Intelligent Environments 2020
C.A. Iglesias et al. (Eds.)

doi:10.3233/AISE200027

A Gateway for Enabling Uniform Communication Among Inter-Platform JADE Agents

Juan A. Holgado-Terriza [a,1], Pablo Pico-Valencia [b] Alvaro Garach-Hinojosa [a]
[a] *University of Granada, Granada, Spain*
[b] *Pontifical Catholic University of Ecuador, Esmeraldas, Ecuador*

Abstract. In general, software agents deployed in a multi-agent systems (MASs) collaborate and share data with each other by means of Foundation for Intelligent Physical Agents (FIPA) communications into the same agent platform (intra-). Java Agent Development framework (JADE) provides a facility software framework to build software agents executing in a specific agent platform. This paper presents a novel agent communication gateway aimed at abstracting FIPA communication between software agents on external (inter-) platforms. The novel agent communication gateway is based on the Agent-as-a-Service model (AaaS) implemented using REST technologies. The use of this gateway can favor and enable the interoperability among agents belonging to different MASs, opening collaborations and interactions among heterogeneous agents everywhere in Internet, whenever FIPA message exchanges are used as standard. In this paper some communication paradigms are presented based on the proposed gateway. Some implementation issues show that the programming complexity of communication blocks of agents can be decreased at both Java applications and Java web systems.

Keywords. Multiagent system, agent, JADE, FIPA, communication patterns.

1. Introduction

Multi-agent systems (MASs) integrate a set of software agents that communicate and collaborate with each other in order to achieve specific goals [1]. Software agents are entities that support properties such as proactivity, intelligence, autonomy, collaboration, mobility, adaptability and social ability in the environment where they operate [2].

According to the architecture and elements that integrate the software agents, these entities can be reactive, deliberative or hybrids [3]. Reactive agents constitute entities that integrate a mechanism to intercept events occurred in their environment and trigger actions to establish control operations that change such environment. On the other hand, deliberative agents, also known as BDI (belief, desire, intention) agents [4], are positioned as a complementary agent model to reactive one. BDI agents model their behavior using a mental state that allows them to make decisions in the environment in which they

[1]Corresponding Author: Juan A. Holgado-Terriza. University of Granada, Daniel Saucedo Aranda Street 18014 Granada, Spain; E-mail:jholgado@ugr.es

operate similarly to a human being. BDI artifacts turn deliberative agents into more intelligent entities than reactive agents. Finally, hybrid agents arise as an artifact that merges the best capabilities of reactive and deliberative agents.

Regardless of the architecture that agents adopt, software agents have inherent communication skills that allow them sharing data and messages. Consequently, modeling collaborative processes to meet their goals is enabled in distributed systems. Nevertheless, the agent communication between MASs is still a limitation in terms of complexity and interoperability [5]. These limitations have arisen due to the proliferation of agent tools that have not been regulated from the beginning of agent oriented technologies [6]. To solve this concern the Foundation for Intelligent Physical Agents (FIPA) [5] has emerged in order to establish standards in the development of software agents. Thus, one of the most significant contributions done by FIPA have been the development of the agent communication language (ACL) [7]. This language is currently a standard that agents tools must comply with in order to provide interoperable MASs.

Many of the tools that implement ACL as the standard for agent communication are currently used for the agent development. This language does not only ensure interoperability between distributed MASs developed in the same tool, but also enables them to communicate with heterogeneous MASs developed with any tool that comply with the FIPA standard. These advances rank the agents as very useful entities for the development of intelligent and collaborative distributed systems in environments such as desktop applications, web sites [8], mobile apps [9] and Internet of Things (IoT) systems [10].

Web and IoT systems have already begun their transition towards the use of services and/or resources [11]. This implies that processes are modeled from the orchestration of services and resources available on the Internet or intranets. In both cases the communications are done using uniform resource identifiers (URIs), providing a common way to access everywhere.

The integration of agents with service technologies can provide a paradigm to control automatically, autonomously and smartly the real world through dynamic network of heterogeneous devices interconnected via Internet [12]. The interplay of web-services and agents are usually managed based on two approaches [13,14]. On one hand, web-services can be integrated into agent platform in order to make accessible resources available in Internet to agents. In this case, the agents can request data or functionality on web-services addressed by a service architecture. Then, no additional mechanism is needed to add web services into the agent framework.

On the other hand, web-services can be used to hide the functionality of agents or a MAS behind the services. From external point of view the web-service is behaved exposing the agent functionality using an URI. An example is the Web Service Dynamic Client (WSDC), an add-on of JADE that provide web-service facilities to JADE agent. In this case, the web-service employs also an Agent-as-a-Service model based on a service oriented architecture (SOA). Then, a consumer service can invoke web-service by message exchanges using Simple Object Access Protocol (SOAP).

In this paper, the proposed gateway provides a novel JADE mechanism to interconnect heterogeneous agents living in different platforms located anywhere. On one hand, the gateway exposes an receiver agent of a specific platform as a web-service based on the second approach and, on the other hand, this web-service is provided as facility to a sender agent that wants accessing to receiver agent based on the first approach. The involved communications paradigms are presented in this work.

This paper is structured in five sections. Section 2 presents the review of the main agent platforms and their capability to support FIPA communication as well as the main patterns of communication implemented in JADE. Section 3 describes a gateway oriented to create blocks for establishing agent communication at both intra- and inter-communication agent platforms. In Section 4 some examples of the communication paradigm is detailed. Finally, Section 5 summarizes the conclusions.

2. Software Agents and Communication Patterns

Some of the most relevant studies aimed at comparing agent tools have been surveyed. These studies have compared and evaluated the most important agent tools categorizing them as programming languages [15,16,17] and agent platforms [18,19,20,21,6]. However, we have only considered agent platforms to be analyzed in this paper.

2.1. Agent Platforms

The comparison of the existing agent tools considers the following aspects: (i) type of tool such as toolkits, programming languages [17]; (ii) supported agent model such as reactive, deliberative and hybrid [3]; (iii) communication standard compliance such as FIPA [22,23]; and (iv) additional supported mechanisms such as mobility and security.

It is remarkable that some of the existing agent tools have not become popular among developers because they are not compatible with FIPA standards. However, tools that support this standard such as JADE has become widely used to create intelligent agent-based systems useful in ambient intelligence, healthcare, smart cities and industry [24].

Positive features of JADE framework such as its popularity, full documentation and its distribution as open source have motivated developers to use this tool as the baseline to create MASs. Nonetheless, a strong criterion for selecting JADE as a platform for creating agent-based systems is its usability and supporting of FIPA standards [19]. In addition to JADE, others consolidated FIPA-compatible tools have been developed, some based on Java such as Jadex or JACK and others based on Python such as Python Agent DEvelopment Framework (PADE) and Smart Python multi-Agent Development Environment (SPADE). Independently of the used technology, achieving interoperability between these platforms is technically feasible when FIPA is supported [25]. Only few tools do not support FIPA standars such as MADKIT.

2.2. Communications in JADE

JADE is a software framework for the development of agent applications in compliance with FIPA specifications for inter-operable intelligent MASs [7]. In terms of communication, JADE supports the following features: (i) a Java API to send/receive ACL messages to/from other agents, (ii) a library of FIPA interaction protocols ready to model complex interactions, (iii) compliance with the Internet InterORB Protocol (IIOP) to connect different agent platforms, and (iv) an usable GUI to manage agents and their interactions.

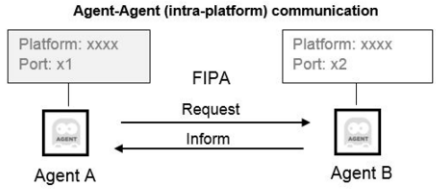
Agents developed in JADE use FIPA-ACL to share messages. An ACL message is a set of encoding elements that allows to transmit a series of knowledge expressed in a content language. The terms used correspond to a common vocabulary in order that any agent that supports this language understands each received message. Main elements that

make up an ACL message are the following: (i) a sender, (ii) a receiver, (iii) a content, (iv) a language, and (v) an ontology that describes the semantics of the message. In addition, it is also needed to define the message type to be composed, that is, INFORM, QUERY, REQUEST or PROPOSE.

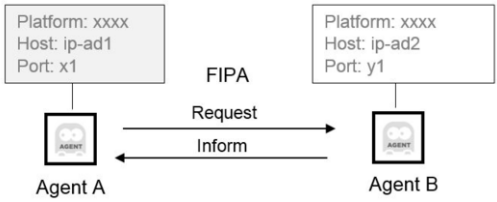
The ACL language allows any agent developed in any programming language to use the same message format to interact with each other. So, agents can have the ability to share messages and data with agents running on heterogeneous agent platforms from different MASs, regardless of the programming language used.

2.3. JADE Communication Pattern

JADE uses the concept of platform to distribute agents in different hosts. Figure 1 illustrates the main communication patterns implemented by this tool. These patterns include two-level communication processes, that is, intra-platform (Figure 1a) and inter-platform (Figure 1b). In both cases, the communication patterns are request/inform processes in a similar way as the request/response paradigm in resource-oriented architecture (ROA) [26].



(a) Agent-Agent (intra-platform) communication



(b) Agent-Agent (inter-platform) communication

Figure 1. Basic patterns of communication supported by JADE.

Each request/inform communication process involves two entities, a sender and a receiver. Generally, in the basic communication patterns illustrated in Figure 1, both sender and receiver are agent entities that share an ACL message.

2.3.1. Communication at Intra-Platform Level

The agent-agent communication pattern illustrated in Figure 1a constitutes the basic communication processes that JADE provides to MAS developers to share messages between agents running on the same platform. These patterns are used to implement the FIPA communication protocols such as: FIPA-Request, FIPA-Query, FIPA-Contract-Net, FIPA-Subscribe, FIPA-Propose [27].

In the pattern shown in Figure 1a both sender and receiver run on the same MAS. It is the typical communication at intra-platform level. To carry out this request/response process it is needed to define two arguments. The first one is the name of the platform in which the agent is running and the second aspect is the port from which the platform makes the connection. Both are mandatory for sender and receiver to establish a message exchange.

This pattern is elemental because it includes a single sender and a single receiver. However, it is possible to model more complex communication patterns with more than two receivers [28]. In this paper, we focus on one sender and only one receiver because our objective is to create a gateway to communicate with single agents that can be compatible with more complex communication patterns.

2.3.2. Communication at Inter-Platform Level

The communication patterns between agents at the inter-platform level follow the same structure at the intra-platform level. However, as illustrated in Figure 1b, in this pattern, both sender and receiver agents reside on different platforms that run on the same/different hosts in same/different networks. In this case, a communication can be set when agents know the platform name of the corresponding counterpart agent, their IP addresses and specific ports of the hosts where agent platforms of the counterpart agents are running.

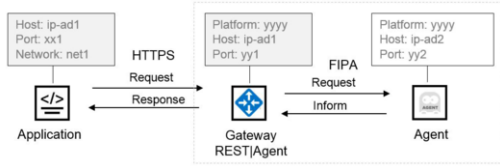
Although the communication pattern at inter-platform level is rather general, some differences can be set depending where the platform is placed. Some differences and restrictions can be found in the execution of the communication pattern depending where the agent platforms are running. When the sender and receiver agents are running on platforms operating on the same host, they should have defined different ports. This case corresponds to agents of two MASs running on the same machine but sharing certain goals. On the other hand, when sender and receiver agents run on different hosts that connect to the same network, agent platforms that integrate the both agents involved in the communication can run on the same port without conflict as in the previous case. This pattern is recommended when it is necessary to have two MASs that run with their own computational resources.

Finally, when sender and the receiver agents are located on hosts in different networks, additional permissions are required to establish communication between two hosts. This pattern has similarities with the previous one but in this case, hosts where the agent platforms run are connected to different networks. Therefore, port conflicts, the use of the same platform names and difficulties to establish the link between different networks caused by firewalls may occur. However, it is feasible and advisable to give different names to avoid confusion when sending a message to a distributed agent.

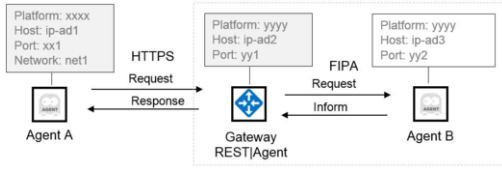
3. Agent Communication Gateway

The agent communication gateway proposed in this paper was designed especially to assist the agent-agent communication at inter-platform level, since in general it is more difficult to establish and maintain the communication link between both parties. This opens up the possibility to make collaborations between agents from different MASs in a simple way, whenever FIPA-ACL is used in the message exchange.

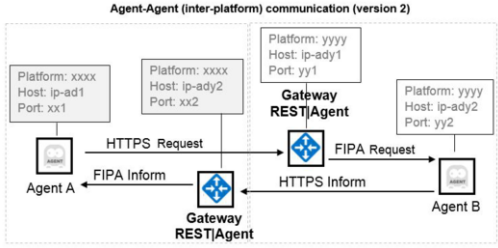
This agent communication gateway was developed based on the JADE framework. The popularity of JADE among the community of agent developers as well as its wide use in the development of MASs for the intelligent control of scenarios have been some of the motivations that have led us to use JADE as a baseline for the development of our proposal. In order to abstract the programming complexity of blocks concerning the JADE communication patterns a specialized API for Java has been developed. This API implements a gateway based on RESTful services. RESTful web services provide autonomous, lightweight, and scalable software entities to manage contents (functionalities and data) in terms of resources. These resources enclosed in a web server can be exposed by web services through a standard common interface, the 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 [29].



(a) Application-Agent communication



(b) Agent-Agent (inter-platform) communication (vers. 1)



(c) Agent-Agent (inter-platform) communication (vers. 2)

Figure 2. Basic communication patterns of the proposed gateway.

Three different usages of the gateway can be shown schematically in Figure 2. The three communication patterns are: (i) application-agent communication in Figure 2a, (ii) agent-agent (inter-platform) communication (version 1) in Figure 2b and (iii) agent-agent (inter-platform) communication (version 2) in Figure 2c.

3.1. Application-Agent Communication

In this case, the agent communication gateway is managed by an external application such as a web service, a web component or an executable program, as shown in Figure 2a. This pattern is usually common when an external application requires the execution of specialized actions of intelligent agents who autonomously search for the resources to execute the desired action (e.g., a travel agency website that searches for the most appropriate means of transport to the client's request).

In a general view, the communication gateway acts by separating the FIPA communications that are established between the gateway and the agent to be accessed, with respect to the external communications that are carried out through the gateway.

3.2. Agent-Agent (Inter-Platform) Communication

Two versions of the agent-agent communication paradigm at inter-platform level were identified. The lighter version of the agent-agent communication paradigm shown in Figure 2b establishes a communication link between the sender agent and the receiver agent behind the gateway. Since the gateway and the receiver agent are operating on the same platform in the same network domain, the FIPA message exchange between the gateway and receiver agent is simplified; only the agent name is needed.

On the other hand, the sender agent, probably running on a host in a different network domain, is forced to use an HTTP request-response protocol in order to have access to the REST interface of the RESTFul service implemented in the gateway. This web service has defined a specific resource associated to the invocation of the FIPA message interchange to the receiver agent. When the resource of the service in the gateway is consumed by a HTTP request, the FIPA message interchange is started.

This communication paradigm is especially interesting when the sender agent is an autonomous, distributed and isolated agent or an agent that belongs to a MAS. In both cases the agent requires to have access to the services offered by one or several agents that form part of a MAS. In this way it is possible to interoperate between different agent platforms as long as FIPA-ACL is maintained in the message exchange between agents.

In contrast, the heavier version of the agent-agent communication paradigm shown in Figure 2c provides a more elaborate communication among inter-platform agents. The HTTP request-response message on the RESTful service of the gateway is decoupled into two POST requests, each one to the RESTful service of the counterpart gateway. Each HTTP request to counterpart gateway encloses the ACL message only in one-way direction as occurs in general in FIPA messages. This allows a set of benefits to agent-agent communications: (i) the FIPA message exchanges can be asynchronous; (ii) more complex FIPA messages can be exchanged among JADE agents using different performatives, not only REQUEST and INFORM; (iii) FIPA messages can be shared among more parties, not only between two ones. This can be especially interesting in order to make many-to-many collaborations among agents at both domains as if they were living in a single MAS.

3.3. Implementation of the Agent Communication Gateway

The agent communication gateway has been defined under the Agent-as-a-Service model. Therefore the gateway from an external point of view can be seen as a RESTful

service with a REST API containing a resource or several ones with an URI that corresponds to the agent or agents accessible by the gateway. When a resource is consumed through a POST request that encloses the FIPA message as a JSON object in the body of this HTTP request, the RESTful service will start the agent management component implicit in the gateway and send a FIPA REQUEST to the agent specified in that agent platform.

The architecture of the agent communication gateway is composed of two parts: i) RESTful service management components and ii) agent management components that will exchange the message with the receiving agent, as shown in Figure 3. The design of the gateway allows decoupling both sides. The representation of the agents shared with the proposed gateway based on resources with an URI simplifies its implementation and handling.

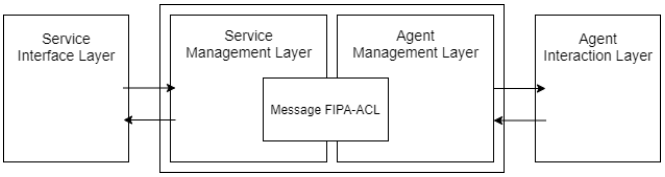


Figure 3. Architecture of the agent communication gateway.

In the implementation of the agent-agent communication paradigm of Figure 2b the agent management component must obtain the response through a FIPA INFORM request from the receiving agent, and this is encapsulated in the body of the HTTP response. On the other hand, in the implementation of agent-agent communication paradigm of Figure 2c, the requests only occur in one direction, without a response. Then, the implementation of the gateway is slightly different from the above case.

4. Results

In order to demonstrate the operation of the communication patterns proposed in this study, two of these patterns, the Application-Agent (Figure 2a) and the Agent-Agent pattern (Figure 2b), are detailed.

The Application-Agent communication pattern was applied to communicate a web application developed with Java Server Pages (JSP) into an agent developed in JADE. In order to reproduce the communication pattern a web-server was installed into a PC where JSP pages were hosted and the gateway was implemented in JAVA as a a RESTfull web-service with JAX-RS 2.0 as is shown in 4. Inside the logic of the web-service a gateway agent is built to send a FIPA message to the agent that is running into a Raspberry Pi embedded device.

On the other hand, the Agent-Agent pattern (Figure 2b) was developed and tested with a specific example. In this case, three entities were implemented. A receiver JADE agent with a cyclic behavior that is waiting to read a FIPA message received from gateway. Next, the gateway is implemented as a RESTful web-service using JAX-RS 2.0 library using Java programming language. Inside the logic of RESTful web-service a gateway agent is built which includes the conversion of the ACL message from JSON to an ACL object in JADE. Finally, a sender agent is developed in JADE that has associ-

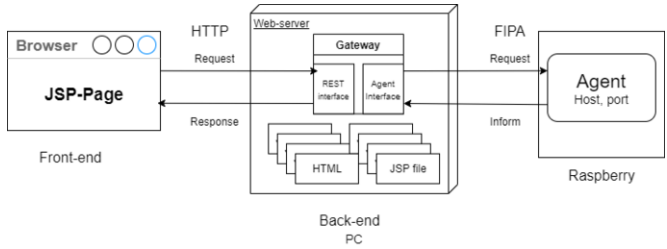


Figure 4. Deployment of the Application-Agent communication paradigm.

ated a simple behavior which contains the invocation of HTTP request in order to get a response from the receiver agent.

5. Conclusions

Resource Oriented Architecture (ROA) technologies have now reached a maturity level to develop distributed systems compatible with desktop, web, mobile, and pervasive computing scenarios such as IoT. However, services are passive entities and are not able to search for the suitable resources to achieve their goals as a software agent would. Developing a communication interface such as the one proposed in this paper not only facilitates communication with agents distributed in MASs but also enables other entities to communicate with them. Web applications and users are some of the entities that can communicate with agents only by composing a URI that incorporates simple arguments to remember.

The communication with JADE agents using URIs abstracted the complexity of agent communications and facilitated the modeling of heterogeneous processes based on the composition of services and agents. This makes it possible to take advantage of intelligence distributed through agents from emerging applications such as: web and IoT. Therefore, the gateway contributes a tool to work with smart agents interactions from heterogeneous MASs as if they were living in a single MAS.

References

[1] Stone P, Veloso M. Multiagent systems: A survey from a machine learning perspective. Autonomous Robots. 2000;8(3):345–383.
[2] Rusell S, Norvig P. Artificial intelligence: A modern approach. Pretice Hall Series in Artificial Intelligence. 2003;1.
[3] Balke T, Gilbert N. How do agents make decisions? A survey. Journal of Artificial Societies and Social Simulation. 2014;17(4):13.
[4] Thangarajah J, Padgham L, Harland J. Representation and reasoning for goals in BDI agents. Australian Computer Science Communications. 2002;24(1):259–265.
[5] Poslad S, Charlton P. Standardizing agent interoperability: The FIPA approach. In: ECCAI Advanced Course on Artificial Intelligence. Springer; 2001. p. 98–117.
[6] Kravari K, Bassiliades N. A survey of agent platforms. Journal of Artificial Societies and Social Simulation. 2015;18(1):11.
[7] Bellifemine F, Poggi A, Rimassa G. Developing multi-agent systems with a FIPA-compliant agent framework. Software: Practice and Experience. 2001;31(2):103–128.

[8] Bădică C, Bassiliades N, Ilie S, Kravari K. Agent reasoning on the web using web services. Computer Science and Information Systems. 2014;11(2):697–721.
[9] Corchado JM, Corchado ES, Pellicer MA. Design of cooperative agents for mobile devices. In: International Conference on Cooperative Design, Visualization and Engineering. Springer; 2004. p. 205–212.
[10] Singh D, Tripathi G, Jara AJ. A survey of Internet-of-Things: Future vision, architecture, challenges and services. In: 2014 IEEE world forum on Internet of Things (WF-IoT). Seoul, South Korea: IEEE; 2014. p. 287–292.
[11] Pico-Valencia P, Requejo-Micolta B, Holgado-Terriza JA. Multiagent System for Controlling a Digital Home Connected Based on Internet of Things; 2020.
[12] Li S, Da Xu L, Zhao S. The internet of things: a survey. Information Systems Frontiers. 2015;17(2):243–259.
[13] Braubach L, Alexander P. Conceptual Integration of Agents with WSDL and RESTful Web Services. In: Dastani M LBe Hübner J F, editor. Programming Multi-Agent Systems. ProMAS 2012. vol. 7837 of Lecture Notes in Computer Science. Springer; 2013. p. 17–34.
[14] Pico-Valencia P, Holgado-Terriza JA. Integration of MultiAgent Systems with Resource-Oriented Architecture for Management of IoT-Objects. In: Intelligent Environments 2018. vol. 5 of Ambient Intelligence and Smart Environments. IOS Press; 2018. p. 567–576.
[15] Bordini RH, Braubach L, Dastani M, El Fallah Seghrouchni A, Gomez-Sanz JJ, Leite J, et al. A Survey of Programming Languages and Platforms for Multi-Agent Systems. Informatica. 2015;30(10):33–44.
[16] Bădică C, Budimac Z, Burkhard HD, Ivanović M. Software agents: Languages, tools, platforms. Computer Science and Information Systems. 2011;8(2):255–298.
[17] Dastani M. In: Shehory O, Sturm A, editors. A Survey of Multi-agent Programming Languages and Frameworks. Springer Berlin Heidelberg; 2014. p. 213–233.
[18] Vrba P. JAVA-Based Agent Platform Evaluation. In: Mařík V, McFarlane D, Valckenaers P, editors. Holonic and Multi-Agent Systems for Manufacturing. Springer Berlin Heidelberg; 2003. p. 47–58.
[19] Leszczyna R. Evaluation of agent platforms. European Comission, Joint Research Centre, Institute for the Protection and Security of the Citizen; 2004.
[20] Trillo R, Ilarri S, Mena E. Comparison and Performance Evaluation of Mobile Agent Platforms. In: Third International Conference on Autonomic and Autonomous Systems (ICAS'07); 2007. p. 41–41.
[21] Shakshuki E, Jun Y. Multi-agent Development Toolkits: An Evaluation. In: Orchard B, Yang C, Ali M, editors. Innovations in Applied Artificial Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg; 2004. p. 209–218.
[22] Soon GK, On CK, Anthony P, Hamdan AR. A review on agent communication language. In: Computational Science and Technology. Springer; 2019. p. 481–491.
[23] Chaib-draa B, Dignum F. Trends in agent communication language. Computational Intelligence. 2002;18(2):89–101.
[24] Pico-Valencia P, Holgado-Terriza JA, Herrera D, José S. Towards the internet of agents: an analysis of the internet of things from the intelligence and autonomy perspective. Ingeniería e Investigación. 2018;38:121–129.
[25] Bellifemine F, Caire G, Poggi A, Rimassa G. JADE: A software framework for developing multi-agent applications. Lessons learned. Information and Software Technology. 2008;50(1):10–21.
[26] Lucchi R, Millot M, Elfers C. Resource oriented architecture and REST. Assessment of impact and advantages on INSPIRE, Ispra: European Communities. 2008;.
[27] Juneja D, Jagga A, Singh A. A review of FIPA standardized agent communication language and interaction protocols. Journal of Network Communications and Emerging Technologies. 2015;5(2):179–191.
[28] Moran S, Luger E, Rodden T. Exploring patterns as a framework for embedding consent mechanisms in human-agent collectives. In: International Conference on Active Media Technology. Springer; 2014. p. 475–486.
[29] Saad MK, Abed R, Hamad HM. Performance evaluation of restful web services for mobile devices. International Arab Journal of e-Technology. 2010;1(3):72–78.