A Reference Architecture for the IoT Services' Adaptability
Using Agents to Make IoT Services Dynamically Reconfigurable

Ademir José Barba and Fernando Antonio de Castro Giorno
Department of Master's Degree in Computer Engineering, Instituto de Pesquisas Tecnológicas, IPT,
Av. Prof. Almeida Prado, 532 - Cidade Universitária, São Paulo, Brazil

Keywords: Internet of Things, IoT, Software Architecture, Multi-Agent Systems.

Abstract: Internet of Things (IoT) is a concept that illustrates the technological revolution that allows the interaction between physical things (devices) and virtual things (software) thanks to the Internet and to the evolution of the sensing and acting devices. This interaction promotes the creation of advanced services that contribute to society. The evolutionary maintenance of IoT services and the inclusion of new services demand a software architecture that adapts to these changes without causing damage to the rest of the system. In this work this requirement is called "IoT Services Adaptability" and to propose an architecture that contributes with this requirement is the objective of this work.

1 INTRODUCTION

According to (Vermesan et al., 2015), Internet of Things is a concept and also a paradigm consisting of an environment that contains the pervasive presence of things or objects capable of communicating and cooperating with each other in order to provide services and achieve common goals. This is done through wireless or wired connections and unique addressing schemes. IoT uses the synergy generated by the convergence of consumer Internet, business Internet and industrial Internet to create a global, open network that connects people, data, and things. In this context it is possible to conclude that IoT Services are services that are not obvious without the intelligence resulting from the interaction between these elements, which is made possible thanks to this level of connectivity.

An example of a service could be the intelligent house, where sensors scattered around the house help in the automatic accomplishment of tasks such as adjusting the brightness according to the occasion, making a purchase order in the supermarket after finding the lack of a product in the refrigerator, etc. This concept is similar to the concept of Agent of Things proposed by (Mzahm et al., 2014). In their proposal, an Agent of Things is a software agent that allows the reasoning about the associated agents and provides services that would be infeasible without these interaction.

This work is in agreement with these concepts of IoT Services. Here, the services also contextualizes the environment, makes decisions based on the contextualized data and acts over this environment.

IoT Services' Adaptability, in the context of this work, means to reconfigure IoT Services parts without other Services parts stop working. It also means adding new IoT Services without the other services ceasing to work. All these changes are made at runtime.

To try compose a software architecture that contributes to the IoT ServicesAdaptability, a working method was followed. The steps of this method are:

- Bibliographical review;
- Composition of the reference architecture;
- Composition of the architecture of a Railway Management System;
- Implementation of the prototype of a Railway Management System;
- Testing of the reference architecture;
- Generation of the conclusions of the work;

2 BIBLIOGRAPHICAL REVIEW

Before starting this work it was not known the requirements that a software architecture should attend to contribute to the adaptability of IoT Services, but

Barba, A. and Giorno, F.
A Reference Architecture for the IoT Services' Adaptability.
DOI: 10.5220/0006577101870194
In *Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS 2018)*, pages 187-194
ISBN: 978-989-758-296-7

it was already known that for the reconfiguration of IoT Services at runtime, independent pieces of software should allow their replacement without damaging the other parts. This influenced the choice of an agent-based architecture, since agents do not expose their internal actions and they interact with the environment through sensing and acting. The dynamic changing and the dynamic including of IoT Services in the system besides being part of the objective of this work are also considered fundamental requirements of this architecture.

The other requirements of the reference architecture proposed by this work arose from the observation of elements that could contribute with the objective of this architecture. These elements were selected from the chosen works. And the chosen works were selected from the works that address the following subjects: Internet of Things, software architecture, multi-agent systems.

This section lists the main contributions of the related works for the composition of these requirements.

The work of (Mzahm et al., 2014) creates a concept where software agents enable the reasoning, the negotiation and the delegation of tasks to the devices which they are associated. This concept is called Agent of Things. The main contribution of their work to this one was to make clear the importance of the interaction between IoT devices in order to perform services that would not be possible without this interaction. These observations gave rise to the requirements of decision-making by the IoT Service about the facts collected from other agents and of acting by the IoT Service in the environment based on the results of inferences about these facts.

(Dimakis et al., 2006) propose a middleware architecture for the integration of services that recognize the context of the environment they are inserted. Its proposal also includes a great amount of features that maximize the autonomy of these services. This architecture includes components for context acquisition and components for modeling situations and services. Many of these components are implemented as software agents, which allows for greater autonomy to perform services thanks to the advantages of the multi-agent approach. The main contribution to this work was to make clear the need for a layer composed of agents to contextualize the data provided by IoT devices so that other agents can benefit from this data and perform their assigned actions. By this way, the act of contextualizing the data provided by other agents becomes one of the requirements of the IoT Service in the reference architecture.

(Leppnen and Riekki, 2013) present an architecture based on agents, on the REST principles and on the Internet Drafts of the IETF CoRE Working Group. In this proposal software agents describe the state of a computational task, which is in turn disseminated through messages among the system participating devices. Their work contributes to the concept of Resource Directory, an element that exposes the resources and services existing in the system.

(Leppnen et al., 2014) propose an approach in which objects exposed on the Web offer an internal architecture that enables the interaction of humans with mobile software agents. These objects are called smart objects. In this work the concept of Resource Directory is also mentioned. The work exposes an interface that users can use to interact with the system. This highlighted the importance of having an interface so that the system administrator can configure IoT Services dynamically (at runtime). This is therefore another requirement of the reference architecture.

The work of (Kim et al., 2012) shows an architectural approach based on FIPA's standards, in which it proposes a repository (Facilitator Directory) of services provided by software agents. This repository incorporates the feature of service context categorization. The classification of services into categories assists in the search for IoT Services based on the context of the application or based on the context of the consumer user. Both the Resource Directory and the Facilitator Directory concepts contribute to the existence of an agent (Facilitator Agent) that takes care of the message routing requirement because they allow other agents to know who is interested in their data by consulting an element that has all system information consolidated in one place.

(Ayala et al., 2012) propose an internal modularization of IoT software agents applying Aspect-Oriented Software Development concepts that allows the creation of an agent that works in a lightweight device capable of interacting with its environment and with other agents orienting its behavior by goals and by the recognition of the context which it is inserted. The agent's behaviors are encapsulated through components and aspects. This modularization makes it possible to enable, disable, configure, and compose different agent properties at runtime. The identification of contexts of the environment, the decision based on the acquired contexts and the separation of the agent's behavior are concepts that the IoT Services proposed by this work will use, since it allows their reconfiguration. In this work the separation of IoT Service perception, reasoning and behavior into agents gives the inherent independence of the concept of agent to the parts that compose this service, allowing these parts to be manipulated without the others

parts of the service ceasing work. This separation contributes, therefore, to the existence of independent pieces of software.

Another requirement of the reference architecture is the existence of a message structure common to all agents for communication to be viable, as this would establish a communication protocol among the agents. This requirement is a based in the belief that even independent software pieces need to know the composition of the messages that flow over the environment if they want to communicate each other.

3 COMPOSITION OF THE REFERENCE ARCHITECTURE

All the contributions reported in the previous section were selected based on the benefits that could bring to the IoT Services' Adaptability. These reported elements allowed to formulate the requirements for the construction of an architecture that contributes to this goal. Table 1 contain these requirements.

Table 1: Reference architecture requeriments.

1. Dynamic IoT Service inclusion (with system administrator action)
2. Dynamic IoT Service changing (with system administrator action)
3. Existence of independent software parts (agents)
4. Existence of an IoT Service configuration interface
5. Contextualization of the components that serve the IoT Service (perception)
6. IoT Service decision-making capacity (reasoning)
7. IoT Service acting over environment capacity (behavior)
8. Existence of a common language structure
9. Routing of messages and middleware

3.1 Development Model

The development method chosen to design the reference architecture and to implement a prototype based on it was the MDA (Model Driven Architecture). This method was created by OMG (Object Management Group) and puts the modeling at the heart of development. One of the reasons for choosing this method is that it is capable of generating independent platforms models, thus increasing the possibilities of using the reference architecture.

(Elammari and Issa, 2013) propose the use of a model-driven approach to the development of a MAS (Multi-agent System) architecture. Their work shows how to apply Use Case Maps diagrams to define the problem domain illustrating agents and responsibilities. In addition, they make use of tables that define the roles of the agents and tables that define the relationship contracts between the agents.

The MDA has three development phases, the first two was used to design the reference architecture. The definition of the reference architecture is shown in the next subsection.

3.2 Reference Architecture Design

The first fase of MDA is called Computational Independent Model (CIM). This phase consists of the definition of the problem domain and the description of the system requirements. In this phase the functions performed by the system are defined by Use Case Maps (UCM) diagrams.

Figure 1 shows the definition of the responsibilities and agents that compose an IoT Service and the definition of the agents that interact with this service.

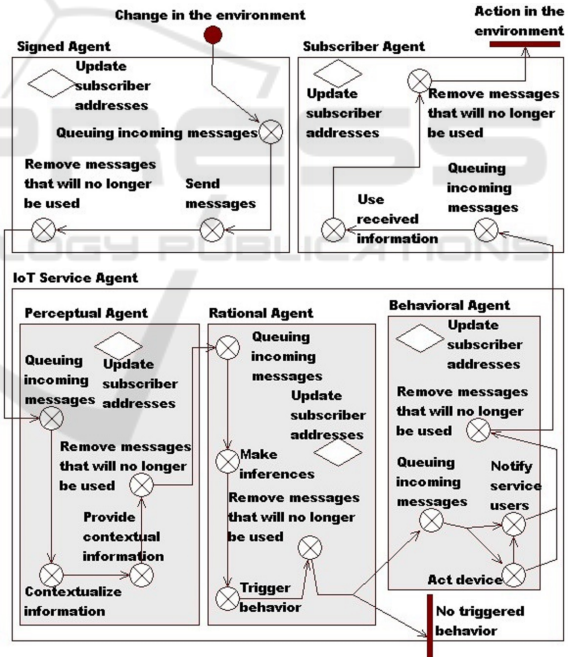

Figure 1: Conceptual IoT Service Agent.

In the used notation each rectangle corresponds to an agent. Circles containing X represent the responsibilities of the respective agent. The diamond symbol represents a stub (responsibility that may be more detailed). The arrows represent the flow of responsibilities activation. It's possible to notice bifurcations in this flow by showing alternate paths. In Figure 1 it is possible to visualize the application of the requirements of the reference architecture. The IoT Service

agent is a conceptual agent composed of at least one Perceptual Agent, one Rational Agent and one Behavioral Agent. This gives flexibility to change a IoT Service since it is composed of agents that respond or not to stimuli of the environment in which they are. Assuming that a service has several behaviors (Behavioral Agents), it is possible to replace one without the perceiving, the reasoning and the other behaviors of the service being affected.

The interaction of the System Configuration Interface with the Facilitator Agent by changing the system through agent exchanges or through the insertion of new agents is also portrayed in UCM diagrams. Once exchange or inclusion occurs, the Facilitator Agent informs the agents of the new addresses of those interested in their data, in addition, the Facilitator updates your own database and the interface's database. Each agent keeps a address list in a own database.

The second fase of MDA describes the static and dynamic models of a system without to worry with the platform on which the system will be developed.

The static model of the system is composed of class diagrams that illustrate the composition of the environment of a MAS that follows the reference architecture. In this case, each class represents each of the types of agents that may exist in the reference architecture and the environment is represented by a class that aggregates the agents that are part of this environment.

To complement the static view of architecture, tables containing objectives, plans, tasks, and beliefs describe the role of these agent types.

The dynamic model of the system are described through sequence diagrams illustrating the interaction of the IoT Service agents with the agents signed by the service and with the service signing agents. The dynamic model is also described by sequence diagrams showing the interaction of the Facilitator Agent with the Configuration Interface and with other agents. Tables containing a list of authorizations, obligations and policies between the types of agents, which are part of the reference architecture, establish the relationship contract between the agents.

Figure 2 shows one of the sequence diagrams that was produced. This diagram illustrates the interaction between the agents in the realization of an IoT Service according to the reference architecture.

After completion of the reference architecture design, a list of expected actions for its implementation was generated. The next lines show these actions, as well as the contributions and requirements related to them:

- Configuration interface implementation: It allows the existence of a visual interface for system administrator make the IoT Services configurations. It works as a link between system administrator and Facilitator Agent. Requirements met: 1, 2 and 4.

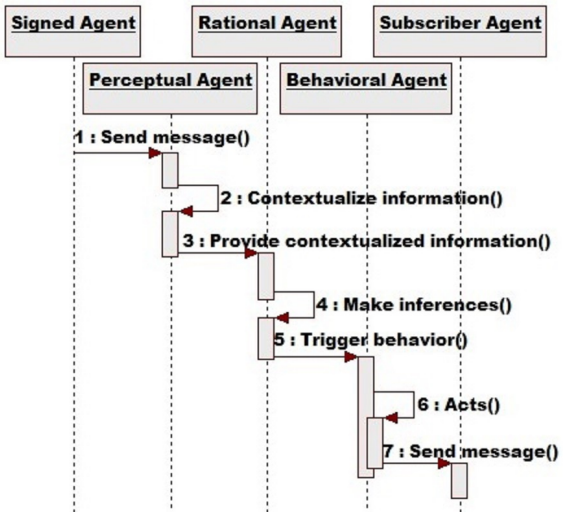

Figure 2: Agents interacting in the realization of an IoT Service.

- Facilitator Agent implementation: It allows the adaptability of the system through configurations changes and guides the communication between the agents. Requirements met: 1, 2, 3 and 9.
- Perceptual Agent implementation: It enables the contextualization of the received messages for the Rational Agents, based on the knowledge acquired about the status of the system. Requirements met: 3 and 5.
- Rational Agent implementation: It works as a rule base and a inference mechanism which can be removed and included dynamically in the system. Requirements met: 3 and 6.
- Behavioral Agent implementation: It works as a mechanism in charge of the actions performed by IoT Service. This mechanism can be removed and included dynamically in the system. Requirements met: 3 and 7.
- Implementation of the agents that represent the devices: They are the devices themselves, with the responsibilities defined by the reference architecture for the agents of the Basic type. These agents are actived by IoT Services or serve as data providers for the Perceptual Agents of the IoT Services. Requirements met: 3.
- Implementation of message structures used by agents: It contributes to the understanding of the message thanks to the standardization of attributes

of messages exchanged between agents. Requirements met: 8.

- Middleware implementation: Implementation of the elements that make possible the communication between agents (example: sockets infrastructure). Requirements met: 9

4 COMPOSITION OF THE RAILWAY MANAGEMENT SYSTEM ARCHITECTURE

Once the reference architecture has been defined, it needs to be applied and tested. This section contains a description of a concrete architecture derived from the reference architecture. The concrete architecture defines a prototype of a Railway Management System that follows the principles of the reference architecture.

4.1 Concrete Architecture Design

The concrete architecture definition follows the same steps of the MDA (CIM, PIM), that were used for the definition of the reference architecture, plus the Platform Specific Model (PSM) step, that has the definitions' details of the platform in which the system will be implemented.

The definition of the railway management services must follow the specification of the IoT Service agents defined by the reference architecture. For this purpose, each service defined by the concrete architecture must have Perceptual Agents, Rational Agents and Behavioral Agents that fulfill the responsibilities defined by the reference architecture's diagrams. In addition, the concrete architecture must have at least one Facilitator Agent and one Configuration Interface. And these must also carry out the defined responsibilities.

In the concrete architecture, agents with specific system features were defined. Some of these agents provide the information that feeds the IoT Services and others consume the data generated by these services. The consumption of these data triggers actions that modify the agents' environment.

The artifacts generated in the CIM and PIM stages of the MDA are similar to those generated in the composition of the reference architecture. In the PSM stage were generated class diagrams describing the basic infrastructure for the creation of agents that follow the reference architecture. This means that the diagrams describe the existing attributes and methods of the classes that represent the agents of Perceptual, Rational, Behavioral, Facilitator, and Basic types. The Basic Agent is an agent that is not part of the composition of the IoT Service. It can be a service's subscriber or can be signed by the service.

Details of the structures used for messages and middleware compositions (in the prototype the middleware is composed of TCP sockets) are also represented in the class diagrams.

A component diagram complements the CIM models by displaying the executables and libraries produced.

In MDA the transition of PIM to PSM is generally aided by tools. In this work this approach was not used. However, in order to guarantee the parity of the agents' internal actions with the responsibilities described by the diagrams of CIM and PIM phases, it was created a table that relates agent's methods with these responsibilities.

4.2 Concrete Architecture Elements

Because of the impossibility of obtaining a real environment to implement a Railway Management System, all physical elements of the prototype were simulated.

The agents that are part of the Railway Management System are as follow:

- Radio-Frequency Sensors: Basic Agents that receive signals from the transponders of the railway's Trains. This information, consisting of speed, identification and position, is sent to the Railway Management Services.
- Speed Radars: Basic Agents that determine the identity, speed, and position of the Trains passing through them. They forward this data to the Redundant Railway Management Service.
- Trains: Basic Agents that interact with the Railway Management Services and with other Trains. Each Train has a radio-frequency (RF) sensor, which receives messages from Railway Transmitters and from other Trains. Each Train also has a transponder, which sends data to Railway Sensors and to other Trains.
- Rail Actuators: Basic Agents that are in charge of changing the railway configuration, redirecting the Trains that travel through it. They respond to commands sent by the Railway Management Service.
- Radio-Frequency Transmitters: Basic Agents that receive, through a network connection, messages from the Railway Management Services, which are sent to the Trains through RF signals.

- Railway Management Service (RMS): Conceptual Service Agent composed of Perceptual Agent, Rational Agent and Behavioral Agents. This agent is aware of the railway situation thanks to messages received from the RF Sensors that inhabit it. Its function is to send control messages to Trains and Rail Actuators, with the intention of avoiding accidents (collisions). These messages instruct the Trains to change their speed and trigger the Rail Actuators redirecting the Trains if necessary.
- Redundant Railway Management Service (RRMS): Conceptual Service Agent with the same function as the RMS. However, the RRMS is aware of the railway situation through Speed Radars messages. This service comes into action when it is identified that the RMS is not aware of one of the Trains that passed through one of the RF Sensors. This failure may be caused by a interference with the RF signal sent by the Train, by a defect in the RF Sensor, or by a defect in the Train's own transponder. In order for the RRMS to reach this conclusion, it must also sign that it wishes to receive the notifications issued by the RF Sensors. Thus, it becomes able to know when a message is no longer sent by one of these Sensors in the same window of time when a message arrives from the corresponding Speed Radar. The RRMS hasn't the feature of changing Trains' direction, since it was only be included in the RMS so that the exchange of the reasoning machine and the inclusion of new behaviors can be tested in an existing IoT Service.
- Facilitator Agent: Agent that gives access to the information about the other agents of the system, allowing its reconfiguration through the inclusion and exclusion of agents and services data in the system. With the existence of a Configuration Interface, the system administrator can be able to access the Facilitator and perform these tasks, indicating the agents that should express interest in the messages sent by other agents. The Interface associated with the Facilitator also allows the configuration of the agents that will be part of the services perception, reasoning and behaviors. The Facilitator sends addresses data to the system agents. By this way, the agents are able to know where to send their messages.

5 PROTOTYPE IMPLEMENTATION AND TESTS

This section shows details of the prototype implementation and the tests created to verify the reference architecture effectiveness and the veracity of reference architecture requirements.

5.1 Used Tools

The following list shows the tools used for design and implementation of the Railway Management System prototype:

- StarUM 5.0.2.1570: used for design of the abstract and concrete architectures.
- Microsoft Visual Studio Community 2015: Integrated Development Environment (IDE).
- Windows 7 and Windows 8: operational systems.
- Mommosoft.ExpertSystem.dll: library for interaction with CLIPS (tool for construction of expert systems - used to manipulate the rules that compose the reasoning of the IoT Services' Rational Agents).
- Transfer Control Protocol/Internet Protocol (TCP/IP): communication protocol used for exchange messages between agents.

5.2 Test Description

The next topics includes the tests created to validate the reference architecture effectiveness in the accomplishing of IoT Services' Adaptability:

- Test 1 - Configure the RMS, including the data of the agents belonging to this service and the data of Trains, Sensors and Transmitters in the system, start the simulation and define the trains' speeds.
- Test 2 - With the RMS configured, manipulate Trains' speeds so that one Train arrives at a distance between 100m and 501m from the other Train.
- Test 3 - Configure the RRMS including the service's belonging agents and Speed Radar Agents data in the system.
- Test 4 - With the RRMS configured, proceed with the deactivation of the RF Sensors Agents through the simulation interface, and manipulate the Trains' speeds so that a Train arrives at a distance between 100m and 501m from the other Train.

- Test 5 - Exchange SlowDownBehavioralAgentRMS by SlowDownBehavioralAgentRMS2.
- Test 6 - With the deceleration behavior of the RMS switched, manipulate the Trains' speeds so that a Train arrives at a distance between 100m and 501m from the other Train.
- Test 7 - Exchange RationalAgentRMS by RationalAgentRMS2 and include RailActuatorA and RailActuatorB as RMS' behaviors.
- Test 8 - With the exchanging of the RMS reasoning and with the including of the Rail Actuators as RMS' behaviors, manipulate the Trains' speeds so that a Train arrives at a distance between 100m and 501m from the other Train.
- Test 9 - Use a distributed platform. To do so, change the IPs of the RF Sensors and the IP of the Configuration Interface to the IP of another machine in the same network and run these agents in this machine. Then reconfigure the agent's IPs in the Configuration Interface so that other agents know about the change.

5.3 Test Results

After the execution of the experiment, it was possible to know if the requirements raised in Table 1 of this work really had importance for the IoT Services' Adaptability.

Only one requirement (Existence of a common language structure) was classified as unimportant for the IoT Services' Adaptability. The belief that the knowledge of the structures of the messages exchanged between the agents was fundamental for them to communicate was unfounded. This knowledge helps to interpret the messages, but communication would not be possible without knowing which are the possible data contained in these structures. So, even if a message was a unformatted data stream, the knowledge of the possible contents of this message would already be sufficient for the receiving agent to make decisions based on it.

The reference architecture effectiveness corresponds to the quantity of requirements in Table 1 that were met in the execution of the tests. It was verified that this effectiveness are 100%, because based on this architecture it was possible to construct a system that met all these requirements.

In this experiment, it can be said that the effort to build a system that has the characteristics of the reference architecture that contribute to the IoT Services' Adaptability was overestimated. The time and work expended in design and implementation of the structures that define attributes and characteristics of the messages could be avoided, since this effort was considered as being of no importance for the IoT Services' Adaptability.

Of all actions listed in the research and required for design and construction of the prototype, 27 actions were considered necessary for creation of a system that minimally performs the reference architecture.

Two of these actions ("Design of the agents message structures" and "Implementation of the agents message structures") are related to the requirement "Existence of a common language structure", that was classified as unimportant for the IoT Services' Adaptability. If we establish the relationship between the actions that are important for the IoT Services' Adaptability and the actions that must be carried out to realize the reference architecture, it is possible to verify the effort that should be applied so that the reference architecture contributes with the adaptability of IoT Services:

$$\frac{25}{27} \times 100 = 92.5926\%$$

So, of all listed actions (100%), 7.4074% of these actions could not be executed.

6 CONCLUSIONS

The tests created to validate the architecture (Test Description subsection) aim to prove that a system built with this architecture allows the inclusion of a new IoT Service and the changing of the reasoning and the behavior of an IoT Service without damaging the other unchanged services' behaviors. All the tests related was executed and the system continued to work as expected.

The reconfiguration of the system' services has proven that the Interface works, that the configuration can be done at runtime without causing system damage (such as shutdown of a service) and that the software parts are actually independent.

The correct system operation, decelerating, accelerating and changing Trains' direction, means that the messages was received by IoT Services, that the Perceptual Agents was successful in contextualizing data for the Rational Agents, that the Rational Agents was successful in selecting the correct behaviors and that the Behavioral Agents was successful in sending data commands to the Trains.

The two paragraphs above show that IoT Services' Adaptability was achieved by following the reference architecture in the design and implementation of the Railway Management System prototype. However

there are important observations that must be taken into account.

The scalability and security requirements were not addressed. This means that in situations where it is necessary to create hundreds of agents or situations where it is necessary to assign agent access control mechanisms and message encryption, the reference architecture may need some adaptations.

The research tested the reference architecture through a prototype where all physical elements, such as sensors, actuators and trains, were simulated by software agents. It is important to remember that in real (non-simulated) systems, devices may have restrictions on their memory and processing capacities and restrictions on the required technology (compilers, programming languages, operating systems, etc.) for the program to be embedded.

An important characteristic of the architecture adopted is the division of IoT Service obligations into agents with well defined functions, thus separating perception, reasoning and behavior - elements visible in the BDI (Belief-Desire-Intention) logic. This gives flexibility so that an IoT Service can be changed without it all ceasing to work; For example: when the RMS' deceleration behavior was changed, the accelerate behavior was not impaired. If new reasoning and new behaviors were included in the system, the old ones would continue working and coexisting with the new ones. This is important in cases where is desirable to extend the features of an IoT Service or to change their characteristics.

One of the resources adopted so that the behavior of the IoT Service can have its independence extended in relation to the reasoning of a service is the adoption of the concept of context. In this way, certain similar behaviors can be created using the same context allowing its substitution without the necessity of Rational Agent substitution. For example: assuming that the system's administrator wants to replace a behavior that requests the speed reduction of a Train by 10% of its speed by a behavior that requests the reduction of that speed in 30%; in this case, it would be necessary to replace the reasoning of the same service if this reasoning would trigger behavior based on behavior's name. In the cases that the reasoning triggers a behavior by its context, it would be enough to disable the current behavior and to include the new behavior data in the system, since the two behaviors have the same context that, simply, can be the "slowdown" string.

Other element relevant of the architecture is the Facilitator Agent. It's plays an important role in establishing communication between agents by indicating the addresses of the subscribing agents to the signed agents. This agent is also important because it maintains the information of the agents that compose the system and because it allows the system reconfiguration based on the requests of user interface.

REFERENCES

Ayala, I., Amor, M., and Fuentes, L. (2012). Exploiting dynamic weaving for self-managed agents in the iot. In *Timm and C. Guttmann (Eds.): MATES 2012, LNAI 7598*. Springer-Verlag Berlin Heidelberg.

Dimakis, N., Soldatos, J., Polymenakos, L., Schenk, M., Pfirmann, U., and Brkle, A. (2006). Perceptive middleware and intelligent agents enhancing service autonomy in smart spaces. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT06)*. IEEE Computer Society.

Elammari, M. and Issa, Z. (2013). Using model driven architecture to develop multi-agent systems. The International Arab Journal of Information Technology, Vol. 10, No. 4.

Kim, D., Lee, G., Lee, K., Heo, S., Choi, K., and Shin, D. (2012). Design and implementation of efficient directory facilitator for context-aware service discovery. In *N. T. Nguyen et al. (Eds.): KES-amsta 2007, Lnai 4496*. Springer-Verlag Berlin Heidelberg.

Leppnen, T. and Riekki, J. (2013). A lightweight agent-based architecture for the internet of things. In *Technical Report, 2013*. Departamento de Cilncia da Computao e Engenharia, Universidade de Oulu, Oulu, Finlndia. IEICE - The Institute of Electronics, Information and Communication Engineers.

Leppnen, T., Riekki, J., Liu, M., Harjula, E., and Ojala, T. (2014). Mobile agents-based smart objects for the internet of things. In *Internet of Things Based on Smart Objects, Internet of Things, DOI*. Springer International Publishing Switzerland.

Mzahm, A., Ahmad, M., and Tang, A. (2014). Enhancing the internet of things (iot) via the concept of agent of things (aot). In *Journal of Network and Innovative Computing*. MIR Labs.

Vermesan, O., Friess, P., Guillemin, P., Giaffreda, R., Grindvoll, H., Eisenhauer, M., Serrano, M., Moessner, K., Spirito, M., Blystad, L., and Tragos, E. (2015). Paper templates. In *Internet of Things beyond the Hype: Research, Innovation and Deployment*. IERC Cluster SRIA.