ACL)

proposed by the Foundation for Intelligent Physical Agents (FIPA). Once the request is mapped to a FIPA-ACL message, counterpart agent can receive that message. After the message is processed, a response is returned by agent receiver to agent sender through REST interface. Figure 3 illustrates the corresponding schema.

The use of URIs to carry out MAS-level communication in MAS-ROA facilitates uniform communications over web, mobile and IoT applications. Additionally, the communication processes in MAS-ROA are mapped with the parameters of a FIPA-ACL message (e.g. sender, receiver, content, language, encoding) in order to maintain support with the agent communication standard.

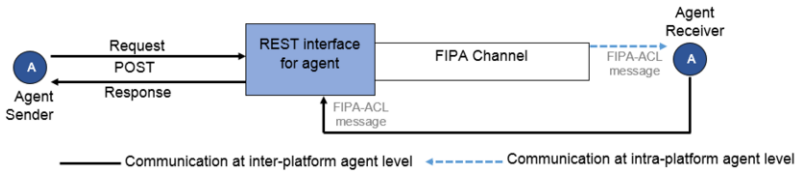


Figure 3. FIPA agent communication via REST.

3.1.2. Agent Control Task

This task is in charge of executing the control actions addressed by workflow over the objects connected to the IoT ecosystems. A workflow is the control unit that contains a description of the sequence of actions that the agent must perform to achieve its goals over the IoT ecosystem where the agent runs. The execution of the specific actions into the workflow is driven by the invocation of one or more requests to RESTful web services to get access indirectly to external resources (e.g. physical IoT-objects).

As Figure 1 shows, a workflow W_i can involve one or more invocations to specific RESTful web services $W_i < R_1, R_2, R_n >$. These services are found through a discovery mechanism executed over distributed repositories on the Web. Therefore, agent can select dynamically the suitable web services at runtime in order to fulfill its resource demands, and even change these web services by other ones when it is required (e.g. faults in servers, resources or IoT-objects).

3.1.3. Agent Discovery Task

This task is responsible to find the candidate web-services required by an agent when the workflow is executed. This task is executed frequently at different times of the agent's life cycle, such as: when the agent starts for the first time, while the agent has not completed its workflow, and when its workflow presents inconsistencies; that is, when some of the previously discovered resources report a failure in its operation.

The service discovery process consists of mapping the repositories of resources in a similar way as the JADE Yellow Pages mechanism does. This mapping process is based on a specific criteria which is based on the location context and data provided by the IoT-objects; for example, the temperature and humidity magnitudes of the room of a smart home.

3.1.4. Agent Execution Environment

The agent model employed by the MAS-ROA approach follows the guidelines of the reactive agent model. This model of agents provides autonomy based on the stimulus-response model with the capacity to react to changes in the environment in which it operates [16].

The *Agent Execution Environment* introduced as part of the agent model proposed for MAS-ROA synchronizes the life cycle of the agent. Taking into account the wide use and efficiency of the JADE framework, we recommend its use to implement this component. However, it is possible to use other frameworks (e.g. Jadex, Jason) [22], but the selected framework complies with the FIPA-ACL standard in order to ensure functional interoperability between agents that run over both intra-platforms and inter-platforms of agents distributed on the Internet.

4. Experimental evaluation

Some experiments are conducted in order to evaluate our proposed MAS-ROA architecture approach to control an IoT ecosystem through RESTful services. The experimental tests have been focused on evaluating the performance of the workflow of two agents invoking RESTful services and their capability in terms of communication (at intra-platform and inter-platform level), adaptation (performed by users and the agent itself), interoperability (using resources deployed on heterogeneous ROAs), and the capability of cooperation for the accomplishment of goals.

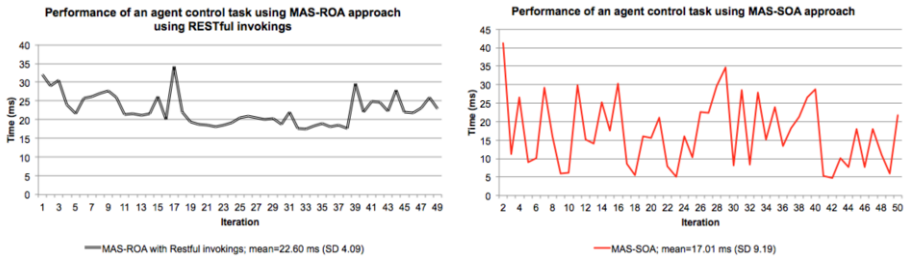
4.1. Scenario of IoT

This scenario consisted of four IoT-objects installed in the room of a smart home from which thermal and lighting comfort to their inhabitants must be offered. These objects —illumination sensor (IS), light bulb (LB), temperature sensor (TS) and heating system (HS)— were implemented with the openHAB platform because it provides a gateway for accessing to the physical IoT-objects through a REST interface.

Data associated to the resources IoT-objects were handled according to two workflows focused mainly on providing thermal ($W_{thermal}$) and lighting ($W_{lighting}$) comfort to inhabitants. The logic of control of $W_{thermal}$, specifies the necessity for reading the room's temperature (TS) and setting the heating system (HS) coherently. Likewise, the logic of control of $W_{lighting}$, defines the necessity for reading the room's lighting level (IS) and setting the light-bulbs (LB) installed in the same location.

4.2. MAS-ROA: workflows performance

We test both $W_{thermal}$ and lighting $W_{lighting}$ workflows —over a laptop with 2.5 GHz i7 Intel Core , 16 GB RAM and Windows 8.1 operating system— in order to compare the times required to be fully processed with an equivalent application based on MAS-SOA. The obtained response times are shown in Figures 4-a for MAS-ROA (invoking resources with HTTP dynamic Client) and 4-b for MAS-SOA (invoking services with the DPWS framework) after the execution of 50 iterations.



(a) Workflow based on MAS-ROA (b) Workflow based on MAS-SOA

Figure 4. Times required for processing (a) workflows based on MAS-SOA (using HTTP Dynamic Client) and (b) workflows based on MAS-SOA (using Devices Profile Web Services).

The response times required for MAS-SOA and MAS-ROA show that the workflows based on an architecture MAS-SOA required an average of 17.01 *ms* while in the case of the equivalent workflows based on MAS-ROA was required 22.60 *ms*. Although some previous works have shown an important difference between the performances of SOAP services compared to RESTful [21] services in favor of the last ones, our experiment tests seems to contradict this result (only 5.59 *ms* equivalent to 24%). The found differences depend essentially in a large extent on the technologies used. In MAS-ROA the OpenHAB platform introduces a complex framework to get access by a REST interface to heterogeneous physical IoT-objects. In contrast, MAS-SOA is implemented on lightweight services oriented to resource-restricted devices such as DPWS.

4.3. MAS-ROA: Communication performance

The communication performance was evaluated adopting a MAS-ROA agent through both the (i) JADE Gateway and a (ii) REST interface. The results obtained are illustrated in Figure 5. In general they show an increase of 76% of the time required by (a) respect to (b) to send a FIPA-ACL message and the corresponding response. However, despite the slight increase in the time required to communicate with an agent via its REST interface, the adoption of this mechanism facilitates greatly the communications at the level of inter-platform agents since the appropriate parameters are setting and encapsulated to avoid errors in communications. Even so, to minimize the time penalization of the MAS-ROA systems, it is possible to adopt a JADE gateway for the intra-agent communication process, which provides a FIPA communication mechanism at a lower level than the REST communication interface.

4.4. Additional evaluation

Further qualitative aspects were evaluated during the quantitative experiment, such as the scope for agent adaptability, and the support for interoperability in MAS-ROA. In summary, the agents invoked successfully RESTful services deployed in heterogeneous ROAs as long as those services had previously been publicly published in the Resource Directory. Additionally, the agents had the ability to manifest self-adaptation and external adaptation. On the hand, self-adaptation was accomplished by agents, which at the moment of receiving inconsistent responses from the current IoT-objects defined in their

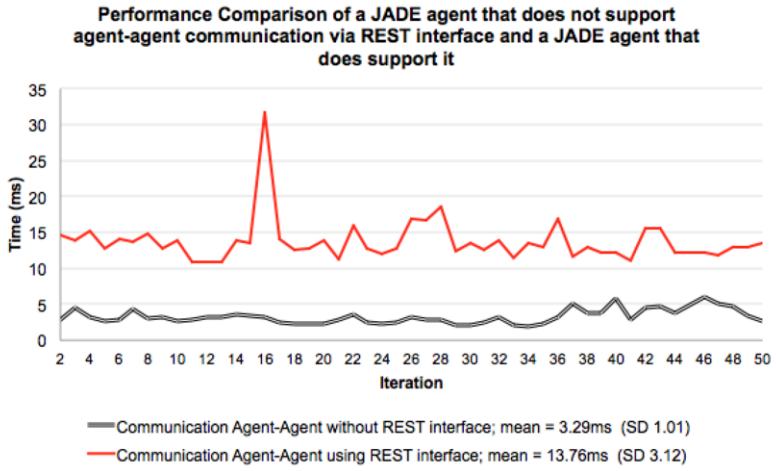


Figure 5. Times required to send&receive a FIPA-ACL message with the JADE Gateway and REST interface.

workflow, they are able to discover new equivalent REST services in the resource ecosystem. Hence, they can be recovered from resource failures. On the other hand, the external adaptation was also successfully supported by the agents after end-users changed the current workflow managed by the agent in observation. From this experience, it is recommended to create one or several workflows for each IoT-object so that agents can manage them consistently.

5. Conclusions

Resource consumption on distributed systems is currently a trend. The results obtained in this study show that agents based on SOA (DPWS services) and ROA (RESTful services) does not imply an excessive extra cost in terms of performance. Merging MASs with ROA allow for better performances because HTTP invocations use lightweight messages than generic SOAP web services [21]. However, this study has demonstrated that invocations on lightweight SOA infrastructures such as DPWS have quite similar results to RESTful services.

The use of URIs for establishing the agent communication facilitates the automation of tasks on IoT agentified scenarios because users, agents, IoT-objects and external applications (e.g. web, mobile) can interact uniformely in order to control heterogeneous IoT ecosystems. In any case, agents based on our proposal are capable to manage communications, control of IoT-objects and user-interaction using a same style. This enables interoperability between heterogeneous IoT ecosystems, agent platforms and external applications as real IoA applications for smart cities, healthcare, and smart industry demand.

MAS-ROA is a useful approach to perform the process of agentification of the IoT. In fact, our approach enables agents to coordinate control actions over IoT-objects deployed on heterogeneous ecosystems in a proactive and smart way by using a workflow that is prepared and put into operation at runtime by the agent itself. This mechanism helps us to carry out the adaptation of agents by both end-users (updating the workflow

instance) and the agent itself (updating invocations to new resources). In addition, the adoption of MAS-ROA agents provide more lightweight MASs that can run within the IoT-objects themselves.

References

[1] C. Savaglio, G. Fortino, M. Ganzha, M. Paprzycki, Costin Bădică, and M. Ivanovic. Agent-Based Computing in the Internet of Things: A Survey. In: *International Symposium on Intelligent and Distributed Computing* (2017) 307-–320.

[2] H. Yu, Z. Shen, and C. Leung. From internet of things to internet of agents. In: *IEEE International Conference on and IEEE Cyber, Physical and Social Computing Green Computing and Communications (GreenCom)* (2013) 1054–1057.

[3] S. Li, L. Xu, and S. Zhao, The internet of things: a survey, *Information Systems Frontiers* **17** (2015), 243–259.

[4] T. Perumal, M.N. Sulaiman, N. Mustapha, A. Shahi, and R. Thinaharan. Proactive architecture for Internet of Things (IoTs) management in smart homes. In: *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)* (2014) 16–17.

[5] P. Pico-Valencia, and J.A. Holgado-Terriza. Semantic agent contracts for Internet of Agents. In: *IEEE/WIC/ACM Conference on International Web Intelligence Workshops* (2016) 76–79.

[6] F. Carlier, and V. Renault. IoT-a, Embedded Agents for Smart Internet of Things. Application on a Display Wall. In: *IEEE/WIC/ACM International Conference on Web Intelligence* (2016) 80–83.

[7] A.M. Mzahm, M.S. Ahmad, A.Y.C. Tang, and A. Ahmad. IoT-a, 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)* (2016) 41:1–41:5.

[8] G. Fortino, A. Guerrieri, and W. Russo. Agent-oriented smart objects development. In: *2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design* (2012) 907–912.

[9] A. Forestiero. Multi-agent recommendation system in Internet of Things. In: *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (2017) 772–775.

[10] G. Xiaofeng, S. Jianjing, Y. Zuwei. On software development based on SOA and ROA. In: 2010 Chinese Control and Decision Conference (CCDC) (2010) 1032–1035.

[11] L. Ribeiro, J. Barata, and P. Mendes. MAS and SOA: complementary automation paradigms. In *Innovation in manufacturing networks* (2008) 259–268.

[12] M. Wooldridge. An introduction to multiagent systems. Second edition. Wiley Publishing, 2009.

[13] D. Wang, L. Ren, and J. Li. Modeling intelligent transportation systems with multi-agent on SOA. In: *2010 International Conference on Intelligent Computing and Integrated Systems* (2010) 717–720.

[14] H. Overdick. The resource-oriented architecture. In: *IEEE Congress on Services* (2007) 340–347.

[15] T. Leppänen. Resource-oriented mobile agent and software framework for the Internet of Things. University of Oulu, Faculty of Information Technology and Electrical Engineering (2018).

[16] M. Wooldridge. *An introduction to multiagent systems*. Second ed. Wiley Publishing, 2009.

[17] J. Lee, S. Lee, and P. Wang. A framework for composing SOAP, non-SOAP and non-web services, *IEEE Transactions on Services Computing* **8** (2015), 240–250.

[18] P. Pico-Valencia, and J.A. Holgado-Terriza, A framework for composing SOAP, non-SOAP and non-web services, *Procedia Computer Science* **94** (2016), 121–128.

[19] J. Wang, Q. Zhu, and Y. Ma. An agent-based hybrid service delivery for coordinating internet of things and 3rd party service providers, *Journal of Network and Computer Applications* **36** (2013), 1684–1695.

[20] R. Lucchi, and M. Millot. Resource oriented architecture and REST, *European Commission, Joint Research Centre, Institute for Environment and Sustainability, EUR* **23397** (2008).

[21] J. Wang, Q. Zhu, and Y. Ma. Performance Evaluation of RESTful Web Services for Mobile Devices, *Int. Arab J. e-Technol* **1** (2010), 72–78.

[22] R. Bordini, L. Braubach, M. Dastani, A. Seghrouchni, J. Gomez-Sanz, J. Leite, G. O'Hare, A. Pokahr, and A. Ricci. A survey of programming languages and platforms for multi-agent systems, *Informatica* **30** (2006).